

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ
ЗАХИСТУ ЕЛЕКТРОННИХ ЛИСТІВ
SOFTWARE IMPLEMENTATION OF THE
EMAIL PROTECTION SYSTEM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-121сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БІЛЯК В.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
« » 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БІЛЯКА Владислава Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація системи захисту електронних листів. Software Implementation of the Email Protection System

керівник проєкту (роботи) д-р філософ., ст.викл. Леоценко Сергій Дмитрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «24» квітня 2024 року
№ 168

2. Строк подання студентом проєкту (роботи) 07 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., ст.викл.		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання «24» квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

Владислав БІЛЯК
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Сергій ЛЕОЩЕНКО
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
105 с., 6 табл., 20 рис., 4 дод., 24 джерел.

ЕЛЕКТРОННА ПОШТА, ЗАХИСТ, ШИФРУВАННЯ, INTEL SGX,
ПОШТОВИЙ КЛІЄНТ, ЗАСТОСУНОК, РОЗРОБКА, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес програмної реалізації системи захисту електронних листів.

Предмет дослідження – програмні засоби для реалізації системи захисту електронних листів.

Метою роботи є програмна реалізація системи захисту електронних листів.

Матеріали, методи та технічні засоби: мова програмування C++, технологія Intel SGX, поштовий клієнт Microsoft Outlook, засоби моделювання UML.

Результати. Розроблено застосунок – систему захисту електронних листів.

Практична цінність роботи полягає у програмній реалізації системи захисту електронних листів у вигляді надбудови, що може бути застосована для поштових клієнтів.

Висновки. Реалізовано систему захисту електронних листів у вигляді надбудови, що може бути застосована для поштових клієнтів.

Галузь використання – користувачі інтернету.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 105 pages, 6 tables, 20 figures, 4 appendixes, 24 sources.

EMAIL, SECURITY, ENCRYPTION, INTEL SGX, EMAIL CLIENT, APPLICATION, DEVELOPMENT, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of software implementation of the email protection system.

The subject of the research is software tools for implementing an email protection system.

The aim of this work is to provide a software implementation of the email protection system.

Materials, methods, and technical tools: C++ programming language, Intel SGX technology, Microsoft Outlook email client, UML modeling tools.

Results. An application has been developed – an email protection system.

The practical value of the work lies in the software implementation of the email protection system in the form of an add-on that can be applied to email clients.

Conclusions. Implemented an email protection system in the form of an add-on that can be used for email clients.

Scope of use – internet users.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області	15
1.1 Захист конфіденційності електронного листування	15
1.2 Сервіси захисту конфіденційності електронного листування в поштових клієнтах	16
1.3 Порівняння існуючих підходів захисту.....	24
1.4 Постановка завдання	30
1.4.1 Функціональність системи	30
1.4.2 Вимоги до інтерфейсу.....	33
1.4.3 Вимоги до безпеки	33
1.4.4 Не функціональні вимоги до продукту	33
2 Матеріали і методи	34
2.1 Технологія Intel SGX	34
2.2 Загальний опис захисних механізмів технології Intel SGX.....	35
2.3 Модель програмування технології Intel SGX	36
2.4 Компоненти апаратних засобів технології Intel SGX	39
2.5 Вибір мови програмування	43
2.6 Вибір середовища розробки	47
2.7 Структурна схема розробки.....	50
3 Основні рішення щодо реалізації компонентів системи	52
3.1 Операції над анклавом в технології Intel SGX	52
3.2 Робота з секретними даними в технології Intel SGX	54
3.2.1 Вимірювання анклаву	57
3.2.2 Атестація анклаву.....	58
3.2.3 Локальна атестація анклаву	59
3.2.4 Дистанційна атестація анклаву	62
3.3 Запечаткування	64

3.4 Архітектура розробленого програмного забезпечення	67
4 Експлуатація, тестування та експериментальне дослідження програми	71
4.1 Призначення й умови застосування програми	71
4.2 Вимоги до інтерфейсу	72
4.3 Умови виконання програми.....	72
4.4 Методика та результати тестування	72
Висновки	76
Перелік джерел посилань	77
Додаток А Технічне завдання	79
Додаток Б Опис програми	83
Додаток В Текст програми	87
Додаток Д Слайди презентації	99

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;
BIOS – Basic Input Output System;
CPU – Central Processing Unit;
EPC – Enclave Page Cache;
EPCM – Enclave Page cache Map;
IKE – Internet Key Exchange;
MAC – Message Authentication Code;
MEE – Memory Encryption Engine;
PAVP – Protected Audio Visual Path;
PGP – Pretty Good Privacy;
PMH – Page Miss Handler;
PRM – Processor Reserved Memory;
SMIME – Secure Multipurpose Internet Mail Extensions;
SHA – Secure Hash Algorithm;
SVN – Security Version Number;
TCB – Trusted Computing Base;
TCS – Thread Control Structure;
ОС – операційна система;
ПЗ – програмне забезпечення.

ВСТУП

Електронна пошта є основним засобом комунікації як в особистій, так і в професійній сфері. Однак її широке використання також робить її головною мішенню для кіберзагроз. Традиційні заходи безпеки електронної пошти, хоча і ефективні певною мірою, стають все більш недостатніми перед обличчям нових загроз [1]–[3].

Фішинг є однією з найбільш поширених загроз, пов'язаних з електронною поштою. Зловмисники використовують шахрайські електронні листи, щоб змусити одержувачів розкрити конфіденційну інформацію, таку як облікові дані для входу, фінансові дані або особисту інформацію. Точковий фішинг, більш цілеспрямована форма фішингу, включає персоналізовані атаки, спрямовані на конкретних людей або організації [1]–[3]. Ці складні атаки часто використовують методи соціальної інженерії, щоб здаватися законними, що робить їх особливо небезпечними.

Зловмисне програмне забезпечення та програми-вимагачі часто доставляються за допомогою вкладень електронної пошти або шкідливих посилань. Після запуску шкідливе програмне забезпечення може красти дані, відстежувати активність користувачів або надавати несанкціонований доступ до систем. Програма-вимагач шифрує дані жертви, вимагаючи викуп за їх передачу. Ці атаки можуть призвести до значних фінансових втрат і збоїв у роботі [1]–[3].

Атаки з метою бізнес-компрометації включають в себе видачу себе за високопоставленого керівника або довіреного ділового партнера, щоб обманом змусити співробітників здійснювати несанкціоновані фінансові транзакції або розголошувати конфіденційну інформацію. Ці атаки є цілеспрямованими і можуть призвести до значних фінансових втрат і репутаційного збитку [1]–[3].

Підrobка адреси електронної пошти передбачає підrobку адреси відправника, щоб електронний лист виглядав так, ніби він надійшов із

надійного джерела. Пов'язана з цим загроза-уособлення домену-передбачає реєстрацію доменів, схожих на законні, для обману одержувачів. Ці методи часто використовуються при фішингу та компрометаційних атаках.

Нажаль, іноді користувачі стикаються із певною обмеженістю традиційних заходів безпеки електронної пошти [1]–[3].

Спам-фільтри призначені для виявлення та блокування небажаних електронних листів. Хоча вони ефективно фільтрують масовий спам, їм часто важко виявити складні фішингові розсилки. Зловмисники постійно вдосконалюють тактику обходу цих фільтрів, що з часом робить їх менш надійними [1]–[3].

Антивірусне програмне забезпечення сканує вкладення електронної пошти на наявність відомих підписів шкідливих програм. Однак воно часто неефективно проти нових, невідомих видів шкідливих програм або складних атак, в яких не задіяні традиційні шкідливі програми. Це обмеження підкреслює необхідність кращих методів виявлення [1]–[3].

Протоколи перевірки автентичності електронної пошти, такі як:

- Sender Policy Framework;
- DomainKeys Identified Mail
- Domain-based Message Authentication, Reporting & Conformance.

Допомагають перевірити справжність відправників електронної пошти. Хоча ці протоколи необхідні, вони не є надійними і можуть бути обійдені досвідченими зловмисниками.

Ситуація з кіберзагрозами постійно змінюється, і зловмисники розробляють нові методи обходу існуючих заходів безпеки. Оскільки електронна пошта залишається основним джерелом кібератак, вкрай важливо розробляти нові методи та підходи, щоб протистояти цим загрозам. Використання виключно традиційних заходів безпеки робить організації вразливими перед виникаючими загрозами [1]–[3].

Зловмисники використовують все більш досконалі тактики, часто використовуючи штучний інтелект та машинне навчання для посилення своїх

атак. Наприклад, штучний інтелект можна використовувати для створення дуже переконливих фішингових електронних листів або для виявлення потенційних вразливостей в системі захисту організації. Щоб протистояти цим сучасним загрозам, рішення безпеки також повинні використовувати штучний інтелект та машинне навчання [1]–[3].

Організації повинні дотримуватися різних нормативних вимог щодо захисту даних та кібербезпеки. Немоżliвість захистити конфіденційну інформацію може призвести до серйозних юридичних та фінансових наслідків. Розробка нових методів захисту електронної пошти допомагає організаціям відповідати цим вимогам і захищати своїх зацікавлених осіб [1]–[3].

Успішні атаки на електронну пошту можуть призвести до значних фінансових втрат, будь то в результаті прямої крадіжки, виплати викупу або збоїв в роботі. Крім того, шкода репутації внаслідок порушення безпеки може мати довгострокові наслідки для довіри до організації. Інвестиції в передові заходи безпеки електронної пошти необхідні для зменшення цих ризиків [1]–[3].

Відповідно, постійно зростає потреба в нових методах і підходах.

Штучний інтелект та машинне навчання здатні революціонізувати безпеку електронної пошти. Ці технології дозволяють аналізувати величезні обсяги даних для виявлення закономірностей і аномалій, що вказують на шкідливу активність. Наприклад, поштові фільтри на базі штучного інтелекту можуть виявляти фішингові листи, аналізуючи їх зміст, структуру і метадані. Алгоритми машинного навчання також можуть адаптуватися та вдосконалюватися з часом, підвищуючи свою ефективність у боротьбі зі зміною загроз [1]–[3].

Поведінковий аналіз включає моніторинг поведінки користувачів для виявлення аномалій, які можуть вказувати на зламаний обліковий запис або триваючу атаку. Наприклад, якщо працівник раптом починає отримувати доступ до конфіденційних даних або надсилати електронні листи в незвичний час, це може бути ознакою злому облікового запису. Поведінковий аналіз

може забезпечити додатковий рівень безпеки шляхом виявлення потенційних загроз, які можуть бути пропущені традиційними заходами.

Передові рішення для захисту від загроз забезпечують багаторівневий підхід до забезпечення безпеки електронної пошти. Зазвичай вони включають такі функції, як сканування вкладень та посилань Електронної Пошти в режимі реального часу, ізольоване середовище для безпечного виконання та аналізу потенційно шкідливих файлів та розширену аналітику загроз для виявлення та блокування відомих загроз. Рішення призначені для захисту від широкого спектру загроз, пов'язаних з електронною поштою, включаючи атаки нульового дня [1]–[3].

Шифрування важливо для захисту конфіденційності та цілісності повідомлень електронної пошти. Наскрізне шифрування гарантує, що вміст електронної пошти зможе прочитати тільки той, кому воно призначене. Захищені поштові шлюзи (Secure Email Gateways – SEG) забезпечують додатковий рівень безпеки, фільтруючи вхідні та вихідні електронні листи на наявність загроз та застосовуючи політику безпеки. Ці технології допомагають захистити конфіденційну інформацію від перехоплення та несанкціонованого доступу [1]–[3].

Людська помилка є важливим фактором у багатьох успішних атаках електронної пошти. Навчання співробітників розпізнавати та реагувати на потенційні загрози має вирішальне значення для ефективного захисту електронної пошти. Регулярні тренінги з підвищення обізнаності про безпеку можуть допомогти співробітникам виявляти фішингові електронні листи, розуміти важливість надійних паролів і слідувати рекомендаціям щодо обробки конфіденційної інформації. Надання користувачам знань та навичок може значно зменшити ризик успішних атак [1]–[3].

Слід розглянути напрямки та інновації на майбутнє.

Модель безпеки з нульовою довірою заснована на принципі, згідно з яким за замовчуванням жодному об'єкту, будь то всередині мережі або за її межами, не можна довіряти. Впровадження архітектури безпеки електронної

пошти з нульовою довірою передбачає постійну перевірку автентичності та цілісності повідомлень електронної пошти. Такий підхід може допомогти запобігти несанкціонованому доступу та зменшити ризик успішних атак на основі електронної пошти.

Технологія блокчейн пропонує децентралізований і захищений від несанкціонованого доступу спосіб перевірки автентичності і цілісності повідомлень електронної пошти. Використовуючи блокчейн, можна створити безпечний, прозорий та незмінний запис транзакцій Електронної Пошти. Ця технологія може допомогти запобігти підробці електронної пошти та забезпечити цілісність вмісту електронної пошти, забезпечуючи додатковий рівень безпеки [1]–[3].

У міру розвитку квантових обчислень традиційні криптографічні методи можуть стати вразливими до квантових атак. Постквантова криптографія передбачає розробку алгоритмів шифрування, стійких до загроз квантових обчислень. Забезпечення безпеки електронної пошти в постквантовому світі вимагатиме впровадження цих передових криптографічних методів [1]–[3].

Поширення пристроїв Інтернету речей ставить нові завдання в області безпеки електронної пошти. Пристрої Інтернету речей часто не мають надійних засобів захисту, що робить їх потенційними мішенями для зловмисників. Інтеграція захисту електронної пошти з безпекою Інтернету речей може допомогти захиститися від загроз, які використовують уразливості в підключених пристроях [1]–[3]. Наприклад, захищені протоколи зв'язку та аутентифікація пристроїв можуть запобігти несанкціонованому доступу до облікових записів електронної пошти за допомогою зламаних пристроїв Інтернету речей.

Мінливий ландшафт загроз і зростаюча складність кібератак вимагають постійної розробки нових методів і підходів до забезпечення безпеки електронної пошти. Традиційних заходів безпеки більше недостатньо для боротьби зі складною і динамічною природою сучасних загроз, пов'язаних з

електронною поштою. Використовуючи нові технології, такі як штучний інтелект, ML, поведінковий аналіз та вдосконалене шифрування, організації можуть підвищити рівень безпеки своєї електронної пошти та краще захистити конфіденційну інформацію.

Крім того, навчання та інформованість користувачів є найважливішими компонентами комплексної стратегії забезпечення безпеки електронної пошти. Надання співробітникам знань і навичок для виявлення потенційних загроз і реагування на них може значно знизити ризик успішних атак [1]–[3].

У міру просування вперед інноваційні підходи, такі як архітектура з нульовим рівнем довіри, технологія блокчейн, постквантова криптографія та інтеграція Інтернету речей – безпеки, відіграватимуть вирішальну роль у захисті електронної пошти. Випереджаючи виникаючі загрози і постійно вдосконалюючи заходи безпеки, організації можуть знижувати ризики, захищати свої активи і підтримувати довіру до себе у все більш цифровому світі.

Отже актуальною задачею залишається програмна реалізація системи захисту електронних листів.

Для виконання такої задачі необхідно виконати наступні завдання:

- проаналізувати проблемну область захисту електронної пошти;
- дослідити існуючі механізми захисту електронного листування;
- визначити та обґрунтувати вибір технічних засобів для захисту електронного листування;
- спроектувати та реалізувати застосунок для захисту листування;
- протестувати нову технологію захисту електронного листування;
- підготувати висновки на основі результатів тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Захист конфіденційності електронного листування

Електронна пошта є одним з найбільш поширених сервісів мережі Інтернет. Вона створює можливість спілкування та обміну будь-якими типами даних незалежно від географічних відстаней. У момент своєї появи електронна пошта зробила зв'язок між людьми набагато швидше, і зручніше. В даний час існує безліч технологій, що забезпечують обмін інформацією в мережі Інтернет, проте популярність електронної пошти не зменшується. Причиною цього є її простота і надійність. Кількість користувачів електронної пошти обчислюється мільярдами, десятки мільярдів повідомлень передаються по електронній пошті щодня. Велика частина обміну інформацією в багатьох компаніях і між ними ведеться по електронній пошті. Багато даних, що передаються по ній є секретними. Таким чином, проблема забезпечення конфіденційності електронної пошти є досить актуальною [1]–[3].

Технологія Intel SGX з'явилася в 2015-му році в процесорах компанії Intel шостого покоління. Ця технологія розроблена для захисту вихідного коду та даних від несанкціонованого доступу та зміни [1]. Важливою особливістю Intel SGX є те, що захист здійснюється, в тому числі від шкідливого програмного забезпечення (ПЗ), запущеного з більш високими привілеями, наприклад виконуваного з ядра операційної системи (ОС), BIOS або будь-якого драйвера. Таким чином, технологія надає нові можливості щодо забезпечення конфіденційності даних в різних додатках. У зв'язку з новизною технології вона ще не отримала широкого застосування і її використання для забезпечення конфіденційності електронної пошти є актуальною і ще не дослідженою областю.

Метою даної роботи є забезпечення конфіденційності електронної пошти з використанням технології Intel SGX шляхом захисту призначених для користувача даних і ключів шифрування на стороні клієнта. Завданнями даної роботи є аналіз загроз конфіденційності електронної пошти, дослідження

можливостей технології Intel SGX щодо усунення знайдених загроз, розробка архітектури програмного забезпечення, що використовує знайдені можливості і створення прототипу програмного забезпечення за розробленою архітектурою [1]–[3].

1.2 Сервіси захисту конфіденційності електронного листування в поштових клієнтах

Електронна пошта залишається наріжним каменем цифрової комунікації, надаючи першорядне значення її безпеці та конфіденційності. Популярні поштові сервіси, такі як Gmail, Yahoo Mail і Outlook (раніше відомий як Bing Mail), використовують ряд технологій для захисту листування користувачів [1]–[3].

Базовий підхід – технології шифрування.

Безпека транспортного шару (TLS) – це стандартний протокол безпеки, який використовується Gmail, Yahoo Mail та Outlook для шифрування даних електронної пошти, що передаються між поштовими серверами та клієнтами. TLS запобігає прослуховуванню та втручанню під час передачі даних шляхом шифрування каналу зв'язку. Це гарантує, що електронні листи не можуть бути перехоплені або змінені зловмисниками під час передачі [1]–[3].

Переваги підходу:

- шифрує передані дані, забезпечуючи конфіденційність і цілісність;
- широко підтримується і реалізується в різних поштових службах і клієнтах;
- забезпечує безпечне з'єднання, не вимагаючи втручання користувача.

Недоліки:

- не захищає електронні листи в режимі очікування (збережені електронні листи);

– залежить від поштових серверів відправника і одержувача, що підтримують протокол TLS.

Наскрізне шифрування (E2EE) гарантує, що вміст електронної пошти зможе прочитати тільки той, кому воно призначене. Такі сервіси, як Gmail, підтримують E2EE за допомогою розширень або сторонніх додатків, таких як ProtonMail або Pretty Good Privacy (PGP). E2EE шифрує вміст електронної пошти на пристрої відправника і може бути розшифрований лише одержувачем, гарантуючи, що навіть постачальники послуг електронної пошти не зможуть читати електронні листи [1]–[3].

Переваги підходу:

- забезпечує найвищий рівень конфіденційності вмісту електронної пошти;
- запобігає доступу постачальників послуг і третіх осіб до даних електронної пошти.

Недоліки:

- потрібно, щоб як відправник, так і одержувач використовували сумісні засоби шифрування;
- для користувачів, які не мають технічних знань, це може бути складним у впровадженні та управлінні;
- підтримка основних постачальників послуг електронної пошти обмежена.

Технології аутентифікації та верифікації.

Аутентифікація доменних повідомлень, створення звітів та забезпечення відповідності (DMARC) – це протокол автентифікації електронної пошти, який допомагає запобігти підробці електронної пошти шляхом узгодження протоколів. Gmail, Yahoo Mail та Outlook підтримують DMARC для перевірки того, що вхідні електронні листи надходять із законних джерел, що захищає користувачів від фішингу та атак, пов'язаних з видачею себе за іншого [1]–[3].

Переваги підходу:

- підвищує ефективність перевірки автентичності електронної пошти за рахунок поєднання SPF і DKIM;
- зменшує кількість випадків фішингу та підміни електронної пошти;
- надає функції створення звітів для моніторингу та підвищення безпеки електронної пошти.

Обмеження:

- вимагає належного налаштування власниками доменів;
- не шифрує вміст електронної пошти.

Структура політики відправників (SPF) – це протокол, який дозволяє власникам доменів вказувати, яким поштовим серверам дозволено надсилати електронні листи від їх імені. Служби електронної пошти, такі як Gmail, Yahoo Mail та Outlook, використовують SPF для узгодження IP-адреси відправника із записом SPF домену, що допомагає запобігти підробці та несанкціонованому надсиланню електронної пошти [1]–[3].

Переваги підходу:

- допомагає запобігти підробці електронної пошти шляхом перевірки IP-адрес відправників;
- простота впровадження та обслуговування;
- широко підтримується найбільшими постачальниками послуг електронної пошти.

Обмеження:

- не захищає вміст електронної пошти або її цілісність;
- вимагає належного налаштування та обслуговування власниками доменів.

DomainKeys Identified Mail (DKIM) – додає цифровий підпис до вихідних електронних листів, дозволяючи службі електронної пошти одержувача перевірити, чи Електронна пошта була надіслана з авторизованого поштового сервера і не змінювалася. Gmail, Yahoo Mail та Outlook

використовують DKIM для підвищення автентичності та цілісності електронної пошти [1]–[3].

Переваги підходу:

- перевіряє справжність і цілісність електронних листів;
- допомагає запобігти підробці електронної пошти;
- широко використовується великими постачальниками послуг електронної пошти.

Недоліки:

- вміст електронної пошти не шифрується;
- власникам доменів потрібно налаштувати та підтримувати записи DKIM.

Розширений захист від загроз.

Фільтри небажаної пошти та виявлення фішингу – необхідні для захисту користувачів від шкідливих електронних листів. Gmail, Yahoo Mail та Outlook використовують вдосконалені алгоритми машинного навчання для аналізу вмісту електронної пошти, метаданих та поведінки користувачів для виявлення та блокування спроб спаму та фішингу [1]–[3].

Переваги:

- автоматично розпізнає та блокує шкідливі електронні листи;
- знижує ризик фішингових атак;
- постійно вдосконалюється завдяки машинному навчанню.

Обмеження:

- час від часу законні електронні листи можуть помилково класифікуватися як спам (помилкові спрацьовування);
- для захисту від нових загроз потрібне постійне оновлення.

Перевірка вкладень і посилань – служби електронної пошти перевіряють вкладення та посилання на наявність шкідливого програмного забезпечення та фішингового вмісту, перш ніж вони потраплять до одержувача. Gmail, Yahoo Mail та Outlook використовують передові технології

захисту від загроз для виявлення та блокування шкідливого вмісту в режимі реального часу [1]–[3].

Переваги:

- захищає користувачів від шкідливих програм і фішингових атак;
- автоматично сканує і блокує шкідливий контент;
- підвищує загальну безпеку електронної пошти.

Обмеження:

- сканування іноді може призвести до затримки доставки електронної пошти;
- може не виявляти складні загрози.

Функції конфіденційності.

Двофакторна автентифікація (2FA) – підвищує рівень безпеки облікових записів електронної пошти, вимагаючи від користувачів надання двох форм підтвердження. Gmail, Yahoo Mail та Outlook пропонують 2FA для захисту облікових записів користувачів від несанкціонованого доступу, навіть якщо паролі порушені [1]–[3].

Переваги:

- значно підвищує безпеку облікового запису;
- захищає від атак, заснованих на паролі;
- легко включається і використовується.

Обмеження:

- вимагає, щоб користувачі мали доступ до додаткового пристрою або методу перевірки;
- це може викликати незручності, якщо додатковий пристрій втрачено або недоступно.

Захист від втрати даних (DLP) – технології захисту від втрати даних (DLP) допомагають запобігти передачі конфіденційної інформації за межі організації. Gmail, Yahoo Mail та Outlook пропонують функції DLP для виявлення та блокування електронних листів, що містять конфіденційні дані, забезпечуючи дотримання правил захисту даних [1]–[3].

Переваги підходу:

- захищає конфіденційну інформацію від випадкового або несанкціонованого розповсюдження;
- допомагає організаціям дотримуватися правил захисту даних;
- політики, що налаштовуються для задоволення конкретних потреб в області безпеки.

Обмеження:

- вимагає належного налаштування та обслуговування;
- не всі порушення даних можна запобігти, особливо через канали, відмінні від електронної пошти.

Навчання користувачів та підвищення їх обізнаності.

Програми підвищення обізнаності про безпеку – служби електронної пошти часто надають ресурси та інструменти для ознайомлення користувачів з рекомендаціями щодо безпеки електронної пошти. Gmail, Yahoo Mail та Outlook пропонують програми поінформованості про безпеку, включаючи імітацію фішингу та навчальний контент, які допомагають користувачам розпізнавати та реагувати на потенційні загрози [1]–[3].

Переваги:

- дозволяє користувачам розпізнавати фішингові атаки та запобігати їм;
- знижує ризик людської помилки при захисті електронної пошти;
- підвищує загальну безпеку.

Недоліки:

- потрібне безперервне навчання;
- відповідність вимогам і залученість користувачів можуть варіюватися.

Тематичні дослідження та приклади з реального життя.

Розширена програма захисту Google – програма розширеного захисту Google призначена для користувачів, яким потрібен найвищий рівень безпеки, таких як журналісти, активісти та учасники політичних кампаній. Ця програма

пропонує розширені функції безпеки, включаючи фізичні ключі безпеки для 2FA, більш суворі процедури відновлення облікового запису та розширене виявлення шкідливих програм [1]–[3].

Переваги заходу:

- забезпечує надійний захист для користувачів, схильних до високого ризику;
- підвищує безпеку облікового запису за допомогою фізичних ключів безпеки;
- включає розширені можливості виявлення загроз і захисту.

Обмеження:

- може бути складною і незручною для звичайних користувачів;
- потрібно, щоб користувачі отримували фізичні ключі безпеки та керували ними.

Функція шифрування електронної пошти в Yahoo – компанія Yahoo Mail у співпраці з такими партнерами, як ProtonMail, експериментувала з сервісами шифрування електронної пошти. Ця функція забезпечує наскрізне шифрування, гарантуючи, що вміст електронної пошти зможуть прочитати тільки ті, кому воно призначене [1]–[3].

Переваги:

- підвищує конфіденційність Електронної Пошти завдяки наскрізному шифруванню;
- захищає вміст електронної пошти від несанкціонованого доступу.

Обмеження:

- обмежена доступність та впровадження;
- потрібно, щоб як відправник, так і одержувач використовували Сумісні засоби шифрування.

Напрямки та інновації на майбутнє.

Шифрування, стійке до квантових впливів – у міру розвитку квантових обчислень традиційні методи шифрування можуть стати вразливими. Служби електронної пошти, такі як Gmail, Yahoo Mail та Outlook, досліджують та

розробляють квантово стійкі алгоритми шифрування для довгострокової безпеки [1]–[3].

Переваги підходу:

- забезпечує безпеку електронної пошти перед обличчям загроз, пов'язаних з квантовими обчисленнями;

- перспективні методи шифрування;

Недоліки:

- все ще знаходиться на стадії досліджень і розробок;

- потрібне широке впровадження.

Штучний інтелект та машинне навчання.

Штучний інтелект та машинне навчання все частіше використовуються для підвищення безпеки електронної пошти. Ці технології дозволяють аналізувати величезні обсяги даних для виявлення закономірностей і аномалій, покращуючи виявлення фішингу і шкідливих програм. Gmail, Yahoo Mail і Outlook використовують штучний інтелект і машинне навчання для постійного вдосконалення своїх заходів безпеки [1]–[3].

Переваги технології:

- покращує виявлення загроз і реагування на них;

- постійно вдосконалюється за рахунок вивчення нових загроз;

- знижує кількість помилкових спрацьовувань і покращує роботу користувачів.

Обмеження:

- вимагає значних обчислювальних ресурсів;

- це може бути важко реалізувати та підтримувати.

Технологія блокчейн пропонує децентралізований і захищений від несанкціонованого доступу спосіб перевірки автентичності і цілісності повідомлень електронної пошти. Майбутні служби електронної пошти можуть використовувати блокчейн для підвищення безпеки та конфіденційності [1]–[3].

Переваги:

- забезпечує безпечний і прозорий спосіб перевірки автентичності електронної пошти;
- запобігає підробці даних.

Обмеження:

- вимагає значних змін в існуючій інфраструктурі електронної пошти;
- проблеми з масштабованістю та продуктивністю.

Захист конфіденційності електронної кореспонденції має вирішальне значення в сучасному цифровому світі. Популярні служби електронної пошти, такі як Gmail, Yahoo Mail та Outlook, використовують ряд технологій для захисту електронної пошти користувачів, включаючи шифрування, автентифікацію, розширений захист від загроз та навчання користувачів. Хоча ці технології забезпечують значний захист, мінливий ландшафт загроз вимагає постійного вдосконалення та інновацій [1]–[3].

1.3 Порівняння існуючих підходів захисту

Для того щоб переконатися в тому, які загрози усуваються існуючими засобами захисту, розглянуто кілька найбільш поширених представників [4], [5].

Першим розглянутим засобом є стандарт S/MIME (Secure / Multipurpose Internet Mail Extensions). Цей стандарт використовує асиметричне шифрування для забезпечення конфіденційності електронної пошти. Його основною перевагою є те, що він підтримується більшістю поштових клієнтів без установки будь-яких плагінів. При його використанні повідомлення в поштовому клієнті зашифровані до моменту його розшифрування. Недоліками є необхідність генерувати сертифікат відкритого ключа окремою програмою і загальна складність налаштування поштового клієнта. Для полегшення роботи з сертифікатами існує плагін Cybersafe Mail для поштового клієнта Microsoft Outlook [4], [5].

Ще один спосіб захисту електронної пошти надає стандарт OpenPGP (Open Pretty Good Privacy), який також використовує асиметричне шифрування. За допомогою PGP можна створювати пари ключів, захищати закритий ключ паролем і використовувати ключі для шифрування і підписання повідомлень. Програми на основі PGP також реалізують сервери зберігання даних, які дозволяють отримувати відкриті ключі інших користувачів та надавати свій відкритий ключ [4], [5].

Однією з реалізацій стандарту є програма PGP Desktop, крім переваг стандарту вона володіє перевагою сумісності з будь-яким поштовим клієнтом. Іншою реалізацією є плагін для поштового клієнта Enigmail. Реалізований для поштових клієнтів Thunderbird і Postbox. Володіє перевагами стандарту і зручний у використанні. Крім того існують безпечні поштові сервіси, такі як HushMail. HushMail реалізує стандарти SSL та OpenPGP, при цьому всі криптографічні операції виконуються на сервері [4], [5].

Так само захист конфіденційності електронної пошти може забезпечуватися плагінами для браузерів. Прикладом такого плагіна є SecureGmail, який реалізує симетричне шифрування, тобто шифрує кожне секретне повідомлення паролем, який повинні знати і відправник, і одержувач. Така система шифрування може використовуватися при повній довірі між її учасниками. Важливо, що при зростанні кількості учасників зростає і кількість ключів, які необхідно пам'ятати. Плагін SecureGmail створений для браузера Chrome. Аналогічний плагін для браузера Firefox носить назву Encrypted Communication. Основна перевага подібних плагінів-простота використання, так як для їх використання не потрібно розуміти ідею асиметричного шифрування і відсутні проблеми пов'язані з сертифікатами ключів [4], [5].

Порівняємо існуючі підходи за наступними критеріями:

- тип шифрування – метод, який використовується для шифрування вмісту електронної пошти. Шифрування гарантує, що електронний лист зможе прочитати лише той, кому він призначений. Розуміння типу шифрування допомагає оцінити рівень безпеки та зручність використання технології.

Асиметричне шифрування (відкритий/приватний ключ) є більш безпечним для обміну повідомленнями електронної пошти порівняно з симетричним (на основі пароля);

- вимоги до сертифікатів – вказує, чи потрібні цифрові сертифікати, видані центром сертифікації, для шифрування та дешифрування. Сертифікати мають вирішальне значення для перевірки особи відправника та одержувача, тим самим запобігаючи персоніфікації. Знання того, чи потрібні сертифікати, допомагає оцінити складність налаштування та рівень безпеки;

- інтеграція – наскільки добре технологія інтегрується з різними поштовими клієнтами та системами. Простіша інтеграція означає більш широке використання та адаптацію. Користувачі віддають перевагу технологіям, які без проблем працюють з існуючими клієнтами електронної пошти;

- простота використання – простота, з якою користувачі можуть налаштувати та використовувати технологію шифрування. Висока простота використання сприяє впровадженню серед користувачів, які можуть не володіти технічними знаннями. Технології, які складні у використанні, можуть бути менш затребувані;

- сумісність – набір поштових клієнтів і платформ, що підтримують дану технологію. Широка сумісність гарантує, що користувачі зможуть використовувати технологію шифрування на різних поштових сервісах і пристроях, що підвищує її практичність;

- ідентифікація – можливість підтвердити особу відправника та переконатися, що електронний лист не був підроблений. Автентифікація запобігає підробці та гарантує одержувачу, що електронний лист надсилається законним відправником і не змінюється;

- управління ключами – як створюються, зберігаються і поширюються ключі шифрування. Ефективне управління ключами має вирішальне значення для підтримки безпеки. Ключі, керовані Користувачем,

можуть бути більш гнучкими, але складними, тоді як ключі, керовані центром сертифікації, забезпечують стандартизоване, але потенційно дороге рішення;

- рівень безпеки – загальна ефективність технології в захисті конфіденційності та безпеки електронної пошти. Показує, наскільки добре технологія може захистити від різних загроз. Більш високий рівень безпеки означає кращий захист конфіденційної інформації;

- шифрування вмісту електронної пошти – чи шифрує технологія весь вміст електронної пошти, включаючи вкладення. Для забезпечення повної конфіденційності важливо забезпечити шифрування всього електронного листа, включаючи вкладення;

- захист метаданих електронної пошти – чи зашифровані заголовки електронної пошти (наприклад, тема, відправник, одержувач). Захист метаданих має вирішальне значення для забезпечення конфіденційності, оскільки заголовки можуть розкривати конфіденційну інформацію, навіть якщо вміст електронної пошти зашифровано;

- прийняття користувачами – ступінь, в якій технологія використовується широкою громадськістю або певними групами. Високий ступінь впровадження користувачами означає, що технологія надійна і ефективна в своїх передбачуваних випадках використання;

- складність налаштування – складність, пов'язана з налаштуванням технології шифрування. Технології з меншою складністю налаштування, швидше за все, будуть прийняті ширшою аудиторією, включаючи користувачів, які не мають технічних знань;

- відкликання – можливість відкликання ключів шифрування в разі їх злому. Відкликання ключів важливе для забезпечення безпеки у разі втрати або компрометації ключів;

- вартість – фінансові витрати, пов'язані з використанням технології, включаючи будь-які необхідні сертифікати або програмне забезпечення. Вартість може бути важливим фактором для користувачів та організацій при виборі рішення для шифрування;

– резервне копіювання та відновлення – доступні методи резервного копіювання та відновлення ключів шифрування. Забезпечення надійного резервного копіювання та відновлення ключів має вирішальне значення для збереження доступу до зашифрованих електронних листів;

– додаткові функції – будь-які інші помітні функції, які покращують функціональність технології, такі як відмова від використання або мітка часу. Додаткові функції можуть забезпечити додаткові переваги в плані безпеки і зробити технологію більш привабливою для користувачів.

Представимо результати у вигляді таблиці – табл. 1.1.

Таблиця 1.1 – Результати порівняння

Критерій	S/MIME	OpenPGP	SecureGmail
1	2	3	4
Тип шифрування	Асиметричний	Асиметричний	Симетричний
Вимоги до сертифікату	Потрібні цифрові сертифікати	Використовує самопідписані сертифікати	Сертифікати не потрібні
Інтеграція	Вбудована підтримка в більшості поштових клієнтів	Потрібні плагіни	Розширення браузера для Gmail
Простота використання	Помірна складність	Від помірної до високої складності	Висока простота використання
Сумісність	Широкий спектр	Широкий спектр	Доступно лише для Gmail та Chrome
Ідентифікація	Так, забезпечує підписання електронної пошти та шифрування	Так, забезпечує підписання електронної пошти та шифрування	Немає цифрових підписів (шифрування на основі пароля)
Управління ключами	Управляється центрами сертифікації	Керований користувачем	Немає управління ключами
Модель довіри	Ієрархічний	Web of Trust	Довіра до спільного паролю
Рівень безпеки	Високий	Високий рівень	Помірний

Продовження таблиці 1.1

1	2	3	4
Шифрування вмісту електронної пошти	Так, шифрує весь вміст електронної пошти та вкладення	Так, шифрує весь вміст електронної пошти та вкладення	Так, шифрує весь вміст електронної пошти
Захист метаданих електронної пошти	Ні	Ні	Ні
Прийняття користувачами	Середній	Помірний	Від низького до помірного
Складність налаштування	Висока	Висока	Низька
Відкликання	Управління за допомогою списків відкликання сертифікатів	Керований користувачем	Не застосовується
Вартість	Може бути дорогим	Зазвичай безкоштовний	Безкоштовно
Резервне копіювання та відновлення	Залежить від сертифікації і дій користувачів	Резервні копії ключів, керовані Користувачем, необхідні	Залежить від управління паролями користувачами
Додаткові функції	Підтримка невідгуку, установка тимчасових міток	Гнучкість алгоритмів шифрування, кросплатформна підтримка	Спрощений інтерфейс користувача для шифрування/дешифрування

S/MIME має високий рівень захисту та добре інтегрується з основними поштовими клієнтами, але вимагає управління цифровими сертифікатами, що може бути складним і дорогим процесом. OpenPGP забезпечує надійний захист і гнучкість в управлінні ключами і алгоритмах шифрування. Він популярний серед користувачів, які піклуються про конфіденційність, але вимагає більш складного навчання завдяки ручному керуванню ключами. Secure Gmail надає користувачам Gmail зручне рішення для шифрування на основі пароля за допомогою розширення Chrome. Незважаючи на простоту

використання, його безпека залежить від надійності обраного пароля, і в ньому відсутні такі функції, як цифрові підписи і повсюдна сумісність.

Оскільки як помітно усі рішення мають свої недоліки актуальною задачею залишається задача розробки нового підходу до забезпечення конфіденційності електронного листування.

1.4 Постановка завдання

У міру просування вперед інноваційні підходи, такі як архітектура з нульовим рівнем довіри, технологія блокчейн, постквантова криптографія та інтеграція Інтернету речей – безпеки, відіграватимуть вирішальну роль у захисті електронної пошти. Випереджаючи виникаючі загрози і постійно вдосконалюючи заходи безпеки, організації можуть знижувати ризики, захищати свої активи і підтримувати довіру до себе у все більш цифровому світі.

Отже актуальною задачею залишається програмна реалізація системи захисту електронних листів.

1.4.1 Функціональність системи

Розглянемо список функціональних вимог:

- аутентифікація та авторизація користувачів. система повинна автентифікувати користувачів за допомогою багатфакторної автентифікації для забезпечення безпечного доступу. Система повинна дозволяти користувачам отримувати доступ до електронної пошти на основі заздалегідь визначених ролей та дозволів. Система повинна підтримувати безпечну сесію для кожного автентифікованого користувача;

- шифрування та дешифрування електронної пошти. Перед відправкою система повинна шифрувати вміст електронної пошти та вкладення за допомогою Intel SGX enclaves. система повинна розшифровувати вміст

отриманих електронних листів і вкладення в межах анклавів Intel SGX. Система повинна використовувати надійні алгоритми шифрування (наприклад, AES-256) для захисту вмісту електронної пошти;

- управління ключами. Система повинна безпечно генерувати ключі шифрування, зберігати їх та керувати ними в анклавах Intel SGX. Система повинна підтримувати механізми обміну ключами для безпечної взаємодії між користувачами. Система повинна забезпечувати можливість відкликання ключів в разі злому ключів;

- цілісність та справжність електронної пошти. Система повинна підписувати вміст електронної пошти та вкладення для забезпечення цілісності та автентичності. Система повинна перевіряти підпис отриманих електронних листів в анклавах Intel SGX. Система повинна генерувати та підтверджувати цифрові сертифікати для ідентифікації користувачів;

- захист метаданих електронної пошти. Система повинна шифрувати конфіденційні метадані електронної пошти (наприклад, тему, відправника, одержувача) для захисту конфіденційності користувачів. Система повинна забезпечувати доступ до метаданих тільки авторизованим користувачам і в межах анклавів Intel SGX;

- безпечне зберігання електронної пошти: система повинна зберігати зашифровані електронні листи та вкладення в захищеному сховищі. Система повинна гарантувати, що ключі дешифрування доступні лише в анклавах Intel SGX. Система повинна забезпечувати безпечні механізми резервного копіювання та відновлення зашифрованих електронних листів;

- фільтрація та сканування електронної пошти: система повинна сканувати електронні листи на наявність шкідливого програмного забезпечення та фішингового вмісту в анклавах Intel SGX. Система повинна фільтрувати спам і шкідливі електронні листи до того, як вони потраплять в поштову скриньку користувача. Система повинна регулярно оновлювати алгоритми виявлення шкідливого ПЗ і фішингу;

– інтерфейс користувача: система повинна забезпечити зручний інтерфейс для складання, надсилання та читання зашифрованих електронних листів. Система повинна дозволяти користувачам керувати ключами шифрування та сертифікатами через інтерфейс. Система повинна надавати повідомлення та оповіщення про події, пов'язані з безпекою (наприклад, невдалі спроби входу в систему, відкликання ключа);

– аудит та ведення журналу: система повинна вести журнал аудиту всіх подій, пов'язаних з безпекою, включаючи доступ до електронної пошти, генерацію ключів і спроби аутентифікації. Система повинна забезпечувати захист журналів аудиту від несанкціонованого доступу та їх надійне зберігання. Система повинна надавати інструменти для аудиту та моніторингу подій, пов'язаних з безпекою електронної пошти.

– відповідність вимогам і нормативні вимоги: система повинна відповідати відповідним положенням про захист даних (наприклад, GDPR, HIPAA). Система повинна забезпечувати настроювані політики для відповідності конкретним вимогам організації. Система повинна підтримувати політику зберігання та видалення даних відповідно до нормативних вимог;

– продуктивність та масштабованість: система повинна забезпечувати ефективні операції шифрування та дешифрування для мінімізації витрат на продуктивність. Система повинна бути масштабованою для роботи з великою кількістю користувачів та електронною поштою без зниження продуктивності. Система повинна підтримувати балансування навантаження і відмовостійкість для забезпечення високої доступності;

– сумісність: система повинна підтримувати взаємодію з існуючими Поштовими службами (наприклад, Gmail, Outlook) за допомогою стандартних протоколів (наприклад, IMAP, SMTP). Система повинна забезпечувати безпечне спілкування з користувачами, які не використовують систему захисту електронної пошти, шляхом надання тимчасових методів безпечного доступу. Система повинна бути інтегрована з іншими системами безпеки

(наприклад, DLP, SIEM) для забезпечення комплексної безпеки електронної пошти.

1.4.2 Вимоги до інтерфейсу

Інтерфейс застосунку має відрізнятися лаконічністю та без зайвих проблем влаштовуватися в стандартні поштові клієнти, наприклад, в програму Microsoft Outlook.

1.4.3 Вимоги до безпеки

Головна мета розробки забезпечити конфіденційність електронного листування. Отже головна вимога до безпеки даних – використання технологій, що будуть гарантувати цілісність даних.

1.4.4 Нефункціональні вимоги до продукту

Нефункціональні вимоги зводяться до вимог експлуатаційних:

- IBM-сумісний персональний комп'ютер;
- процесор із тактовою частотою від 500 МГц;
- оперативна пам'ять об'ємом від 512 Мб;
- вільне місце на накопичувачі об'ємом від 2 Гб;
- встановлений застосунок Microsoft Outlook.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Технологія Intel SGX

У наш час комп'ютерні системи обробляють величезну кількість важливої, цінної та конфіденційної інформації. Ця інформація повинна бути захищена від підробки або крадіжки. Існує ціла індустрія, яка займається крадіжкою інформації, такої як банківські дані або корпоративна Інтелектуальна власність і існує безліч додатків, які повинні зберігати таку та іншу секретну інформацію. У даній роботі розглядається технологія Intel SGX – Intel Software Guard Extensions, що дозволяє забезпечити конфіденційність секретної інформації [6]–[8].

Intel SGX це набір нових інструкцій та змін у доступі до пам'яті, внесених до архітектури Intel. Ці розширення дозволяють програмам створювати захищений контейнер, який називається анклав. Анклав – це захищена область в адресному просторі програми, яка забезпечує конфіденційність і цілісність навіть при наявності привілейованих шкідливих програм. Спроба отримати доступ до такої області пам'яті з ПЗ, яке не належить до цього анклаву, буде зберігатися, навіть якщо спроба здійснюється з привілейованих програм, таких як менеджери віртуальних машин, BIOS або ядро операційної системи [6]–[8].

SGX дозволяє захищеній частині програми поширюватися у відкритому вигляді. До створення анклаву його код та дані відкриті для перевірки та аналізу. Захищена частина завантажується в анклав після того як його код і дані перевірені. Як тільки код і дані програми завантажені в анклав, вони захищені від доступу сторонніх програм. Застосунок може довести свою справжність третій стороні і безпечно отримати ключі та облікові дані. Застосунок так само може запросити ключ унікальний для кожної пари анклаву і платформи, щоб використовувати його для захисту ключів і даних, які воно хоче зберігати поза анклавом [6]–[8].

Крім забезпечення безпеки анклав пропонує масштабованість і продуктивність, що забезпечується виконанням на головному CPU відкритої платформи.

Підтримка SGX включає два головних розширення в архітектурі Intel. Перша – це зміна семантики доступу до пам'яті анклаву. Друге – захищене перетворення адрес [6]–[8].

2.2 Загальний опис захисних механізмів технології Intel SGX

SGX запобігає доступу інших програм до коду та даних, що знаходяться в анклаві, включаючи системне програмне забезпечення та доступ з інших анклавів. Спроби модифікувати вміст анклаву виявляються і або запобігають, або виконання переривається. Технологія SGX має наступні властивості:

- SGX забезпечує виявлення порушення цілісності анклаву внаслідок атак інших програм і запобігає доступ до змінених коду і даних після виявлення;
- SGX забезпечує конфіденційність коду та даних анклаву;
- SGX забезпечує ізоляцію між кількома анклавами;
- SGX запобігає повторному відтворенню анклаву внаслідок атак інших програм.

До того ж, апаратні засоби забезпечують виконання тільки з місць, визначених анклавом, і запобігають витоку інформації з анклаву при незапланованих перериваннях виконання.

Нарешті, дані, що зберігаються всередині анклаву, повинні бути захищені від підробки всіма програмами, що знаходяться поза зоною довіреності анклаву, навіть, коли анклав записаний на диск або незахищену частину пам'яті операційною системою при управлінні ресурсами [6], [8].

Щоб досягти цих властивостей, деякі нові можливості повинні бути впроваджені в апаратні засоби. По-перше, код, що виконується всередині

анклаву повинен мати доступ до коду і даних, що знаходяться в анклаві, при цьому доступ ззовні анклаву повинен бути заборонений. По-друге, перетворення з віртуальної адреси програми в коректну фізичну адресу має бути таким же, як коли розробник створював звичайний застосунок [6], [8].

У той час як дані анклаву розташовані серед регістрів, кешу та інших логічних блоків всередині процесора, несанкціонований доступ за допомогою програм запобігається, використовуючи механізм контролю доступу, вбудований в процесор. Однак, коли дані анклаву залишають кеш, щоб бути записаними в пам'ять платформи, дані автоматично зашифровуються і захищені з точки зору цілісності для запобігання дослідження пам'яті або інших технік перегляду, зміни або заміни даних або коду, що зберігаються всередині анклаву [6], [8].

2.3 Модель програмування технології Intel SGX

Архітектура SGX включає 17 нових інструкцій, нові структури процесора та новий режим виконання. Вони включають в себе завантаження анклаву в захищену область пам'яті, доступ до ресурсів за допомогою відображення адрес через таблиці сторінок і планування виконання додатків, що використовують анклав. Таким чином, системне ПО як і раніше підтримує контроль того, до яких ресурсів у анклаву буде доступ. Додаток може бути інкапсульовано в один анклав або воно може бути розбите на кілька компонентів, так, щоб тільки секретні частини були поміщені в анклав.

Далі розглянуто набір інструкцій SGX. Ці інструкції можуть бути згруповані за такими функціями: створення / видалення анклаву, вхід/вихід в анклав, інструкції безпеки в анклаві, інструкції управління сторінками та інструкції для налагодження [6], [8].

Інструкції створення та видалення анклаву використовуються для виділення захищеної пам'яті для анклаву, завантаження значень у захищену

пам'ять, вимірювання значень, завантажених у захищену пам'ять анклаву та для видалення анклаву після завершення роботи програми [6], [8].

Інструкції входу в анклав і виходу з нього дозволяють увійти в анклав (EENTER) і вийти з нього (EEXIT). Також можна вийти з нього асинхронно (AEX) по переривань або по обробці винятків. У разі AEX, апаратні засоби зберігають всю конфіденційну інформацію всередині анклаву, видаливши її з регістрів, і повертають управління програмі після обробки переривання або виключення. Анклав же продовжить виконання з того місця, на якому вона зупинила своє виконання [6], [8].

Інструкції з управління сторінками дозволяють системному програмному забезпеченню безпечно переміщувати сторінки анклаву в і з незахищеної пам'яті [6], [8].

Інструкції налагодження анклаву дозволяють розробникам використовувати знайомі техніки налагодження всередині анклавів, створених у режимі налагодження. Налагоджувальний анклав можна виконувати поетапно та аналізувати. Налагоджувальний анклав не може обмінюватися даними з виробничим анклавом. Це захищає розробників анклаву у випадку, коли анклав налагодження залишає середовище розробки [6], [8].

Інструкції щодо безпеки анклаву дозволяють анклаву підтвердити третій стороні, що анклав був створений на апаратних засобах, що підтримують набір інструкцій SGX [6], [8]. EPC забезпечує в системі область захищеної пам'яті для анклаву. EPCМ-це службова інформація, пов'язана з безпекою, прикріплена до кожної сторінки EPC. EPCМ містить інформацію, необхідну апаратним засобам для захисту доступу до пам'яті анклаву [6], [8]. Між сторінками EPC і EPCМ записами відповідність один до одного. SECS займає одну сторінку EPC і містить корисну інформацію, необхідну для конкретного анклаву. TCS займає 1 сторінку EPC і містить службову інформацію, яка використовується апаратними засобами для контролю входів в анклав кожного логічного процесора [6], [8].

Набір всіх інструкцій технології Intel SGX представлений в табл. 2.1.

Таблиця 2.1 – Інструкції технології Intel SGX

Інструкція	Опис
ECREATE	Визначає діапазон адрес, запускає побудову
EADD	Додає сторінку розміром 4 Кбайт
EEXTEND	Вимірює 256 байт
EINIT	Оголошує анклав побудованим
EREMOVE	Видаляє анклав
EENTER	Входить в анклав
ERESUME	Відновлює роботу анклаву
EEXIT	Завершує роботу анклаву
AEX	Асинхронно завершує роботу анклаву
EPA	Створює сторінку версійного масиву
ELDB/U	Завантажує вивантажену сторінку в захищений простір
EWB	Вивантажує захищену сторінку
EBLOCK	Готує до вивантаження
ETRACK	Готує до вивантаження
EDBGRD	Читання всередині анклаву налагодження
EDBGWR	Запис всередині анклаву налагодження
EREPORT	Створення структури Report
EGETKEY	Генерація унікального ключа

Далі розглянуто нові структури даних та компоненти. Вони потрібні для підтримки інформації, необхідної для дотримання властивостей безпеки SGX. Основні структури показані в табл. 2.2.

Таблиця 2.2 – Структури технології Intel SGX

Інструкція	Опис
ECREATE	Містить захищений код і дані в сторінках розміром 4 КБайт
EADD	Містить службову інформацію про сторінки анклав
EEXTEND	Службова Інформація про кожен анклав
EINIT	Службова Інформація про кожен потік
EREMOVE	Масив версій вивантажених сторінок
EENTER	Структура, що підписує анклав; запечатують ідентифікаційні дані

Технологія SGX так само містить новий режим процесора-режим анклав. Коли процесор входить в анклав, він починає працювати в режимі анклав. Цей режим змінює алгоритм доступу до пам'яті, щоб провести додаткові перевірки при доступі до пам'яті. Це дозволяє коду всередині анклав отримувати доступ до цього конкретного анклав. В інших випадках, доступ до пам'яті EPC призводить до помилки.

Також, SGX містить механізм управління ресурсами анклав. EPC-це розподілена система, ресурси якої управляються привілейованим програмним забезпеченням. EPC містить сторінки, які повинні виконуватися в анклаві. Для виконання анклав не обов'язково, щоб всі сторінки були присутні в EPC. Управління ресурсами анклав схоже на архітектуру віртуальної пам'яті, де старниці, які в даний момент не виконуються, можуть бути видалені з пам'яті [6], [8].

2.4 Компоненти апаратних засобів технології Intel SGX

Для реалізації захисту пам'яті SGX необхідні нові апаратні засоби та структури [6], [8].

Сторінковий кеш анклаву (The Enclave Page Cache – EPC) – це захищена пам'ять, де зберігаються сторінки анклаву і структури SGX. Ця пам'ять захищена від доступу апаратними і програмними засобами. Усередині EPC розташовуються код і дані багатьох анклавів. Коли анклав отримує доступ до пам'яті EPC, процесор визначає, дозволяти чи ні доступ. Процесор підтримує інформацію про безпеку і контроль доступу для кожної сторінки в EPC в структурі, званої таблицею сторінкового кеша анклаву (Enclave Page cache Map – EPCM). До цієї структури звертається апаратний модуль обробника відсутності сторінок (Page Miss Handler – PMH). PMH служить посередником для доступу до пам'яті та враховує таблиці сторінок, що підтримуються системним програмним забезпеченням, реєстри діапазону та EPCM [6], [8].

EPC розділений на ділянки розміром 4кбайт, звані сторінками EPC. EPC сторінки можуть бути дійсними або не дійсними. Дійсна EPC сторінка містить або сторінку анклаву, або структуру SGX. Атрибути безпеки для кожної EPC сторінки зберігаються у внутрішній структурі, званої EPCM, яка описана нижче. Кожен анклав має структуру контролю анклаву, SECS. Кожна дійсна сторінка анклаву в EPC належить точно одному екземпляру анклаву.

Системне програмне забезпечення вимагає відповідності віртуальних адрес і дійсних EPC сторінок [6], [8].

EPCM складається з набору записів, по одному для кожної сторінки EPC. EPCM управляється процесором в ході виконання різних інструкцій SGX і ніколи безпосередньо недоступна програмному забезпеченню або іншим апаратним пристроям [6], [8]. Формат EPCM мікроархітектурний і залежить від реалізації. Однак, логічно, кожен запис EPCM містить наступну інформацію:

- дійсна або не дійсна EPC сторінка;
- екземпляр анклаву, що володіє цією сторінкою;
- тип сторінки (REG, TCS, VA, SECS);

- віртуальна адреса, за допомогою якої анклав може отримати доступ до цієї сторінки;
- права на читання/запис/виконання цієї сторінки;
- чи є сторінка доступною чи ні (BLOCKED або UNBLOCKED).

Структура EPCM використовується процесором при перетворенні адрес для забезпечення контролю доступу до сторінок анклаву, що завантажуються в ЕРС. Логічно EPCM забезпечує додатковий шар безпеки для контролю доступу на додаток до існуючих механізмів сегментації, таблиць сторінок та розширених таблиць сторінок [6], [8].

Розташування ЕРС залежить від конкретної реалізації і може бути отримано за допомогою інструкції CPUID [6], [8].

Для підтримки технології потрібен процесор, який підтримує SGX і реалізує ЕРС і BIOS, який підтримує можливість резервувати діапазон пам'яті, званий зарезервована процесором пам'ять (Processor Reserved Memory – PRM). BIOS виділяє PRM, конфігуруючи набір регістрів, що називаються PRMRR.

Використання оперативної пам'яті як місця розташування ЕРС необхідно для багатьох програм. Проблема в тому, що існує багато відомих програмних та апаратних атак, які спрямовані на оперативну пам'ять. Криптографічний захист вмісту ЕРС в оперативній пам'яті є одним способом захисту від цих атак.

Механізм, застосовуваний процесором для захисту пам'яті ЕРС називається механізм шифрування пам'яті (Memory Encryption Engine – MEE). MEE це апаратний модуль, який шифрує і контролює цілісність обраного трафіку між корпусом процесора і оперативною пам'яттю. Область пам'яті, на яку впливає MEE, називають областю мее. PRM може складатися з однієї або більше областей MEE [6], [8].

Сенс механізму контролю доступу до пам'яті полягає в наступному. Механізми захисту пам'яті процесора фізично блокують доступ до PRM від усіх зовнішніх пристроїв, обробляючи такі спроби доступу як посилення на неіснуючу пам'ять. Для того, щоб отримати доступ до сторінки в анклаві,

використовуючи MOV або іншу пов'язану з пам'яттю інструкцію, необхідно пройти наступні апаратні перевірки, представлені на рис. 2.1:

- логічний процесор виконується в режимі анклаву;
- сторінка належить анклаву, який виконується на логічному процесорі;
- запитувана сторінка має коректний віртуальний адресу.

Якщо сторінка, до якої запитується доступ, не є частиною віртуального адресного простору анклаву, але є частиною ЕРС тоді, спроба доступу обробляється як посилання на неіснуючу пам'ять. Якщо сторінка знаходиться за межами віртуального адресного простору анклаву, і вона запитує дозвіл на доступ до сторінки PRM, апаратні засоби запобігають такий доступ, видаючи помилку. Спроби доступу процесора, що не перебуває в режимі анклаву, до сторінки анклаву обробляються як посилання на неіснуючу пам'ять.

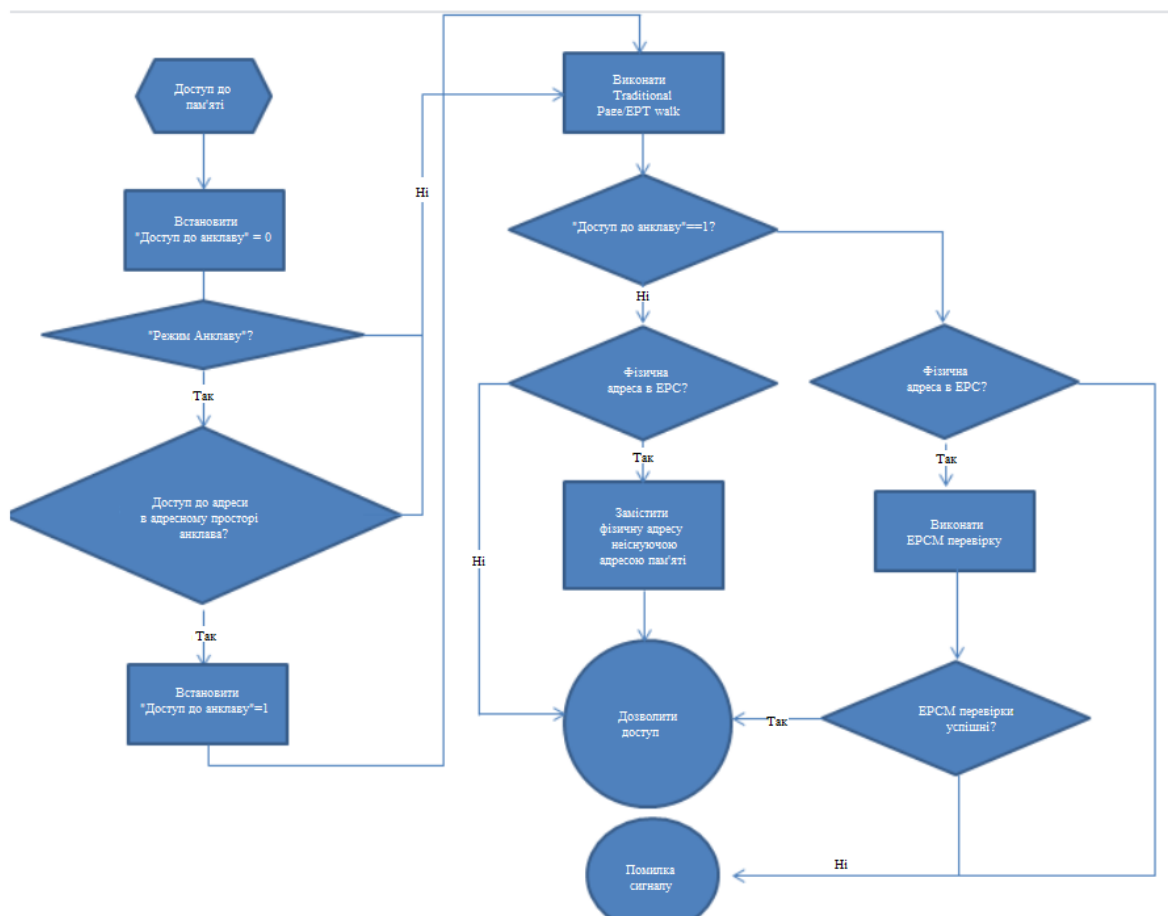


Рисунок 2.1 – Схема доступу до пам'яті анклаву

2.5 Вибір мови програмування

Вибір C++ для розробки застосунків з розширеннями Intel SGX є зваженим рішенням, враховуючи унікальні характеристики мови і сумісність з вимогами SGX [9], [10].

Intel SGX – це набір кодів команд, пов'язаних із безпекою, вбудованих у сучасні процесори Intel. Він дозволяє розробникам захищати вибраний код і дані від розкриття або модифікації за допомогою анклавів – захищених областей виконання в пам'яті. SGX призначений для того, щоб дозволити розробникам створювати безпечні програми, де конфіденційні дані та код можуть виконуватися в ізольованому середовищі, захищаючи їх від зовнішніх загроз, у тому числі від підвищених загроз, таких як операційна система або гіпервізор [9], [10].

C++, потужна і широко використовувана мова програмування, особливо добре підходить для додатків системного рівня та критично важливих для продуктивності програм. Він забезпечує баланс між конструкціями програмного забезпечення високого рівня та можливостями маніпулювання пам'яттю низького рівня, що робить його ідеальним для розробки додатків, які потребують детального контролю над апаратними ресурсами [9], [10].

Прямий доступ до пам'яті – C++ забезпечує механізми прямого управління пам'яттю, що має вирішальне значення для розробки ефективних анклавних додатків. Анклави Intel SGX вимагають дбайливого ставлення до пам'яті для підтримки безпеки і продуктивності.

Вбудований асемблер – C++ дозволяє використовувати вбудований асемблерний код, що може мати вирішальне значення для оптимізації продуктивності в додатках, чутливих до безпеки [9], [10].

Оптимізована компіляція – компілятори C++ мають високий ступінь оптимізації та здатні створювати ефективний машинний код. Це важливо для додатків SGX, де витрати на продуктивність повинні бути мінімізовані, щоб

гарантувати, що функції безпеки не матимуть суттєвого впливу на продуктивність програми [9], [10].

Управління ресурсами – такі функції, як RAII (отримання ресурсів – це ініціалізація), допомагають ефективно управляти ресурсами, що важливо в обмеженому середовищі SGX enclave.

Арифметика покажчиків – C++ надає покажчики та можливості ручного управління пам'яттю, дозволяючи розробникам оптимізувати використання пам'яті в анклавів.

Спеціальні розподільники – розробники можуть реалізувати спеціальні розподільники пам'яті для управління пам'яттю анклавів, що може бути особливо корисним для управління обмеженими ресурсами пам'яті в анклавів SGX [9], [10].

Паралелізм та паралелізація – C++ підтримує різні конструкції паралелізму та паралелізації (наприклад, `std::thread`, OpenMP, Intel TBB), які можна використовувати для максимального підвищення продуктивності захищених додатків, забезпечуючи при цьому їх чуйність та ефективність [9], [10].

Низькорівневі системні виклики – C++ може легко взаємодіяти з системними API низького рівня, що робить його придатним для розробки системного програмного забезпечення та захищених програм, яким потрібно тісно взаємодіяти з операційною системою.

Intel SGX SDK – Intel SGX SDK в першу чергу розроблений з урахуванням C/C++ і забезпечує всебічну підтримку, включаючи бібліотеки, Інструменти та документацію, для полегшення розробки додатків SGX.

Існуюча кодова база – багато існуючих програм безпеки та системного рівня написані на C/C++, що полегшує інтеграцію функцій SGX у ці програми без істотного переписування [9], [10].

Інтеграція з налагоджувачем – такі інструменти, як Intel SGX SDK Debugger, який інтегрується з IDE, такими як Visual Studio, призначені для

розробки на C++, забезпечуючи кращу підтримку для налагодження коду enclave.

Інструменти профілювання та оптимізації – Intel надає ряд інструментів профілювання та оптимізації (наприклад, Intel Vtune Amplifier), які легко взаємодіють із програмами C++ для оптимізації продуктивності в анклавах.

Безпека типу – хоча C++ не настільки безпечний для типу, як деякі сучасні мови, він забезпечує суворе введення тексту та можливість більш суворої перевірки типу, що може допомогти усунути певні типи вразливостей безпеки [9], [10].

Інструменти статичного аналізу – екосистема C++ включає потужні інструменти статичного аналізу (наприклад, Coverity, Clang Static Analyzer), які можуть допомогти виявити потенційні вразливості в кодовій базі.

Кращі практики – зрілість C++ означає, що існують добре зарекомендували себе кращі практики безпечного кодування, які можуть бути застосовані при розробці SGX enclave для мінімізації ризиків безпеки [9], [10].

Прийняття спільнотою та галуззю – C++ має широке співтовариство і широко застосовується в галузі для розробки безпечних додатків, забезпечуючи достатню підтримку та багатий багаж знань.

Кросплатформність – C++ за своєю суттю не залежить від платформи, що означає, що програми SGX, написані на C++, легше переносити на різні платформи та середовища.

Стандартні бібліотеки – стандартна бібліотека C++ надає багатий набір функцій і утиліт, які узгоджені на різних платформах, спрощуючи процес розробки [9], [10]. Інтеграція застарілого коду – C++ може взаємодіяти з C та іншими мовами, що полегшує інтеграцію функцій SGX в існуючі кодові бази.

Сторонні бібліотеки – для C++ доступна велика екосистема сторонніх бібліотек, які можна використовувати для додавання функціональності до програм SGX, не винаходячи велосипед.

Активна спільнота – у C++ існує велика активна спільнота розробників, які беруть участь у форумах, проектах з відкритим кодом та інших

платформах для співпраці. Це цінний ресурс для усунення несправностей та отримання рекомендацій [9], [10].

Проекти з відкритим кодом – багато проектів та бібліотек безпеки з відкритим кодом написані на C++ і пропонують готові до використання компоненти для розробки SGX [9], [10].

Широке впровадження – C++ широко використовується в галузях, що вимагають високої продуктивності та безпеки, таких як фінанси, Телекомунікації та аерокосмічна промисловість. Таке широке впровадження гарантує, що найкращі практики та інструменти для безпечної розробки додатків добре зарекомендували себе.

Освітні ресурси – існує безліч освітніх ресурсів, включаючи книги, навчальні посібники та курси, які охоплюють як C++, так і методи безпечного програмування, що полегшує розробникам освоєння SGX-розробки.

Використовуючи C++ для розробки SGX, розробники можуть створювати застосунки, які є не тільки безпечними, але й ефективними та масштабованими, що відповідають суворим вимогам сучасних середовищ, чутливих до безпеки.

Виконаємо обґрунтування вибору у вигляді таблиці 2.3.

Таблиця 2.3 – Порівняння мов програмування

Критерій	C++	Java	Python
Підтримка технології Intel SGX	Напряму	Через бібліотеку	Через рушій
Захист даних	Високий рівень	Високий рівень	Середній рівень
Продуктивність	Дуже висока	Висока	Середня
Можливості низькорівневого програмування	Присутні	Присутні	Відсутні
Сумісність та кросплатформність	Кросплатформна мова, сумісна	Кросплатформна (вимагає додаткових налаштувань)	Кросплатформна (вимагає додаткових налаштувань)

Враховуючі специфічність технології Intel SGX та вимоги щодо оптимізації, продуктивності та низькорівневості – найкращий варіант C++.

Вибір C++ для розробки застосунків на базі Intel SGX – це стратегічне рішення, зумовлене кількома факторами. Висока продуктивність мови, детальний контроль над апаратними ресурсами, сумісність з SGX SDK і потужна підтримка спільноти роблять його ідеальним вибором для розробки безпечних і високопродуктивних застосунків. C++ надає необхідні інструменти та функції для повного використання можливостей Intel SGX, гарантуючи захист конфіденційних даних та коду в ізольованому середовищі. Крім того, багата екосистема та впровадження C++ у промисловість надають розробникам доступ до великих ресурсів та підтримки, що полегшує розробку надійних та безпечних додатків SGX.

2.6 Вибір середовища розробки

Visual Studio Code (VS Code) стала однією з найпопулярніших інтегрованих середовищ розробки (IDE) для широкого спектру мов програмування, включаючи C++. Її популярність обумовлена поєднанням функціональних можливостей, гнучкості і продуктивності [10], [11].

VS Code розроблений таким чином, щоб бути легким, що означає, що він швидко запускається і споживає менше ресурсів у порівнянні з деякими більш важкими IDE. Незважаючи на свою легку вагу, VS Code може ефективно працювати з великими проектами без значних уповільнень.

VS Code має багату екосистему розширень, доступних на своєму ринку. Розширення для розробки на C++, такі як розширення C/C++, надане корпорацією Майкрософт, значно спрощують процес розробки [10], [11].

Користувачі можуть налаштувати своє середовище розробки за допомогою теми, комбінацій клавіш та налаштувань відповідно до своїх уподобань.

Розширення C/C++ для VS Code надає потужні функції IntelliSense, включаючи автозавершення, інформацію про параметри, коротку інформацію та списки учасників, які допомагають пришвидшити кодування та зменшити помилки [10], [11].

Вбудований налагоджувач: VS Code пропонує надійні можливості налагодження. Розширення C / C++ інтегрується з GDB та LLDB, надаючи такі функції, як точки зупинки, змінні відстеження, стеки викликів та перевірка змінних.

Інтуїтивно зрозумілий інтерфейс налагодження дозволяє розробникам легко виконувати покрокову роботу з кодом, перевіряти змінні та обчислювати вирази [10], [11].

VS Code включає вбудований термінал, що підтримує кілька оболонок. Це дозволяє розробникам запускати команди збірки, сценарії та інші інструменти командного рядка, не виходячи з IDE.

Інтеграція терміналу в IDE спрощує робочий процес, полегшуючи компіляцію та запуск програм безпосередньо з VS Code.

Код VS Code доступний у Windows, macOS та Linux. Така Кросплатформна підтримка гарантує, що розробники зможуть використовувати одне і те ж середовище розробки в різних операційних системах.

Узгодженість роботи на різних платформах допомагає підтримувати продуктивність при перемиканні між різними операційними системами.

VS Code має вбудовану підтримку Git, що полегшує керування версіями в IDE. Користувачі можуть виконувати спільні операції з Git, такі як коміт, надсилання, Витяг та злиття безпосередньо з інтерфейсу [10], [11].

Розширення для інших віртуальних машин – доступні розширення для інших систем контролю версій, таких як Subversion, Mercurial та Perforce.

Система завдань VS Code дозволяє користувачам налаштовувати завдання для створення та запуску проектів на C++. Завдання можна налаштувати для використання систем збірки, таких як Make, CMake та Ninja.

Розширення для інструментів збірки – доступні розширення для популярних інструментів збірки, таких як CMake, які надають такі функції, як налаштування, створення та налагодження проектів CMake безпосередньо в IDE [10], [11]. У VS Code велика та активна спільнота користувачів та членів. Це означає, що допомога легко доступна на форумах, в розділах з питань GitHub і на інших ресурсах спільноти. Доступна повна документація як для коду VS, так і для його розширень, що полегшує розробникам роботу та усунення несправностей. Розширення для віддаленої розробки – VS Code пропонує можливості віддаленої розробки за допомогою таких розширень, як Remote - SSH, Remote - Containers і Remote - WSL. Це дозволяє розробникам безперешкодно працювати з кодом, розміщеним на віддалених комп'ютерах або в контейнерах [10], [11].

Розширення, такі як Visual Studio Codespaces, дозволяють створювати хмарні середовища розробки, які можуть бути особливо корисними для спільних проектів та середовищ із певними залежностями.

Інтерфейс сучасний і зрозумілий, з підтримкою тим оформлення і кастомізації. Це робить його візуально привабливим і зручним для використання під час тривалих сесій програмування.

Розширення для налаштування інтерфейсу – розробники можуть додатково налаштовувати інтерфейс за допомогою розширень, які додають додаткові функціональні можливості або змінюють зовнішній вигляд.

Visual Studio Code виділяється як одна з кращих IDE для розробки на C++ завдяки своїй легкості і високій продуктивності, широкої розширюваності завдяки багатому набору розширень, надійним можливостям IntelliSense і налагодження, кросплатформної підтримки і тісній інтеграції з системами контролю версій і інструментами збірки. Вбудований термінал, можливості віддаленої розробки та активна підтримка спільноти ще більше підвищують його привабливість. Пропонуючи сучасний, настроюваний інтерфейс та потужні функції, VS Code забезпечує гнучке та ефективне середовище розробки для розробників на C++, що робить його кращим вибором у галузі.

У табл. 2.4 наведемо обґрунтування вибору середовища розробки.

Таблиця 2.4 – Порівняння середовищ розробки

Критерій	VS Code	Eclipse	CLion
Ліцензія, розповсюдженість	Безкоштовно, вільна ліцензія	Безкоштовно, вільна ліцензія	Безкоштовно/ платно
Налаштування та гнучкість	Високий рівень	Високий рівень	Середній рівень
Рефакторинг	Високий рівень	Середній рівень	Високий рівень
Продуктивність	Високий рівень	Високий рівень	Середній рівень
Підключення сторонніх розробок	Легко	Легко	Тільки з каталогу

VS Code – це оптимізоване середовище розробки, яке, якнайкраще підходить для випадків використання специфічних технологій та бібліотек. Саме через ці сильні сторони, було обрано VS Code.

2.7 Структурна схема розробки

Загальна схема системи за обраною темою на рис. 2.2.

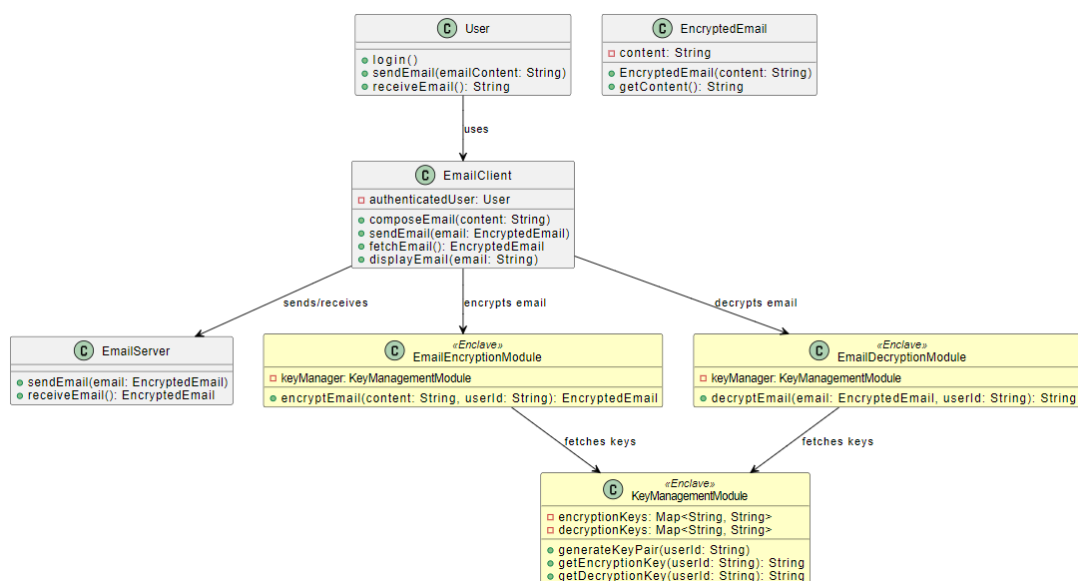


Рисунок 2.2 – Загальна схема системи за обраною темою

Взаємовідносини і взаємодії:

- користувач використовує поштовий клієнт для складання, надсилання та отримання електронних листів;
- поштовий клієнт взаємодіє з поштовим сервером для надсилання та отримання електронних листів;
- клієнт електронної пошти використовує модуль шифрування електронної пошти для шифрування вихідних електронних листів;
- клієнт електронної пошти використовує модуль розшифровки Електронної Пошти для розшифровки вхідних електронних листів;
- модуль шифрування електронної пошти та модуль дешифрування електронної пошти взаємодіють з модулем управління ключами для отримання необхідних ключів для шифрування та дешифрування.

На цій діаграмі класів представлений високорівневий огляд компонентів і їх взаємодії в системі захисту електронної пошти на базі Intel SGX, що забезпечує безпечну електронну переписку за рахунок використання анклавів для конфіденційних операцій.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Операції над анклавом в технології Intel SGX

Процес створення анклаву завантажує двійковий файл анклаву в EPC та встановлює ідентифікаційні дані анклаву. Процес створення анклаву розділений на кілька етапів: ініціалізація керуючої структури анклаву, виділення сторінок EPC і завантаження змісту анклаву в сторінки, вимірювання змісту анклаву і нарешті встановлення ідентифікаційних даних анклаву. Ці кроки підтримуються наступними інструкціями: ECREATE, EADD, EEXTEND та EINIT.

ECREATE запускає процес створення анклаву і ініціалізує SGX керуючу Структуру анклаву (SGX Enclave Control Structure – SECS), яка містить глобальну інформацію про анклав. EADD копіює сторінки в EPC і записує факт копіювання, але не вміст сторінки в SECS. Зміст пам'яті анклаву явно вимірюється інструкцією EEXTEND. EINIT завершує процес створення, завершує вимірювання анклаву та встановлює ідентифікаційні дані анклаву. До тих пір, поки інструкція EINIT не виконана, використання анклаву не дозволяється. Важливим кроком у виконанні інструкції EINIT є порівняння результату вимірювання анклаву при його створенні з еталонним значенням вимірювання анклаву, створеним на етапі розробки і зберігаються в структурі SIGSTRUCT.

Критично важливим для збереження цілісності анклаву є управління передачею виконання програми в анклав і з нього. Процес входу в анклав повинен очистити будь-які кешовані перетворення, які перетинаються із захищеним адресним простором анклаву. Це гарантує, що всі спроби доступу до захищеної пам'яті анклаву правильно перевірені. Процес входу повинен ідентифікувати, куди в анклаві процесор повинен передати управління і включити режим анклаву. Вихід з анклаву повинен знову очистити будь-які кешовані перетворення, що відносяться до захищеного адресного простору

анклаву для того, щоб ніяке інше програмне забезпечення не могло використовувати кешовані перетворення, щоб отримати доступ до захищеної пам'яті анклаву.

При роботі в режимі анклаву можуть статися переривання, помилка або виключення. Зазвичай процесор звертається до обробника помилок, визначеного системним програмним забезпеченням. Обробник помилок зберігає стан регістрів і обробляє подію. Як тільки подія була оброблена, системне програмне забезпечення відновлює стан регістрів і повертає управління туди, де виконання програми було перервано. Дозвіл системному програмному забезпеченню читати та / або змінювати стан регістрів анклаву поміщає системне програмне забезпечення в довірчий простір анклаву. Отже, SGX представляє новий механізм захисту цілісності та конфіденційності анклаву.

SGX пропонує інструкції EENTER і EEXIT, для програмного входу і виходу з анклаву (наприклад, в ході послідовності викликів і повернень). При виході з анклаву через деякої події, процесор викликає спеціальну внутрішню підпрограму під назвою асинхронний вихід (Asynchronous Exit – AEX), який зберігає стан регістрів анклаву, очищає регістри, встановлює значення адреси обробної помилки інструкції в значення, визначене EENTER. Інструкція ERESUME відновлює стан назад, щоб дозволити анклаву відновити виконання.

Можливість системного програмного забезпечення перевищити намічену кількість станиць EPC збільшує число захищених додатків, які можуть бути запущені одночасно. Архітектура SGX пропонує інструкції щодо вирішення цієї проблеми шляхом безпечного завантаження та завантаження сторінок анклаву та структур SGX. При цьому, у вмісту сторінки анклаву, вивантаженої з EPC в оперативну пам'ять, повинен бути той же рівень цілісності, конфіденційності та захисту від відтворення як у вмісту сторінки, що знаходиться в EPC. Для досягнення цієї мети, інструкції управління сторінками дотримуються наступних правил:

- сторінка анклаву повинна бути вивантажена тільки після того, як всі кешовані перетворення з цієї сторінки були вивантажені з усіх логічних процесорів;
- зміст вивантаженої сторінки анклаву має бути зашифровано перед переміщенням в оперативну пам'ять;
- коли вивантажена сторінка анклаву заново завантажена в ЕРС, вона має мати ідентичний тип сторінки, повноваження, віртуальну адресу, зміст, і бути пов'язана з тим же анклавом, що і до вивантаження;
- тільки останньої вивантаженої версії сторінки анклаву можна дозволити бути заново завантаженої.

3.2 Робота з секретними даними в технології Intel SGX

ПЗ з підтримкою SGX не постачається з конфіденційними даними. Тільки після встановлення програма взаємодіє з постачальником послуг, щоб отримати віддалений доступ до цих даних. Потім програмне забезпечення шифрує дані та зберігає їх для подальшого використання. Рис. 3.1 демонструє цей процес.

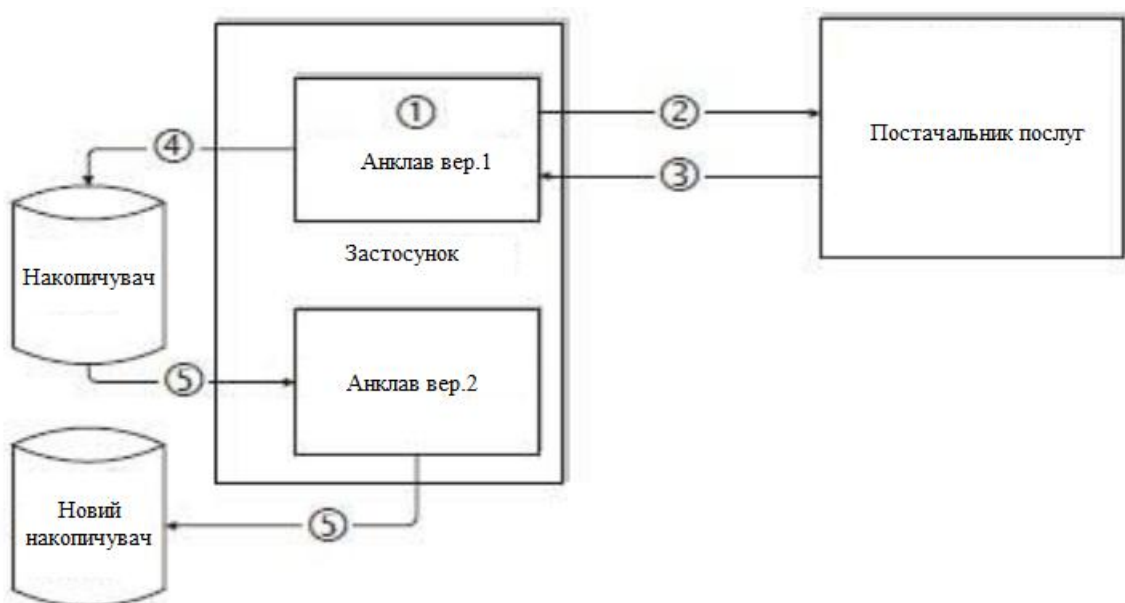


Рисунок 3.1 – Життєвий цикл програми використовує Intel SGX

Запуск анклаву – ненадійний застосунок запускає анклав для взаємодії з програмним забезпеченням постачальника послуг. Під час запуску опис даних та процесу завантаження, що містяться в ньому, записується в захищений журнал. Цей безпечний журнал називається вимірюванням анклаву.

Атестація – анклав повідомляє постачальнику послуг прохання про надання конфіденційних даних анклаву. Платформа генерує докази безпеки, які ідентифікують апаратне середовище та анклав.

Забезпечення – постачальник послуг оцінює надійність анклаву. Він використовує докази безпеки для встановлення безпечного зв'язку та надання конфіденційних даних анклаву. За допомогою захищеного каналу, постачальник послуг відправляє дані в анклав.

Запечатування / відкриття – анклав використовує стійкий апаратний ключ шифрування, щоб надійно зашифрувати і зберегти свої конфіденційні дані. Це забезпечує можливість отримання даних тільки в разі відновлення роботи довіреної середовища.

Оновлення ПЗ – оновлення ПЗ анклаву може знадобитися постачальником послуг. Щоб спростити міграцію даних зі старої версії програмного забезпечення в нову, програмне забезпечення може запросити Ключі старих версій для роздрукування даних, а так само запросити запечатування нової версії. Таким чином, запечатані дані не будуть доступні для попередніх версій програмного забезпечення.

Нарешті, коли власники платформи планують передати право власності на платформу, конфіденційні дані, доступні протягом періоду їх власності повинні бути зроблені недоступними. Для цього SGX реалізує спеціальну константу належить користувачеві, зміна якої змінює всі доступні для програмного забезпечення ключі.

Постачальник послуг, який збирається надати можливість зберігання конфіденційної інформації на віддаленій платформі повинен заздалегідь знати, що політика забезпечення безпеки даної платформи відповідає вимогам щодо

захисту даних, які будуть на неї передані. Відповідно до моделі безпеки SGX, довірена обчислювальна база (trusted computing base – TCB), що несе відповідальність за забезпечення захисту конфіденційних даних, включає в себе прошивку, апаратне забезпечення процесора, а так само програмне забезпечення тільки всередині анклаву.

Творець анклаву може використовувати спеціальні ущільнювальні та атестаційні компоненти SGX, які відповідають описаним нижче твердженням, щоб довести, що конфіденційні дані постачальника послуг будуть захищені відповідно до передбачуваного рівня безпеки:

- SGX забезпечує екземпляр анклаву засобом для запиту свідоцтва безпеки від платформи, на якій він розташований. SGX також дозволяє пов'язувати тимчасові дані анклаву з цим свідченням безпеки;
- SGX забезпечує екземпляр анклаву засобом для перевірки свідоцтв безпеки виходять від інших екземплярів анклавів на тій же платформі;
- SGX надає віддаленому об'єкту можливість перевірки свідоцтва безпеки екземпляра анклаву;
- SGX надає екземпляру анклаву спосіб для отримання ключів, які прив'язані до платформи і анклаву. SGX запобігає доступу програмного забезпечення до ключів інших екземплярів анклаву.

Архітектура SGX надає апаратні інструкції для підтримки герметизації та оцінки. Власники конфіденційних даних, які приймають модель безпеки SGX, можуть використовувати ці інструкції, щоб взаємодіяти з довіреною обчислювальною системою містить ці дані.

Для створення середовища анклаву, недовірене програмне забезпечення також використовує інструкції SGX. Ці інструкції крім іншого обчислюють криптографічну міру запущеної середовища.

Для підтримки запечатування і атестації, апаратні засоби забезпечують дві додаткові інструкції EREPORT і EGETKEY. Інструкція EREPORT надає доказову структуру, криптографічно прив'язану до апаратних засобів, використовувану для перевірки атестації. EGETKEY забезпечує програмне

забезпечення анклаву доступом до ключів, що використовуються в процесах атестації та запечатування.

3.2.1 Вимірювання анклаву

Архітектура SGX відповідає за підтвердження ідентифікаційних даних в процесах атестації і запечатування. Для кожного анклаву забезпечується два вимірювальних регістра, MRENCLAVE і MRSIGNER; MRENCLAVE забезпечує ідентифікаційні дані коду і даних анклаву, MRSIGNER забезпечує ідентифікаційні дані творця анклаву. Ці значення створюються під час побудови анклаву і записуються перед початком виконання анклаву. Щоб забезпечити точне визначення ідентифікаційних даних при атестації і запечатуванні, тільки довірена обчислювальна система має доступ на запис в ці регістри.

Ідентифікаційними даними анклаву є значення MRENCLAVE, яке є SHA-256 хеш-сумою внутрішнього журналу, який зберігає в собі всі операції, вироблені під час побудови анклаву. Журнал складається з наступної інформації:

- вміст сторінок (код, дані, стек, купа);
- відносне положення сторінок в анклаві;
- прапори безпеки, пов'язані зі сторінками.

Після того, як ініціалізація анклаву завершується за допомогою інструкції EINIT, значення MRENCLAVE не може бути змінено. Остаточне значення MRENCLAVE є SHA-256 хеш-сумою, що ідентифікує код, дані і стек, поміщені всередині анклаву, порядок і положення, в якому були розміщені сторінки анклаву і властивості безпеки кожної сторінки. Будь-яка зміна будь-якої з цих змінних призведе до іншого значення в MRENCLAVE.

Анклав має другий ідентифікатор, який використовується для захисту даних і називається запечатують ідентифікаційні дані.

Ідентифікаційні дані, що запечатують, включають герметичну сутність, ідентифікатор продукту та номер версії. Запечатувальна сутність це сутність, яка підписала анклав перед його розповсюдженням, як правило, творець анклаву. Творець анклаву представляє сертифікат (SIGSTRUCT), підписаний за допомогою RSA, який містить очікуване значення ідентифікаційних даних анклаву, MRENCLAVE і відкритого ключа запечатує сутності. Апаратне забезпечення перевіряє підпис на сертифікаті, за допомогою відкритого ключа, що міститься всередині, а потім порівнює значення виміряного MRENCLAVE з його підписаною версією. Якщо ці перевірки проходять, хеш відкритого ключа запечатує сутності зберігається в реєстрі mrsigner. Важливо зазначити, що якщо кілька анклавів підписані однією і тією ж герметичною сутністю, всі вони матимуть однакове значення MRSIGNER. Значення запечатувальних ідентифікаційних даних може бути використано для запечатування даних таким чином, що анклавів від тієї ж запечатує сутності (Наприклад, різні версії одного і того ж анклаву) можуть спільно використовувати і переносити свої запечатані дані.

3.2.2 Атестація анклаву

Атестація це процес демонстрації того, що певне програмне забезпечення було правильно розгорнуто на платформі. SGX це механізм, за допомогою якого інша сторона може переконатися, що відповідне програмне забезпечення надійно працює в межах анклаву на задіяній платформі. Для цього Архітектура SGX включає атестаційне свідоцтво, що передає наступну інформацію:

- ідентифікаційні дані атестується програмного середовища;
- детальна інформація про будь-якому незмірному стані (наприклад, режим в якому запущена програмне середовище);
- дані, пов'язані з програмним оточенням;

– криптографічна прив'язка до TCB платформи, що створює свідоцтво.

Архітектура SGX забезпечує механізм для створення атестації між двома анклавами, що працюють на одній платформі (локальну атестацію), а також механізм розширення локальної атестації, необхідний для надання атестації третій стороні за межами платформи (віддалену атестацію).

Для того, щоб отримати максимальну надійність системи, ключ атестації повинен бути пов'язаний тільки з певною TCB платформи. Якщо відбувається зміна TCB платформи, наприклад, при оновленні мікрокоду, ключ атестації платформи повинен бути замінений, щоб належним чином представляти надійність TCB.

3.2.3 Локальна атестація анклаву

У розробників застосунків може виникнути бажання або необхідність реалізації анклавів, що взаємодіють один з одним для виконання деяких загальних функцій більш високого рівня. Для цього їм потрібен механізм для аутентифікації анклавами один одного. Для цієї мети в архітектурі SGX служить команда EREPORT.

Коли анклавом викликається EREPORT, створюється підписана структура, яка називається звітом. Структура звіту містить два види ідентифікаційних даних, атрибути, пов'язані зі звітом (ці атрибути ідентифікують режими та інші властивості, встановлені в ході ECREATE), інформація про надійність апаратної TCB, а також додаткова інформація від розробників анклаву, яку потрібно передати цільовому анклаву і тег коду аутентифікації повідомлення MAC. Цільовий анклав це інший анклав, який перевіряє MAC звіту, що дозволяє йому визначити що анклав, яким був створений звіт, працює на тій же платформі.

MAC виробляється за допомогою ключа під назвою – ключ звіту. Ключ звіту відомий лише цільовому анклаву та інструкції EREPORT. Перевіряючий

(цільовий) анклав може отримати власний ключ звіту за допомогою інструкції EGETKEY. EGETKEY забезпечує анклав ключами, в тому числі ключами звіту, які можуть використовуватися для симетричного шифрування і аутентифікації. Цільовий анклав використовує ключ звіту для перерахунку MAC звіту, а також, щоб переконатися в тому, що звіт був створений анклавом, що атестується (звітує).

Кожен звіт також включає поле для даних користувачів. Це поле пов'язує дані, що знаходяться всередині анклаву, з ідентифікаційними даними анклаву (вираженими у звіті). Це поле може бути використано для розширення звіту додатковими даними, за рахунок заповнення його хеш значенням допоміжних даних, що надаються разом зі звітом. Використання поля даних користувача дозволяє анклаву побудувати протокол вищого рівня, щоб сформуванати безпечний канал між собою та іншою сутністю.

Наприклад, шляхом обміну звітами які підтверджують ідентифікаційні дані відкритих ключів Діффі-Хеллмана, які були випадковим чином згенеровані всередині анклаву. Використовуючи при цьому взаємно узгоджені параметри, анклав може згенерувати загальний секретний ключ і використовувати його для захисту обміну інформацією між собою. Архітектура Intel Підтримує генерацію випадкових значень за допомогою інструкції RDRAND, яка доступна для використання програмним забезпеченням анклаву.

На рис. 3.2 показаний приклад того, як протікає процес атестації двох анклавів на одній платформі - перевіряються ідентифікаційні дані один одного, а потім перевіряється факт того, що інша сторона працює всередині анклаву на тій же платформі, і, отже, відповідає моделі безпеки SGX.

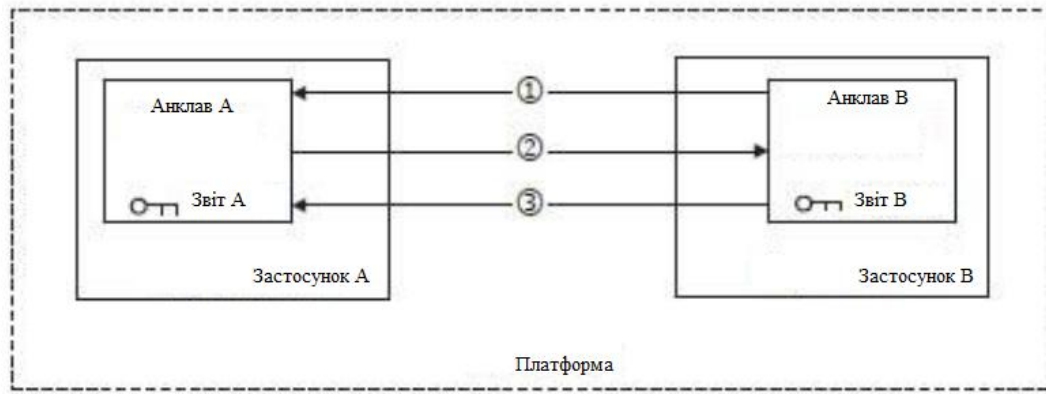


Рисунок 3.2 – Локальна атестація

- Після того, як канал зв'язку між анклавом А і В встановлений, анклав А отримує значення MRENCLAVE анклаву В важливо, що канал зв'язку на цьому етапі не обов'язково повинен бути безпечним;
- анклав А викликає команду EREPORT, передаючи MRENCLAVE анклаву В, щоб створити підписаний звіт, призначений для анклаву В. Анклав А передає свій звіт анклаву В через ненадійний канал зв'язку;
- після отримання звіту від анклаву А: анклав В викликає EGETKEY, щоб отримати свій ключ звіту, перераховує MAC звіту, і порівнює результат з MAC, що супроводжує звіт. Збіг значень MAC підтверджує, що взаємодія дійсно відбувається з анклавом, який працює на тій же платформі, що і анклав В, а значить знаходиться в рамках моделі безпеки SGX. Після того, як прошивка і апаратні компоненти TCB були перевірені, анклав В може досліджувати звіт анклаву А для перевірки програмних компонентів TCB: MRENCLAVE відображає зміст образу програмного забезпечення, що працює всередині анклаву. MRSIGNER відображає запечатуючу сутність. Тепер анклав В після може створити звіт для анклаву А, використовуючи значення MRENCLAVE зі звіту, який він отримав. Анклав В передає свій звіт анклаву А. потім анклаву А необхідно перевірити прийшов звіт аналогічним чином, для підтвердження того, що анклав В існує на тій же платформі, що і сам анклав А.

3.2.4 Дистанційна атестація анклаву

Механізм перевірки ідентифікаційних даних, який використовується процесом локальної атестації анклаву, використовує систему з симетричним ключем, в якій тільки перевіряючий звіт анклав і інструкція EREPORT, яка створює звіт, мають доступ до ключа. Створення структури, яка може бути перевірена поза платформою, вимагає використання асиметричної криптографії. SGX забезпечує спеціальний анклав, який називається анклавом цитування, який створений для віддаленої оцінки. Цитуючий анклав перевіряє звіти інших анклавів на платформі з використанням локальної атестації анклаву, яка описана вище, а потім замінює Мас звіту підписом створеної певним для конкретного пристрою асиметричним ключем. Результат цього процесу називається цитатою.

Використання стандартних асиметричних схем підпису для атестації викликає ряд питань пов'язаних з конфіденційністю в разі, коли за час всього життєвого циклу платформи використовується невелика кількість ключів. Для вирішення цієї проблеми Intel ввела розширення в схему прямої анонімною атестації (Direct Anonymous Attestation) використовуваної довірем платформним модулем (TPM) і назвала його поліпшеним ідентифікатором конфіденційності (Enhanced Privacy ID – EPID). EPID використовується у віддаленій атестації анклаву, а саме в процесі генерації цитати.

EPID – це групова схема підпису, яка дозволяє платформі підписувати об'єкти без однозначної ідентифікації себе або зв'язку між різними підписами. Замість цього, кожен підписувач належить до групи, і перевіряючий використовує відкритий ключ групи для перевірки підписів. EPID підтримує два режими підписів. У повністю анонівному режимі EPID перевіряючий не може пов'язати даний підпис з конкретним членом групи. У псевдоанонівному режимі перевіряючий має можливість визначити, чи були раніше перевірені підписи даної платформи.

Анклав цитування створює ключ EPID, який використовується для підписання атестацій на платформі, який потім перевіряється інфраструктурою EPID. Ключ EPID відображає надійність не тільки платформи, але і апаратних засобів. Тільки анклав цитування має доступ до ключа EPID, коли система працює, а ключ EPID прив'язаний до версії процесора. Таким чином, цитату можна вважати створеної самим процесором.

На рис. 3.3 показаний приклад того, як додаток може провести віддалену атестацію з перевіряючим постачальником послуг, щоб отримати від нього деякі секретні дані. Важливо, що зазвичай цей процес використовується тільки при першому з'єднанні для надання анклаву ключа зв'язку, який буде використовуватися в наступних з'єднаннях.

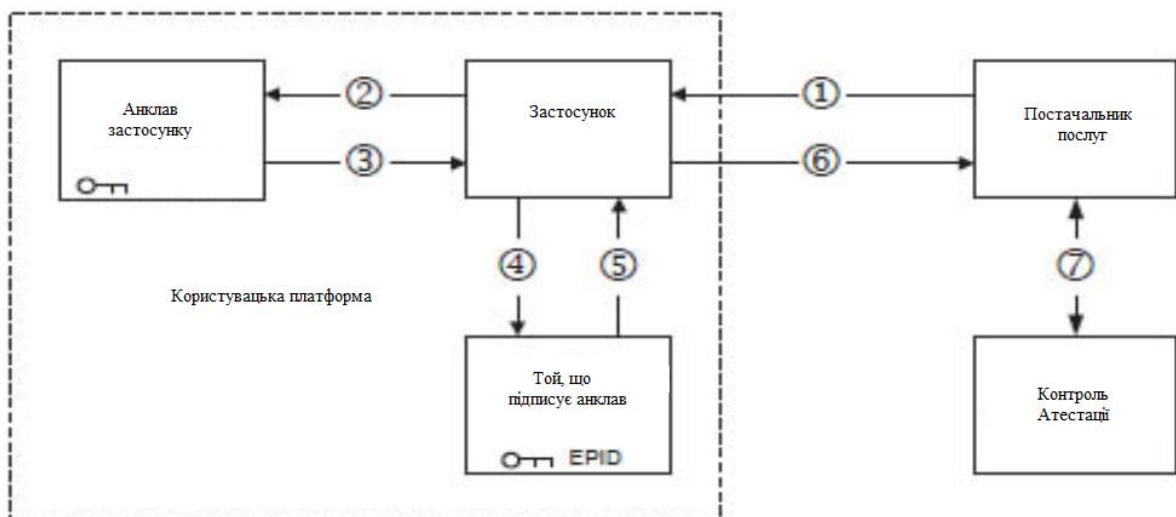


Рисунок 3.3 – Дистанційна атестація

Спочатку застосунку потрібна деяка послуга за межами платформи, воно встановлює зв'язок з системою постачальника послуг. Постачальник послуг запускає перевірку програми на те, що вона дійсно містить необхідні компоненти всередині одного або декількох анклавів.

Застосунок містить ідентифікаційні дані цитує анклаву і передає їх разом з перевіркою постачальника послуг анклаву застосунки.

Анклав створює маніфест, який включає відповідь на запит і одноразовий відкритий ключ, який буде використовуватися перевіряючим для передачі секретів назад в анклав. Потім він генерує хеш-суму маніфесту та включає його в дані користувача для команди EREPORT, яка генерує звіт, що пов'язує маніфест з анклавом. Потім анклав надсилає звіт додатку.

Застосунок направляє звіт цитующому анклаву для підпису.

Анклав цитування витягує свій ключ звіту за допомогою команди EGETKEY і перевіряє звіт. Потім анклав цитування створює структуру цитати та підписує її своїм ключем EPID. Цітіруючий анклав повертає цитату в додаток.

Застосунок надсилає цитату та будь-які пов'язані з маніфестом допоміжні дані постачальнику послуг.

Перевіряючий використовує сертифікат відкритого ключа EPID і послугу перевірки атестації для перевірки підпису цитати. Потім він перевіряє цілісність маніфесту за допомогою призначених для користувача даних, а так само перевіряє в маніфесті відповідь на запит, що посилається на кроці 1.

3.3 Запечаткування

Коли створюється екземпляр анклаву, захист його даних (їх конфіденційність і цілісність) в межах анклаву забезпечується апаратними засобами. Але коли існування анклаву закінчується, він знищується, а значить і всі дані, що існують в анклаві, втрачаються. Якщо ці дані призначені для повторного використання, анклав повинні вживати спеціальних заходів для зберігання даних за своїми межами.

Інструкція EGETKEY забезпечує доступ до ключів запечаткування, які може використовувати програмне забезпечення анклаву для шифрування і забезпечення цілісності даних. Захист від повторного відтворення даних також можливий при використанні механізму в поєднанні з іншими пропонованими службами, такими як монотонні лічильники.

При виклику EGETKEY, анклав вибирає критерії або політику, відповідно до яких анклав може отримати доступ до ключа запечатування. Ці політики потрібні для контролю доступу до конфіденційних даних майбутніми версіями анклаву.

SGX підтримує дві політики для ключів запечатування:

- запечатування за допомогою ідентифікаційних даних анклаву;
- запечатування за допомогою запечатуючих ідентифікаційних даних.

Запечатування за допомогою ідентифікаційних даних анклаву проводиться ключем, який доступний для будь-якого екземпляра точно такого ж анклаву. Це заважає майбутньому програмному забезпеченню отримувати доступ до секретів цього анклаву. Запечатування за допомогою запечатуючих ідентифікаційних даних створює ключ, який доступний деяким іншим анклавам, підписаних тією ж запечатує сутністю. Це може бути використано, щоб дозволити новішим анклавам отримати доступ до даних, що зберігаються в попередніх версіях.

Тільки запущений пізніше цей же анклав, виконавши інструкцію EGETKEY з тією ж політикою, матиме можливість отримати ключ запечатування і розшифрувати дані, які були запечатані за допомогою цього ключа при попередньому запуску.

При запечатуванні з використанням ідентифікаційних даних анклаву, EGETKEY засновує вироблення ключа на величині MRENCLAVE. Будь-яка зміна, яка впливає на MRENCLAVE, дасть інший ключ. Це призводить до різних ключів для кожного анклаву, забезпечуючи повну ізоляцію між анклавами. Побічним ефектом використання цієї політики є те, що різні версії одного і того ж анклаву також матиме різні ключі запечатування, не дозволяючи переміщення даних в автономному режимі. Ця політика корисна для тих випадків, коли старі дані не повинні бути доступними після виявлення вразливості. Наприклад, якщо це дані автентифікації облікових записів, постачальник послуг може скасувати ці облікові записи та попросити надати

нові. Доступ до старих облікових даних у більшості випадків є небажаним, і ця Політика запобігає цьому.

При запечатуванні анклаву за допомогою запечатують ідентифікаційних даних, EGETKEY засновує вироблення ключа на величині mrsigner анклаву і версії цього анклаву. MRSIGNER відображає ключ і ідентифікаційні дані запечатує сутності, яка підписала сертифікат анклаву. Перевагою запечатування за допомогою запечатуючої сутності над використанням ідентифікаційних даних анклаву є те, що вона дозволяє в автономному режимі переміщення секретних даних між версіями анклаву. Герметична сутність може підписати кілька анклавів і дати їм можливість отримати той самий ключ герметизації. Ці анклавів можуть прозоро отримати доступ до даних, які були запечатані іншим.

При запечатуванні за допомогою запечатує сутності, старе програмне забезпечення не повинно отримувати доступ до даних, створеним новішими версіями. Це необхідно, коли причиною випуску нового програмного забезпечення було виправлення проблем із безпекою. Для полегшення реалізації цього механізму, запечатує сутність має можливість прописувати Номер безпечної версії (Security Version Number – SVN) як частина запечатують ідентифікаційних даних. EGETKEY дозволяє анклаву вказати, які SVN використовувати при виробленні ключа запечатування. Коли анклав запечатує дані, він має можливість встановити мінімальне значення SVN анклаву, що дозволяє доступ до ключа ущільнення. Це захищає майбутні секрети від доступу старого вразливого програмного забезпечення, але все ще забезпечує плавний перехід до оновлення, після чого всі попередні конфіденційні дані залишаються доступними.

SVN це не те саме, що версія продукту. Продукт може мати кілька версій, з різними функціональними можливостями, але з тим же самим SVN. Відповідальність повідомити своїм клієнтам (за необхідності), які версії продукту мають однакову безпеку, лежить на творці анклаву.

Нарешті, архітектура SGX пропонує механізм, відомий як OwnerEpoch, який дозволяє власнику платформи змінити всі ключі в системі шляхом зміни одного значення. Так як OwnerEpoch автоматично враховується при запиті ключа за допомогою інструкції EGETKEY, об'єкти даних, які були запечатані з певним значенням OwnerEpoch можуть бути роздруковані тільки якщо в OwnerEpoch встановлено те ж значення.

Основна мета цього механізму полягає в тому, щоб дозволити власнику платформи заборонити доступ до всіх запечатаних секретів на платформі перед передачею платформи комусь іншому (назавжди або тимчасово). До передачі платформи, власник Платформи може змінити OwnerEpoch, в разі тимчасової передачі (наприклад, для ремонту платформи), власник Платформи може відновити OwnerEpoch до початкового значення, як тільки платформа повернеться до нього і, отже, може відновити доступ до запечатаних секретів.

3.4 Архітектура розробленого програмного забезпечення

У цій роботі метою розробки ПЗ є забезпечення конфіденційності електронної пошти за допомогою технології Intel SGX шляхом захисту даних користувачів та ключів шифрування на стороні клієнта. Як встановлено в попередніх розділах, існуючі засоби захисту конфіденційності Електронної Пошти, не усувають загрозу з боку привілейованого шкідливого програмного забезпечення на стороні клієнта. Технологія SGX однією зі своїх основних цілей має захист секретних даних від таких загроз. Таким чином, головним завданням розроблюваного ПЗ є захист секретних даних від привілейованого шкідливого ПЗ на стороні клієнта. Однак, загрози, які успішно вирішуються існуючими засобами захисту, такі як загроза перехоплення пошти при передачі даних або загроза витоку даних на стороні сервера, так само повинні вирішуватися розроблюваним програмним забезпеченням.

В існуючих засобах захисту для усунення перерахованих загроз використовується шифрування. Повідомлення шифруються на стороні

передавального повідомлення перед відправкою. Після отримання повідомлення приймаючою стороною, воно розшифровується. Конфіденційність зашифрованих даних не може бути порушена при передачі або на сервері. Однак системне програмне забезпечення на стороні клієнта може отримати дані до їх шифрування. Відповідно розробляється ПЗ так само може використовувати шифрування для вирішення загроз при передачі і зберіганні на сервері. Крім того при використанні технології SGX, шифрування дозволяє захиститися і від шкідливого системного ПЗ. Для цього необхідно, щоб передані секретні дані не залишали в незашифрованому вигляді довірену область, тобто анклав технології SGX. Таким чином, в розроблюваному програмному забезпеченні повинно здійснюватися шифрування переданих даних, причому функції шифрування і розшифрування повинні бути реалізовані в анклаві. При цьому функції відправки і прийому повідомлень працюють з секретними даними вже в зашифрованому вигляді, тому вони можуть бути реалізовані поза довіреної області, тобто в основному застосунку. При розробці застосунків, що використовують технологій SGX, необхідно прагнути до мінімізації розмірів довіреної області, тому функції, які можуть бути реалізовані в недовірєній області, повинні бути реалізовані саме там.

Для здійснення обміну зашифрованими повідомленнями між двома машинами, необхідна процедура обміну ключами. У деяких засобах захисту процедура обміну ключами відсутня. Передбачається, що для обміну шифрованими повідомленнями користувачі повинні знати секретні ключі один одного. Однак така система має серйозні недоліки, вона передбачає наявність у користувачів способу безпечного обміну ключами. А так само наявність безпечного сховища для ключів, так як при великій кількості співрозмовників, запам'ятати Ключі стає важко.

В інших захисних засобах в процедурі обміну ключами присутня проблема, яка полягає в тому, що користувачі повинні володіти відкритими ключами підписів один одного. При використанні цих ключів необхідна деяка

гарантія їх цілісності. Для вирішення цієї проблеми може використовуватися надійний сервер, який зберігає та надає ці відкриті ключі.

Технології SGX надає більш просте рішення цієї проблеми. Для її вирішення може використовуватися механізм віддаленої атестації. В поле призначених для користувача даних, переданої в ході віддаленої атестації цитати, можуть бути записані дані або хеш даних, які повинні бути підписані. Крім цього при використанні віддаленої атестації вона виконує свої стандартні функції, тобто засвідчує в тому, що програмне і апаратне забезпечення на стороні співрозмовника задовольняє необхідним вимогам безпеки. Вироблений в результаті обміну ключами, ключ шифрування зберігається в незахищеній області в зашифрованому з використанням механізму запечатування вигляді.

Основною проблемою розроблюваного програмного забезпечення є той факт, що в технології SGX не може виконуватися привілейований код, а зокрема, операції введення/виведення. Це призводить до того, що при виведенні повідомлення на екран і при введенні повідомлення з клавіатури, що захищаються дані повинні опинитися в недовірєній області. Це робить марними всі заходи щодо захисту від шкідливого системного програмного забезпечення.

У статті компанії Intel, що містить прикладні рішення, що використовують технологію SGX, для виведення на екран використовуються технологія захищеного аудіо відео шляху (Protected Audio Visual Path – PAVP), використання якого, не будучи розробником компанії Intel, не представляється можливим. Проблема введення з клавіатури не вирішена зовсім. Це показує складність вартісної проблеми. Існують різні технології, що забезпечують безпеку виведеної на екран інформації. Дослідження цих технологій, ймовірно, могло б стати темою окремої роботи.

У даній роботі проблема безпечного введення/виведення звужена до завдання не допустити потрапляння в процесі введення/виведення в

незахищену область секретних даних в явному вигляді, тобто у вигляді тексту. Представлені наступні способи вирішення цього завдання.

При виведенні повідомлення на екран, перед тим, як бути переданим в незахищену область, текст повідомлення перетворюється з зображення. У вигляді зображення він передається в незахищену область і виводиться на екран. Таким чином, в незахищеній області секретні дані присутні тільки у вигляді зображення, а в явному текстовому вигляді не залишають довірену область.

При введенні даних з клавіатури використовується екранна клавіатура. У незахищеній пам'яті присутній тільки номер натиснутої клавіші. Його перетворення в символ відбувається в довірній області. Таким чином, при введенні з клавіатури секретна інформація так само не потрапляє в явному вигляді в незахищену пам'ять. Цей механізм дозволяє обійти такий поширений вид шкідливого програмного забезпечення як *keylogger*, тобто програму, що запам'ятовує і передає зловмиснику всі дані, що вводяться з клавіатури.

Підсумковий поділ функцій між довірною і недовірною областями коду представлено в табл. 3.1.

Таблиця 3.1 – Розділення функцій у розробленому програмному забезпеченні

Довірена частина	Незахищена частина
Шифрування повідомлень	Відправлення повідомлень
Розшифрування повідомлень	Прийом повідомлень
Довірена частина обміну ключами	Незахищена частина обміну ключами
Перетворення тексту в зображення	Виведення зображення
Робота з екранною клавіатурою	Обробка кліків миші

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

У міру просування вперед інноваційні підходи, такі як архітектура з нульовим рівнем довіри, технологія блокчейн, постквантова криптографія та інтеграція Інтернету речей – безпеки, відіграватимуть вирішальну роль у захисті Електронної Пошти. Випереджаючи виникаючі загрози і постійно вдосконалюючи заходи безпеки, організації можуть знижувати ризики, захищати свої активи і підтримувати довіру до себе у все більш цифровому світі.

Отже актуальною задачею залишається програмна реалізація системи захисту електронних листів.

З метою забезпечення універсального функціонального застосунку були поставлені завдання, основні з яких:

- аутентифікація та авторизація користувачів;
- шифрування та дешифрування електронної пошти;
- управління ключами;
- цілісність та справжність електронної пошти;
- захист метаданих електронної пошти;
- безпечне зберігання електронної пошти;
- фільтрація та сканування електронної пошти;
- інтерфейс користувача;
- аудит та ведення журналу;
- відповідність вимогам і нормативні вимоги;
- продуктивність та масштабованість;
- сумісність.

4.2 Вимоги до інтерфейсу

Інтерфейс застосунку має відрізнятися лаконічністю та без зайвих проблем влаштовуватися в стандартні поштові клієнти, наприклад, в програму Microsoft Outlook.

4.3 Умови виконання програми

Нефункціональні вимоги зводяться до вимог експлуатаційних:

- IBM-сумісний персональний комп'ютер;
- процесор із тактовою частотою від 500 МГц;
- оперативна пам'ять об'ємом від 512 Мб;
- вільне місце на накопичувачі об'ємом від 2 Гб;
- встановлений застосунок Microsoft Outlook.

4.4 Методика та результати тестування

Тестування відбувалося на віртуальній машині під управлінням ОС Windows 7.

Розроблене ПЗ є плагіном, тобто додає нові функції до вже існуючого поштового клієнта. В якості поштового клієнта для поліпшення обраний Microsoft Outlook, так як він є найбільш поширеним поштовим клієнтом, а так само має найбільш зручне API для розробки плагінів. Розроблений плагін додає до Outlook вкладку з двома кнопками в стрічку з опціями, а також кнопку в форму вхідного повідомлення. Кнопка write mail відкриває форму опрацювання повідомлення, представлену на рис. 4.1.

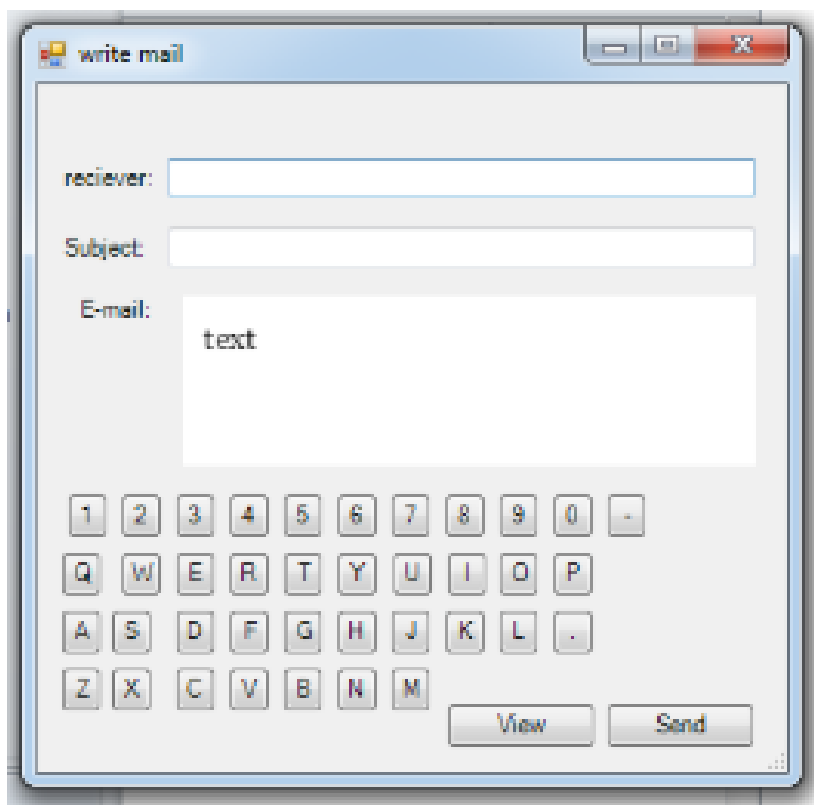


Рисунок 4.1 – Форма відправки листа

Кнопка add contac" відкриває форму Додавання контакту, яка запускає процедуру обміну ключами і представлена на рис. 4.2.

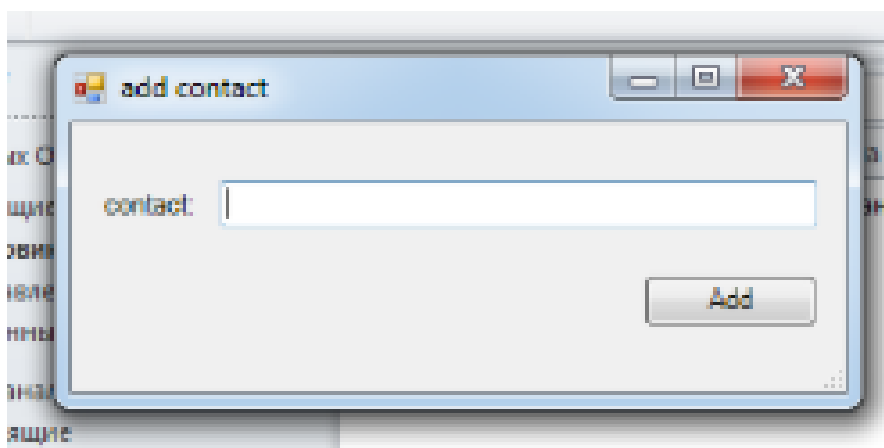


Рисунок 4.2 – Додавання контакту

Кнопка decode здійснює розшифровку повідомлення і представлена на рис. 4.3.

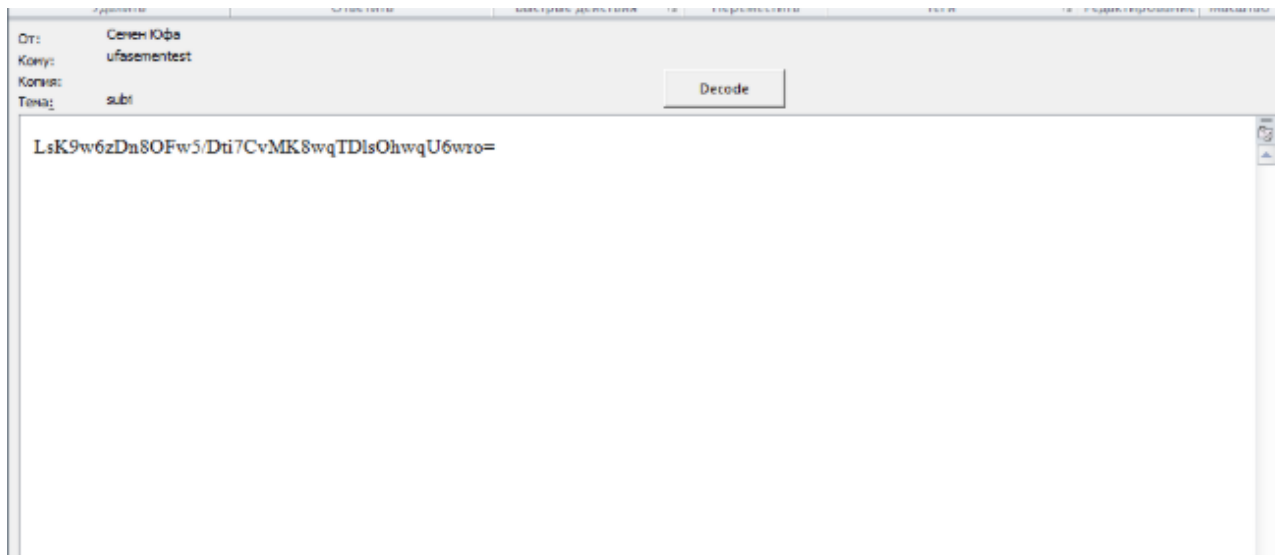


Рисунок 4.3 – Декодування повідомлення

Для процедури обміну ключами використовується приклад з SGX SDK, що називається Remote Attestation. У прикладі реалізований протокол обміну ключами аналогічний протоколу IKE, який є частиною стандарту IPSec. Цей протокол реалізує алгоритм обміну ключами Діффі-Хеллмана та використовує інфраструктуру відкритих ключів. У використаному прикладі реалізований обмін ключами Діффі-Хеллмана на еліптичних кривих. Замість інфраструктури відкритих ключів один з учасників використовує віддалену атестацію. Ключ другого визначено вручну. У розробленому плагіні обидва учасники використовують віддалену атестацію. Недоліком реалізації є те, що обмін ключами відбувається в режимі реального часу, що незручно використовувати при роботі з електронною поштою. Проте розроблений плагін вимагає знаходження обох учасників в мережі при проходженні процедури обміну ключами. Після обміну ключами Діффі-Хеллмана на еліптичних кривих з отриманого загального секрету виводиться 128-бітний ключ для шифрування алгоритмом AES. Перед закриттям програми всі використовувані в даному сеансі ключі проходять через механізм запечатування технології SGX. Таким чином, в розробленому плагіні шифрування і розшифрування проводиться з використанням алгоритму AES в режимі з контролем цілісності.

Екранна клавіатура реалізована як набір кнопок на формі відправки повідомлення. Надрукований текст виводиться у вигляді картинки при натисканні кнопки, призначеної спеціально для цього.

Отримане ПЗ реалізує шифрування контролю цілісності та безпечного обміну ключами. Реалізований обмін ключами має ту перевагу, що не вимагає від користувача роботи з інфраструктурою відкритих ключів. Однак його недоліком є необхідність його здійснення в режимі реального часу.

При цьому розроблений плагін реалізує захищені операції введення/виводу, а також захист від шкідливого системного програмного забезпечення. Цих переваг немає у існуючих засобів захисту

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію системи захисту електронних листів. Для цього було виконано ряд завдань:

- проаналізувати проблемну область захисту електронної пошти;
- дослідити існуючі механізми захисту електронного листування;
- визначити та обґрунтувати вибір технічних засобів для захисту електронного листування;
- спроектувати та реалізувати застосунок для захисту листування;
- протестувати нову технологію захисту електронного листування;
- підготувати висновки на основі результатів тестування.

Отримане ПЗ реалізує шифрування контролю цілісності та безпечного обміну ключами. Реалізований обмін ключами має ту перевагу, що не вимагає від користувача роботи з інфраструктурою відкритих ключів. Однак його недоліком є необхідність його здійснення в режимі реального часу.

При цьому розроблений плагін реалізує захищені операції введення/виводу, а також захист від шкідливого системного програмного забезпечення. Цих переваг немає у існуючих засобів захисту

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог. В майбутньому можливо вдосконалити систему додавши інші функції шифрування, щоб не вимагати від співрозмовника онлайн статусу.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Common Threats to Email Security [Electronic resource]. – Access mode: <https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-email-security/>.
2. Email security defined [Electronic resource]. – Access mode: <https://www.microsoft.com/en-us/security/business/security-101/what-is-email-security>.
3. What is Email Protection? [Electronic resource]. – Access mode: <https://www.barracuda.com/support/glossary/email-protection>.
4. Multi-layered, adaptive threat defense [Electronic resource]. – Access mode: <https://www.proofpoint.com/us/products/threat-defense>.
5. Mimecast Advanced Email Security blocks the most dangerous attacks [Electronic resource]. – Access mode: <https://www.mimecast.com/products/email-security/>.
6. Intel® Software Guard Extensions (Intel® SGX) [Electronic resource]. – Access mode: <https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/software-guard-extensions.html>.
7. Relentless Attention to Security Innovation. Our Work is Never Done [Electronic resource]. – Access mode: <https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html>.
8. What is Intel SGX (Software Guard Extensions)? [Electronic resource]. – Access mode: <https://www.trentonsystems.com/en-us/resource-hub/blog/what-is-intel-sgx>.
9. Get Started [Electronic resource]. – Access mode: <https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/get-started.html>.
10. Intel® Software Guard Extensions (Intel® SGX) SDK for Linux* OS [Electronic resource]. – Access mode: <https://www.orientatech.es/en/aplicacion-loc>

lizacion-life-360?gad_source=1&gclid=Cj0KCQjwxqayBhDFARIsAANWRnSMqn
gElEHm62tjTtYYlqkTQlK5emRRleZbumfsIMyE93WxXUs9QOEaAvNVEALw_wcB.

11. Visual studio code [Electronic resource]. – Access mode:
<https://families.google/familylink/>.

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок присвячений захисту електронних листів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація системи захисту електронних листів», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Основні вимоги до програми, що розробляється

У міру просування вперед інноваційні підходи, такі як архітектура з нульовим рівнем довіри, технологія блокчейн, постквантова криптографія та інтеграція Інтернету речей – безпеки, відіграватимуть вирішальну роль у захисті електронної пошти. Випереджаючи виникаючі загрози і постійно вдосконалюючи заходи безпеки, організації можуть знижувати ризики, захищати свої активи і підтримувати довіру до себе у все більш цифровому світі.

Отже актуальною задачею залишається програмна реалізація системи захисту електронних листів.

A.2.1 Вимоги до функціональних характеристик

З метою забезпечення універсального функціонального застосунку були поставлені завдання, основні з яких:

- аутентифікація та авторизація користувачів;
- шифрування та дешифрування електронної пошти;
- управління ключами;

- цілісність та справжність електронної пошти;
- захист метаданих електронної пошти;
- безпечне зберігання електронної пошти;
- фільтрація та сканування електронної пошти;
- інтерфейс користувача;
- аудит та ведення журналу;
- відповідність вимогам і нормативні вимоги;
- продуктивність та масштабованість;
- сумісність.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс застосунку має відрізнятися лаконічністю та без зайвих проблем влаштовуватися в стандартні поштові клієнти, наприклад, в програму Microsoft Outlook.

A.2.3 Вимоги до надійності

Головна мета розробки забезпечити конфіденційність електронного листування. Отже головна вимога до безпеки даних – використання технологій, що будуть гарантувати цілісність даних.

A.2.4 Вимоги до складу та параметрів технічних засобів

Нефункціональні вимоги зводяться до вимог експлуатаційних:

- IBM-сумісний персональний комп'ютер;
- процесор із тактовою частотою від 500 МГц;
- оперативна пам'ять об'ємом від 512 Мб;
- вільне місце на накопичувачі об'ємом від 2 Гб;

- встановлений застосунок Microsoft Outlook.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

У міру просування вперед інноваційні підходи, такі як архітектура з нульовим рівнем довіри, технологія блокчейн, постквантова криптографія та інтеграція Інтернету речей – безпеки, відіграватимуть вирішальну роль у захисті електронної пошти. Випереджаючи виникаючі загрози і постійно вдосконалюючи заходи безпеки, організації можуть знижувати ризики, захищати свої активи і підтримувати довіру до себе у все більш цифровому світі.

Отже актуальною задачею залишається програмна реалізація системи захисту електронних листів.

Б.2 Функціональне призначення

З метою забезпечення універсального функціонального застосунку були поставлені завдання, основні з яких:

- аутентифікація та авторизація користувачів;
- шифрування та дешифрування електронної пошти;
- управління ключами;
- цілісність та справжність електронної пошти;
- захист метаданих електронної пошти;
- безпечне зберігання електронної пошти;
- фільтрація та сканування електронної пошти;
- інтерфейс користувача;
- аудит та ведення журналу;
- відповідність вимогам і нормативні вимоги;
- продуктивність та масштабованість;
- сумісність.

Б.3 Опис логічної структури

Загальна схема системи за обраною темою на рис. Б.1.

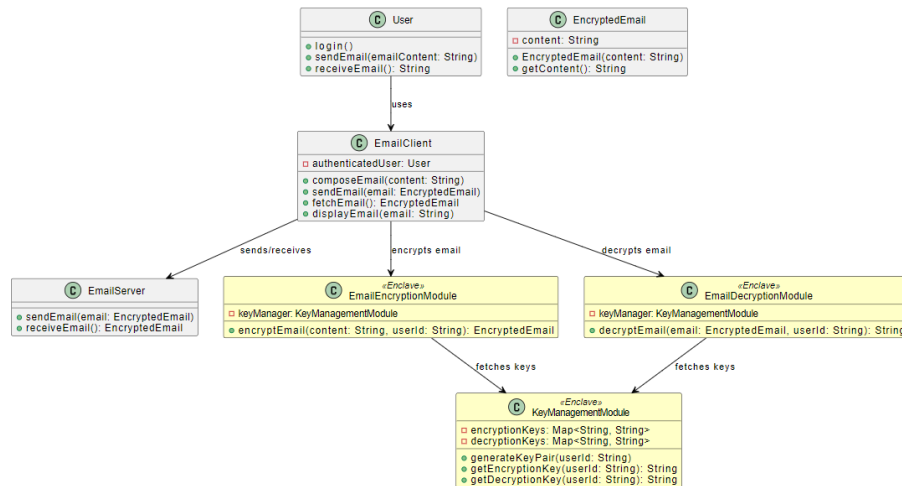


Рисунок Б.1 – Загальна схема системи за обраною темою

Взаємовідносини і взаємодії:

- користувач використовує поштовий клієнт для складання, надсилання та отримання електронних листів;
- поштовий клієнт взаємодіє з поштовим сервером для надсилання та отримання електронних листів;
- клієнт електронної пошти використовує модуль шифрування електронної пошти для шифрування вихідних електронних листів;
- клієнт електронної пошти використовує модуль розшифровки Електронної Пошти для розшифровки вхідних електронних листів;
- модуль шифрування електронної пошти та модуль дешифрування електронної пошти взаємодіють з модулем управління ключами для отримання необхідних ключів для шифрування та дешифрування.

На цій діаграмі класів представлений високорівневий огляд компонентів і їх взаємодії в системі захисту електронної пошти на базі Intel SGX, що забезпечує безпечну електронну переписку за рахунок використання анклавів для конфіденційних операцій.

Б.4 Використані технічні засоби

Нефункціональні вимоги зводяться до вимог експлуатаційних:

- IBM-сумісний персональний комп'ютер;
- процесор із тактовою частотою від 500 МГц;
- оперативна пам'ять об'ємом від 512 Мб;
- вільне місце на накопичувачі об'ємом від 2 Гб;
- встановлений застосунок Microsoft Outlook.

ДОДАТОК В
Текст програми

```

long  getFrompsFl(const  char*  psFl_name,  unsigned
char** pp_dat)
{
    psFl *inpsFl;
    errono_t erro;
    long fmas = 0;
    erro = fopen_s(&inpsFl, psFl_name, "rb");
    if (erro == 0)
    {
        fseek(inpsFl, 0L, SEEK_END);
        fmas = ftell(inpsFl);
        rewind(inpsFl);
        *pp_dat  =  (unsigned  char*)calloc(fmas,
masof(unsigned char));
        unsigned char* tmp = *pp_dat;
        mas_t len = fget(tmp, masof(unsigned char),
fmas, inpsFl);
        fclose(inpsFl);
    }
    else
    {
        printf("Failed  to  open  psFl  %s",
psFl_name);
    }
    return fmas;
}

//set dat from p_dat to psFl system
void  setTopsFl(const  char*  psFl_name,  const
unsigned char* p_dat, mas_t len)
{

```

```

    psFl *outpsFl;
    errono_t erro;
    erro = fopen_s(&outpsFl, psFl_name, "wb");
    if (erro == 0)
    {
        for (int i = 0; i < len; i++)
        {
            fputc(p_dat[i], outpsFl);
        }
        fclose(outpsFl);
    }
    else
    {
        printf("Failed to open psFl %s",
psFl_name);
    }
}

//Ocall (will be executed within the enclave)
void sevt_set_dat(const char* psFl_name, const
unsigned char* p_dat, mas_t len)
{
    setTopsFl(psFl_name, p_dat, len);
}

//Ocall (will be executed within the enclave)
void sevt_get_dat(const char* psFl_name, unsigned
char** pp_dat, mas_t* len)
{
    *len = getFrompsFl(psFl_name, pp_dat);
}

```

```

typedef enum tsevt_reg {
    START,
    VERIFY,
    SIGN,
}sevt_reg_t;

int main(int argc, char* argv[])
{
    gtr_enclave_id_t    eid;
    gtr_status_t        ret = gtr_SUCCESS;
    gtr_launch_token_t token = { 0 };
    const      wchar_t*      ENCLAVE_psFl_NAME      =
L"System_Security_Trusted.signed.dll";
    int updated = 0;

    char* failedInMethod = (char*)"SampleMethod";
    int res = -1;

    sevt_reg_t reg=START;
    int command;
    char* sig_psFl_name = NULL;
    char* sent_key_psFl_name = NULL;
    char* sealed_dat_name = NULL;
    char* mesg = NULL;

```

```

        char    usg[]    = "usg:  %s    [-e  psFl_name]    [-i
sealed_keypsFl]    [-s  mesg_to_sign]    [-v  mesg_to_verify -S
signature_psfFl]\n";

        while      ((command      =      getopt(argc,      argv,
"e:i:s:v:S:")) != -1)
            switch (command)
            {
            case 'e':
                sent_key_psfFl_name = optarg;
                break;
            case 's':
                mesg = optarg;
                reg = SIGN;
                break;
            case 'v':
                mesg = optarg;
                reg = VERIFY;
                break;
            case 'S':
                sig_psfFl_name = optarg;

                break;
            case 'i':
                sealed_dat_name = optarg;
                break;
            case '?':
                fprintf(stderr, usg, argv[0]);

                return 1;
            default:

```

```

        abort();
    }
    if(reg == START)
        fprintf(stderr, usg, argv[0]);

    //Check if gtr can be enabled and activate
enclave
    gtr_device_status_t gtr_device_status;

    ret = gtr_enable_device(&gtr_device_status);

    if (ret != gtr_SUCCESS) {
        failedInMethod =
(char*)"gtr_enable_device";
        goto error;
    }

    ret = gtr_create_enclave(ENCLAVE_psFl_NAME,
gtr_DEBUG_FLAG, &token, &updated, &eid, NULL);

    if (ret != gtr_SUCCESS) {
        failedInMethod =
(char*)"gtr_create_enclave";
        goto error;
    }

    ret = sevt_init(eid, &res, sealed_dat_name);
    if (ret != gtr_SUCCESS || res != gtr_SUCCESS) {

```

```

        failedInMethod = (char*)"sevt_init";
        goto error;
    }

    switch (reg)
    {
    case VERIFY:
        if (sig_psFl_name != NULL)
        {
            gtr_ec256_signature_t* sig;
            getFrompsFl(sig_psFl_name,      (unsigned
char**) &sig);

            ret = sevt_verify(eid, &res, mesg,
(void*)sig, masof(gtr_ec256_signature_t));
            if (ret != gtr_SUCCESS || res !=
gtr_EC_VALID) {
                failedInMethod =
(char*)"sevt_verify";
                goto error;
            }
            printf("\nSignature      of      mesg      %s
successfully verified!\n", mesg);
            break;
        }
        else
        {
            fprintf(stderr, "Signature psFl not
specified");

```

```

        goto erroor;
    }

    case SIGN:
        gtr_ec256_signature_t sig;
        ret = sevt_sign(eid, &res, mesg,
            (void*)&sig, masof(gtr_ec256_signature_t));
        if (ret != gtr_SUCCESS || res !=
gtr_SUCCESS) {
            failedInMethod = (char*)"sevt_sign";
            goto erroor;
        }
        printf("\nSignature of mesg %s successfully
signed!\n", mesg);
        break;
    default:
        fprintf(stderr, "no reg specified (-v or -
s)");
        goto erroor;
    }
    if (sent_key_psFl_name != NULL)
    {
        ret=sevt_seal_keys(eid, &res,
sent_key_psFl_name);
        if (ret != gtr_SUCCESS || res !=
gtr_SUCCESS)
        {
            failedInMethod =
(char*)"sevt_seal_keys";
            goto erroor;
        }
    }

```



```

int optreset;                                /* reset getopt */

char*optarg;                                /* argument associated
with option */
char BADCH = (int) '?';
char BADARG = (int) ':';
char* EMSG = (char*) "";
int getopt(int nargc, char * const nargv[], const
char *ostr)
{
    static char *place = EMSG;                /*
option letter processing */
    const char *oli;                          /*
option letter list index */

    if (optreset || !*place) {                /*
update scanning pointer */
        optreset = 0;
        if (optind >= nargc || *(place =
nargv[optind]) != '-') {
            place = EMSG;
            return (-1);
        }
        if (place[1] && *++place == '-') {    /*
found "--" */
            ++optind;
            place = EMSG;
            return (-1);
        }
    }

```

```

    }
    /*
option letter okay? */
    if ((optopt = (int)*place++) == (int)':' ||
        !(oli = strchr(ostr, optopt))) {
        /*
        * if the user didn't specify '-' as an
option,
        * assume it means -1.
        */
        if (optopt == (int) '-')
            return (-1);
        if (!*place)
            ++optind;
        if (opterro && *ostr != ':')
            (void)printf("illegal option -- %c\n",
option, optopt);
        return (BADCH);
    }
    if (*++oli != ':') {
        /* don't
need argument */
        optarg = NULL;
        if (!*place)
            ++optind;
    }
    else {
        /* need
an argument */
        if (*place)
            /* no white
space */
            optarg = place;
    }

```

```

else if (nargc <= ++optind) {      /* no arg
*/
    place = EMSG;
    if (*ostr == ':')
        return (BADARG);
    if (opterro)
        (void)printf("option requires an
argument -- %c\n", optopt);
    return (BADCH);
}
else                                /* white
space */
    optarg = nargv[optind];
    place = EMSG;
    ++optind;
}
return (optopt);                    /* dump
back option letter */
}

```

ДОДАТОК Д
Слайди презентації

**Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмна реалізація системи захисту електронних листів

Виконав: Владислав БІЛЯК

ст. групи КНТ-121сп

Керівник: Сергій ЛЕОЩЕНКО

д-р філософ., ст. викладач

Рисунок Д.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес програмної реалізації системи захисту електронних листів.

Предмет дослідження – програмні засоби для реалізації системи захисту електронних листів.

Метою роботи є програмна реалізація системи захисту електронних листів.

Рисунок Д.2 – Слайд 2

Порівняння існуючих аналогів

	S/MIME	OpenPGP	SecureGmail
Тип шифрування	Асиметричний	Асиметричний	Симетричний
Вимоги до сертифікату	Потрібні цифрові сертифікати	Використовує самопідписані сертифікати	Сертифікати не потрібні
Інтеграція	Вбудована підтримка в більшості поштових клієнтів	Потрібні плагіни	Розширення браузера для Gmail
Простота використання	Помірна складність	Від помірної до високої складності	Висока простота використання
Сумісність	Широкий спектр	Широкий спектр	Доступно лише для Gmail та Chrome
Ідентифікація	Так, забезпечує підписання електронної пошти та шифрування	Так, забезпечує підписання електронної пошти та шифрування	Немає цифрових підписів (шифрування на основі пароля)
Управління ключами	Управляється центрами сертифікації	Керований користувачем	Немає управління ключами
Модель довіри	Ієрархічний	Web of Trust	Довіра до спільного паролю
Рівень безпеки	Високий	Високий рівень	Помірний

3

Рисунок Д.3 – Слайд 3

Постановка завдання

З метою забезпечення універсального функціонального застосунку були поставлені завдання, основні з яких:

- аутентифікація та авторизація користувачів;
- шифрування та дешифрування електронної пошти;
- управління ключами;
- цілісність та справжність електронної пошти;
- захист метаданих електронної пошти;
- безпечне зберігання електронної пошти;
- фільтрація та сканування електронної пошти;
- інтерфейс користувача;
- аудит та ведення журналу;
- відповідність вимогам і нормативні вимоги;
- продуктивність та масштабованість;
- сумісність.

4

Рисунок Д.4 – Слайд 4

Порівняння мов програмування

	C++	Java	Python
Підтримка технології Intel SGX	Напрямую	Через бібліотеку	Через рушій
Захист даних	Високий рівень	Високий рівень	Середній рівень
Продуктивність	Дуже висока	Висока	Середня
Можливості низькорівневого програмування	Присутні	Присутні	Відсутні
Сумісність та кросплатформність	Кросплатформна мова, сумісна	Кросплатформна (вимагає додаткових налаштувань)	Кросплатформна (вимагає додаткових налаштувань)

5

Рисунок Д.5 – Слайд 5

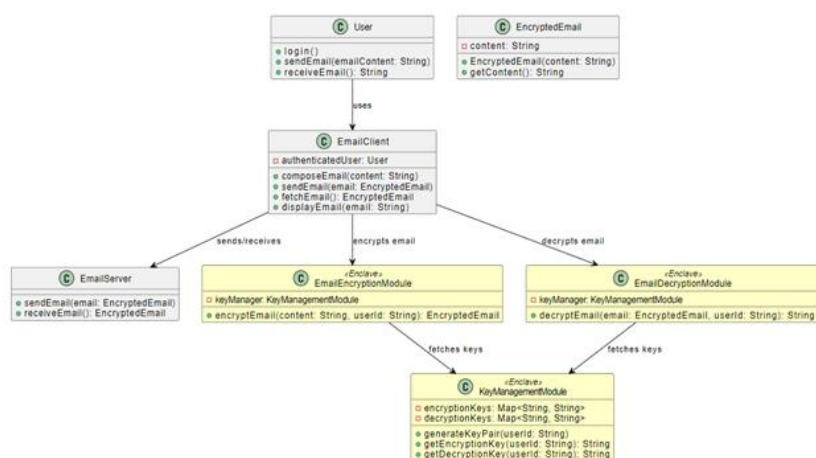
Порівняння середовищ розробки

	VS Code	Eclipse	CLion
Ліцензія, розповсюдженість	Безкоштовно, вільна ліцензія	Безкоштовно, вільна ліцензія	Безкоштовно/ платно
Налаштування та гнучкість	Високий рівень	Високий рівень	Середній рівень
Рефакторинг	Високий рівень	Середній рівень	Високий рівень
Продуктивність	Високий рівень	Високий рівень	Середній рівень
Підключення сторонніх розробок	Легко	Легко	Тільки з каталогу

6

Рисунок Д.6 – Слайд 6

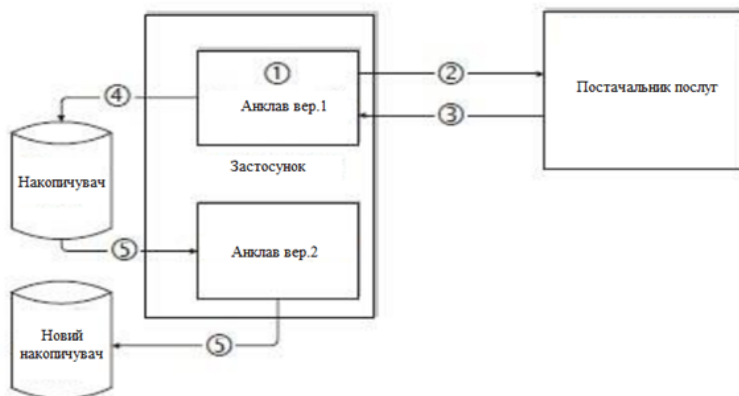
Загальна схема застосунку



7

Рисунок Д.7 – Слайд 7

Життєвий цикл програми використовуючи Intel SGX



8

Рисунок Д.8 – Слайд 8

Схема доступу до пам'яті

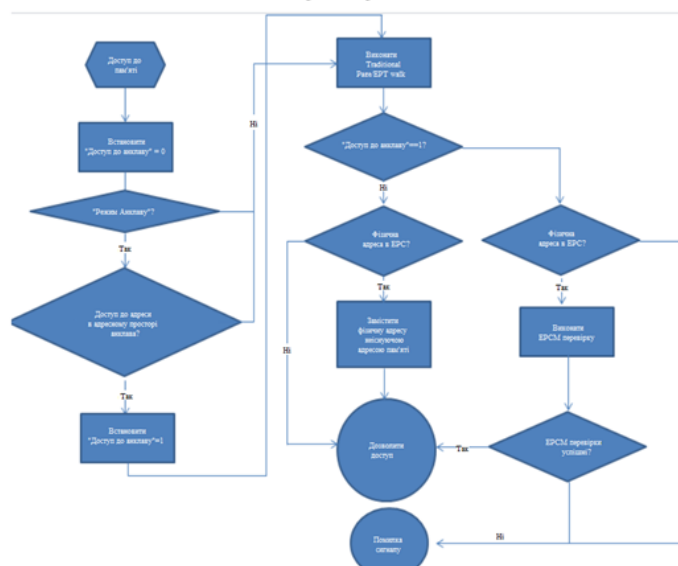


Рисунок Д.9 – Слайд 9

Тестування

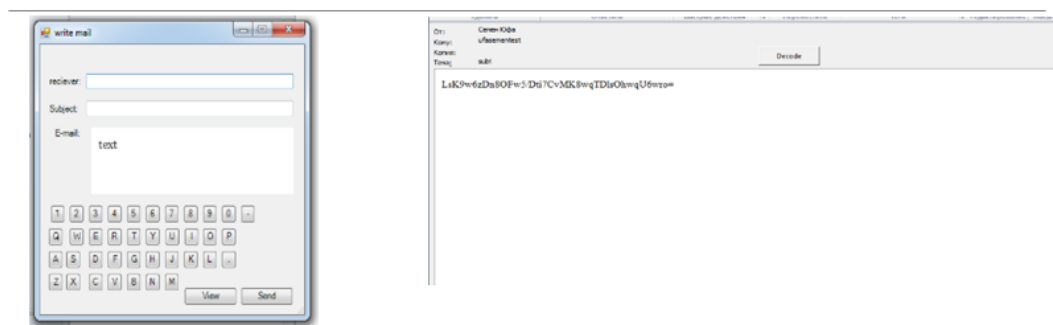


Рисунок Д.10 – Слайд 10

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію системи захисту електронних листів.

Для цього було виконано ряд завдань:

- ❖ проаналізувати проблемну область захисту електронної пошти;
- ❖ дослідити існуючі механізми захисту електронного листування;
- ❖ визначити та обґрунтувати вибір технічних засобів для захисту електронного листування;
- ❖ спроектувати та реалізувати застосунок для захисту листування;
- ❖ протестувати нову технологію захисту електронного листування;
- ❖ підготувати висновки на основі результатів тестування.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена система є зручною для використання та виконує всі необхідні функції. В майбутньому можливо вдосконалити систему додавши інші функції шифрування, щоб не вимагати від співрозмовника онлайн статусу.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.