Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

# БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Програмна реалізація шифрованого каналу передавання стеганографічних ключів для закритих систем

Виконав: студент 4 курсу, гр. 1КІТС-20б
спеціальності 125 – Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем

_____ Михайлина Б. В._____

Керівник: к.т.н., доц. каф. МБІС
_____Адлер О.О._____

«____»_____2024 р.

Рецензент: к.т.н., доцент каф. ОТ
_____Колесник І.С._____

«____»_____2024 р.

**Допущено до захисту**
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
«____»_____2024 р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

**ЗАТВЕРДЖУЮ**
**Голова секції УБ, кафедра МБІС**

_____ **д.т.н., проф. Яремчук Ю.Є.**
“ 22 ” березня 2024 р.

**З А В Д А Н Н Я**
**на бакалаврську дипломну роботу студенту**
Михайлині Богдану Володимировичу
(прізвище, ім`я, по-батькові)

1. Тема роботи Програмна реалізація шифрованого каналу передавання стеганографічних ключів для закритих систем
Керівник роботи к.т.н., доцент кафедри МБІС Адлер Оксана Олександрівна
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 11 ” березня 2024 року № 80
2. Термін подання студентом роботи_____
3. Вихідні дані до роботи Електронні джерела та наукові статті по темі. Програмне забезпечення, яке стосується теми бакалаврської дипломної роботи.
4. Зміст текстової частини:
Теоретичний аналіз предметної області. Проектування програмного продукту. Програмна реалізація.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов’язкових креслень)
NFT зображення. Блок схема роботи системи. Use Case діаграма. MVC архітектура. MVP архітектура. MVVM архітектура. Встановлення розширення. Швидкість генерації і вставки. Швидкість обробки даних. Швидкість перевірки. Головна сторінка. Форма авторизації. Створення NFT зображення. Швидкість перевірки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Розділ I | Адлер О.О., к.т.н., доц. каф. МБІС | | |
| Розділ II | Адлер О.О., к.т.н., доц. каф. МБІС | | |
| Розділ III | Адлер О.О., к.т.н., доц. каф. МБІС | | |

7.Дата видачі завдання 22 березня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Аналіз предметної області | 22.03.2024 | 30.03.2024 | Виконано |
| 2 | Аналіз методів розв'язання задачі | 01.04.2024 | 10.04.2024 | Виконано |
| 3 | Постановка задачі розробки програми | 11.04.2024 | 18.04.2024 | Виконано |
| 4 | Дослідження інформаційних джерел | 19.04.2024 | 30.04.2024 | Виконано |
| 5 | Аналіз варіантів роботи програми | 01.05.2024 | 10.05.2024 | Виконано |
| 6 | Розробка алгоритму роботи | 11.05.2024 | 18.05.2024 | Виконано |
| 7 | Реалізація програми | 19.05.2024 | 25.05.2024 | Виконано |
| 8 | Тестування програми | 26.05.2024 | 28.05.2024 | Виконано |
| 9 | Оформлення матеріалів до захисту БДР | 29.05.2024 | 07.06.2024 | Виконано |

**Студент** _____ Михайлина Б.В.
( підпис ) (прізвище та ініціали)

**Керівник роботи** _____ Адлер О.О.
(підпис ) (прізвище та ініціали)

# АНОТАЦІЯ

Бакалаврська дипломна робота складається із 60 сторінок формату А4, на яких є 14 рисунків, список використаних джерел містить 30 найменувань.

Тема роботи: «Метод передавання стеганографічних ключів у закритих системах через шифрований канал».

Метою роботи є розроблення методу передавання стеганографічних ключів у закритих системах через шифрований канал. В роботі проаналізовано існуючі методи шифрування та стеганографії, обрано оптимальні алгоритми для реалізації шифрованого каналу, а також розроблено програмне забезпечення для захищеного обміну ключами.

Дослідження полягає у створенні ефективного та безпечного механізму передавання стеганографічних ключів, що забезпечить високий рівень конфіденційності та захисту даних у закритих системах. Запропоноване рішення комбінує переваги симетричних та асиметричних алгоритмів шифрування для досягнення максимальної безпеки та ефективності. Результати роботи включають розроблений метод, програмне забезпечення та експериментальну оцінку його ефективності та безпеки. Робота має високу практичну цінність і може бути застосована у різних сферах, де необхідно забезпечити високий рівень захисту інформації.

**Ключові слова:** інформаційні технології, шифрування, обмін ключами, захист інформації, безпека даних.

## ABSTRACT

The bachelor's thesis consists of 60 pages of A4 format, with 14 figures, , the list of references contains 30 titles.

Theme of the work: "A method for transmitting steganographic keys in closed systems through an encrypted channel".

The aim of the work is to develop a method for transmitting steganographic keys in closed systems over an encrypted channel. The paper analyzes existing encryption and steganography methods, selects optimal algorithms for implementing an encrypted channel, and develops software for secure key exchange.

The research is aimed at creating an efficient and secure mechanism for the transmission of steganographic keys, which will ensure a high level of confidentiality and data protection in closed systems. The proposed solution combines the advantages of symmetric and asymmetric encryption algorithms to achieve maximum security and efficiency. The results of the work include the developed method, software, and experimental evaluation of its efficiency and security. The work is of high practical value and can be applied in various fields where it is necessary to ensure a high level of information protection.



**Keywords:** information technology, encryption, key exchange, information protection, data security.

# ЗМІСТ

# ВСТУП

**Актуальність.** У світі закритих систем, забезпечення безпеки інформації є ключовою задачею для захисту конфіденційності та цілісності даних у цифровому середовищі. Особливо це стосується передавання стеганографічних ключів - важливого ресурсу для захисту інформації в закритих системах, які використовуються в різноманітних сферах, включаючи військові та урядові організації. Проте, існуючі методи передавання ключів залишаються вразливими перед можливими атаками зловмисників через недостатній рівень захисту та шифрування. Ця проблема стала стимулом для мого дослідження, спрямованого на розробку програмної реалізації шифрованого каналу передавання стеганографічних ключів для закритих систем. Актуальність цього дослідження підкреслюється не лише потребою в захисті конфіденційної інформації, а й загальною необхідністю у підвищенні рівня кібербезпеки в закритих системах.

**Мета роботи.** Метою даного дослідження є створення програмної реалізації шифрованого каналу передавання стеганографічних ключів для закритих систем, що забезпечить безпеку передавання даних завдяки застосуванню шифрування та інших криптографічних методів.

**Завдання дослідження.** Для досягнення мети було поставлено та вирішено такі завдання:

- проаналізувати існуючі методи передавання стеганографічних ключів;
- виокремити переваги та недоліки існуючих методів, зокрема, відсутність ефективного шифрування;
- запропонувати концепцію нового методу, який поєднує переваги існуючих аналогів та додає новий рівень безпеки завдяки застосуванню шифрування даних;
- розглянути технічні аспекти реалізації цього методу, зокрема вибір мови програмування, розроблення архітектури системи та теоретична розроблення модуля шифрування даних та стеганографії;
- реалізувати програмне забезпечення для адаптивної оптимізації передавання стеганографічних ключів на основі запропонованого методу;

– провести тестування та оцінку ефективності розробленої системи.

**Об'єкт дослідження.** Процес передавання стеганографічних ключів у закритих системах з механізмом захисту конфіденційності, цілісності та доступності цих даних під час їх передачі.

**Предмет дослідження.** Методи управління передаванням стеганографічних ключів з інтеграцією криптографічних методів для забезпечення безпеки даних.

**Практична цінність роботи.** Практична цінність роботи полягає у можливості впровадження розробленої системи у різні галузі для забезпечення високого рівня захисту інформації. Розроблена система управління передаванням стеганографічних ключів з використанням криптографічних методів дозволить автоматизувати процеси генерації, розподілу та управління ключами, що значно підвищить стійкість системи до атак та забезпечить конфіденційність і цілісність даних.

# 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі було здійснено теоретичний аналіз предметної області з метою дослідження методів та підходів у рамках заданої теми. Описано основні поняття та терміни, що стосуються досліджуваної сфери, включаючи визначення ключових концепцій та їх взаємозв'язків. Наведено детальний опис предметної області, що дозволяє зрозуміти контекст та специфіку досліджуваної проблеми. Проведено аналіз важливості та актуальності розробки в сучасних умовах, показано, чому досліджувана тема є критично важливою та які проблеми вона покликана вирішити. Зазначено, як сучасні виклики та потреби впливають на необхідність даної розробки. Виконано огляд існуючих рішень та аналогів, які вже застосовуються для вирішення подібних завдань. Порівняно різні методи та інструменти, що використовуються в поточній практиці, з метою визначення їх сильних та слабких сторін. Виявлено прогалини, які потребують подальшого дослідження та удосконалення.

## 1.1 Предметна область досліджуваної сфери

Забезпечення безпеки інформації у закритих системах є важливим завданням, особливо в умовах зростання кіберзагроз. Одним з підходів до захисту конфіденційних даних є використання стеганографії та шифрування. Стеганографія дозволяє приховувати існування передавання даних, тоді як шифрування забезпечує захист вмісту повідомлень. Програмна реалізація шифрованого каналу передавання стеганографічних ключів є складним завданням, яке потребує врахування багатьох аспектів безпеки та ефективності.

Стеганографія – це метод приховування інформації у інших даних, так щоб існування прихованої інформації було непомітним для неавторизованих користувачів. Одним з поширених методів стеганографії є вбудовування повідомлень у цифрові зображення, аудіо або відео файли [1].

Шифрування – це процес перетворення інформації в такий вигляд, який

неможливо прочитати без спеціального ключа. Існують різні алгоритми шифрування, включаючи симетричні (наприклад, AES) та асиметричні (наприклад, RSA) методи.

Закриті системи – це інформаційні системи, доступ до яких має обмежене коло користувачів. Безпека таких систем є критично важливою, оскільки вони можуть містити конфіденційну або стратегічно важливу інформацію [2].

Реалізація системи в якій буде забезпечена можливість передачі даних у зображеннях є актуальним проектом адже NFT (Non-Fungible Token) є відносно новим та інноваційним феноменом у сфері цифрових активів, який використовує технологію блокчейну для створення унікальних, невзаємозамінних токенів. Ці токени можуть представляти різноманітні цифрові активи, включаючи зображення, музику, відео, та інші форми цифрового контенту. Предметна область досліджуваної сфери – це передача NFT зображень, яка є актуальною темою завдяки стрімкому розвитку цифрового мистецтва та зростанню інтересу до блокчейн-технологій.

NFT або невзаємозамінні токени – це унікальні цифрові активи, які зберігаються на блокчейні. Вони відрізняються від звичайних криптовалют, таких як біткойн або ефіріум, тим, що кожен NFT має унікальні характеристики і не може бути обміняний на інший токен еквівалентної вартості [3].

Перші спроби створити цифрові активи, які можна було б колекціонувати та обмінювати, датуються серединою 2010-х років. Одним з перших відомих проєктів у цій сфері стали "CryptoPunks" – колекція 10,000 унікальних персонажів, створених компанією Larva Labs. Цей проєкт був запущений у 2017 році на платформі Ethereum і швидко набув популярності, заклавши основу для подальшого розвитку NFT.

Процес передачі NFT зображень включає кілька ключових етапів:

− створення NFT – художник або власник створює NFT, використовуючи смарт-контракт, що містить унікальні метадані, які описують зображення;

− збереження зображення – самі зображення можуть зберігатися на

децентралізованих платформах зберігання даних, таких як IPFS (InterPlanetary File System);

− передача NFT – власник може передати NFT іншому користувачу, ініціюючи транзакцію на блокчейні. Смарт-контракт автоматично оновлює інформацію про власника токена.

Блокчейн – це список записів, які називаються блоками, які пов'язані за допомогою шифрування. Кожен блок містить зашифрований хеш, мітку часу та дані транзакцій попереднього блоку. Позначка часу доводить, що дані транзакцій існували в той час, коли блок був випущений для введення хешу. Блок містить хеш попереднього блоку, який утворює ланцюжок, але кожного жовтня додатковий блок підсилює попередній блок. Це означає, що якщо 1 блок у ланцюжку буде змінено, незабаром стане зрозуміло, що він був підроблений. Якщо хакер хоче зруйнувати блокчейн-систему, необхідно замінити кожен блок в ланцюжку кожної розподіленої версією мережі, це нереально[4].

Блокчейни, такі як біткойн і Ефіріум, неухильно зростають у міру додавання блоків в ланцюжок, що супроводжується значними поліпшеннями в області безпеки. Довіра стала основоположним принципом створення цифрових валют. Якщо хтось створює нову валюту під назвою X доларів, як ми можемо допомогти їм отримати мільйон x доларів або вкрасти ваші x доларів для себе, біткойн був розроблений для вирішення цієї проблеми за допомогою певної бази даних, яка називається блокчейн. У найпоширеніших базах даних, таких як бази даних SQL, є адміністратори, які можуть змінювати записи (наприклад, дайте собі мільйон x доларів). Блокчейн відрізняється тим, що його не існує administrators.It ним керують люди, які ним користуються. Крім того, Біткойн не можна підробляти, зламати або використовувати 2 рази, тому люди, які володіють цими грошима, можуть вважати, що вони мають певну цінність. Децентралізований блокчейн може використовувати приватну передачу повідомлень та розподілену мережу. Одним із ризиків відсутності децентралізації є" атака 51%", при якій центральні органи влади контролюють більше половини мережі і можуть вільно маніпулювати цим

конкретним записом блокчейна.Це збільшує вартість в 2 рази.

Мережі блокчейнів не мають централізованих точок вразливості, які можуть бути використані хакерами. Так само немає центральної точки відмови. Методи безпеки blockchain включають використання шифрування з відкритим ключем. Відкритий ключ (довгий рядок випадкових чисел) – це адреса в ланцюжку блоків. Токени значень, відправлені по мережі, реєструються як такі, що належать цій адресі. Приватний ключ схожий на пароль і дозволяє власнику отримувати доступ до цифрових активів або коштів через інші взаємодії з різними функціями, які в даний час підтримує блокчейн. Кожен вузол децентралізованої системи має копію ланцюжка блоків. Якість даних підтримується за рахунок реплікації великих обсягів баз даних і надійності обчислень. Немає центральної" офіційної " копії, і жодному користувачеві не можна довіряти більше, ніж іншим користувачам. Процес надсилається в мережу за допомогою програмного забезпечення [5].

Вузол майнінгу перевіряє транзакцію, вставляє в створені ним блоки і передає завершені блоки іншим вузлам. Щоб виконати операцію, вам потрібно вирішити дуже складну математичну задачу, щоб знайти одноразовий номер, що становить прийнятий хеш. Це завдання вирішують Майнер-комп'ютери, які беруть комісію за транзакції в мережі. Оскільки одноразовий номер має 32 біти, а хеш-256, існує близько 40 мільярдів комбінацій одноразових і хеш-кодів, які, можливо, доведеться витягти, перш ніж буде знайдена правильна комбінація одноразових і хеш-кодів. Коли це трапляється, шахтарі кажуть, що знайшли "золотий одноразовий номер" і що їх блок прив'язаний до ланцюга. Процес виконується таким чином. На перший погляд система дуже приваблива, але є й інший аспект AMI. Конфіденційність у мережі blockchain захищає користувачів від злому та захищає їх конфіденційність, а також дозволяє здійснювати незаконні транзакції та дії в мережі blockchain. Найбільш часто цитованим прикладом використання блокчейну для незаконних транзакцій є, мабуть, Шовкова дорога, онлайн-неділя наркотиків, що підтримується DarkWeb, яка діяла з 2011-2 по 2013-10 роки і була закрита ФБР.

Веб-сайт дозволяє користувачам переглядати веб-сайт і здійснювати незаконні покупки в біткойнах або інших криптовалютах, не відстежуючись браузером Tor. Чинні правила США вимагають, щоб постачальники фінансових послуг дізнавалися про своїх клієнтів при відкритті облікового запису, перевіряли особу кожного клієнта і вносили в список терористичні організації, клієнти яких відомі або знаходяться в них. сумнівний. Вам буде запропоновано переконатися, що ваш обліковий запис не включений у ваш обліковий запис. Цю систему можна розглядати як позитивну, так і негативну. Це дозволить кожному отримати доступ до фінансових рахунків, але полегшить здійснення правопорушниками транзакцій. Багато хто стверджує, що корисне використання криптовалют, таких як банківська справа, в небанківському світі переважає неадекватне використання криптовалют, особливо коли велика частина незаконної діяльності все ще здійснюється з використанням коштів, які неможливо відстежити [6].

Смарт-контракти – це програми, що працюють на блокчейні Ethereum. Це набір кодів (функцій) і даних (статусу), що містяться за певною адресою в блокчейне Ethereum. Смарт-контракти-це свого роду обліковий запис Ethereum. Це означає, що вони мають баланс і можуть надсилати транзакції через мережу. Однак вони не контролюються користувачем, а розподіляються по мережі і працюють за розкладом.

Обліковий запис Користувача може взаємодіяти зі смарт-контрактом, надсилаючи транзакцію, яка виконує функції, визначені в смарт-контракті. Розумні контракти визначають правила, як і звичайні контракти, і можуть застосовуватися автоматично за допомогою коду. Будь-хто може створити смарт-контракт і розгорнути його в мережі.

Укладення смарт-контракту технічно є транзакцією, тому вам доведеться заплатити за простий переказ газу (плата) і ETH. Але вартість газу для виконання контракту набагато вище. Крім того, для створення NFT потрібне сховище, посилання на яке може зберігати медіафайли, зазначені в метаданих блоку блокчейну. Цей тип сховища є міжпланетною файловою системою (IPFS),

розподіленою системою для зберігання та доступу до файлів, веб-сайтів, програм та даних[7].

IPFS може зберігатися на веб-сторінках і у всіх файлах, які може зберігати ваш комп'ютер, таких як документи, електронні листи або записи в базі даних.

### 1.2 Аналіз актуальності розробки

Програмна реалізація шифрованого каналу передавання стеганографічних ключів є важливим кроком у забезпеченні безпеки інформації у закритих системах. Використання стеганографії та шифрування дозволяє значно підвищити рівень захисту конфіденційних даних. Проте, ефективна реалізація такої системи вимагає ретельного врахування технічних, юридичних та етичних аспектів. Подальші дослідження та розробки у цій сфері можуть сприяти підвищенню безпеки та ефективності систем захисту інформації [8].

Передача NFT зображень є складною та багатогранною предметною областю, яка поєднує в собі технологічні, соціальні та економічні аспекти. Цей феномен відкриває нові можливості для цифрових художників, але також викликає численні питання та виклики, які потребують подальшого дослідження та регулювання. Технологія NFT продовжує розвиватися, і її вплив на різні сфери життя лише збільшується.

Актуальність розробки включає такі аспекти як:

- безпека системи;
- продуктивність;
- юридичні та етичні аспекти.

Незважаючи на високий рівень захисту, існують ризики, пов'язані з використанням стеганографії та шифрування. Одним з основних викликів є виявлення стеганографічних повідомлень, що може призвести до розкриття прихованої інформації.

Шифрування та стеганографія можуть вимагати значних обчислювальних ресурсів, особливо при обробці великих обсягів даних. Оптимізація алгоритмів та

використання апаратних прискорювачів може покращити продуктивність системи.

Використання стеганографії та шифрування може мати юридичні та етичні наслідки, оскільки ці технології можуть використовуватися як для законних, так і для незаконних цілей [9].

У 2020 році багато компаній додають біткойн як платіжну систему за послуги. Наприклад, компанія Ілона Маска оголосила, що Тесла продасть автомобіль за біткойни. Цифрові активи стали настільки важливими, що навіть світові лідери, такі як транснаціональні фінансові компанії Mastercard і Visa, оголосили про співпрацю з криптовалютами. Зараз вони готуються до майбутнього цифрових платежів, пропонованих біткойнами або Ethereum. Вони хочуть допомогти цим концепціям розвиватися і розкрити свій потенціал. Якщо ми говоримо про NFT, тут застосовується дещо інша концепція. NFT (незмінний маркер) – це незмінний маркер, який є цифровим маркером, який є типом криптовалюти, такою як біткойн або Ефіріум [10].

Однак, на відміну від стандартних монет у блокчейні біткойнів, NFT унікальний і не може бути замінений подібними токенами. Існують взаємозамінні токени і незмінні токени.

Оскільки кожен NFT несе унікальну інформацію (метадані), незамінним екземпляром маркера є лише один NFT. Картину Леонарда да Вінчі "Джоконда" теж можна вважати чимось незамінним. Ви можете створювати нескінченну кількість копій зображення, але тільки оригінал має велику цінність. З часом традиційне мистецтво перейшло в Цифрове.

Спочатку різні твори мистецтва продавалися на вулиці, в майстернях і магазинах. Професійні художники, особливо любителі, розуміють, що на такому рівні їх недостатньо виховують, щоб робити те, що вони хочуть, замість того, щоб відволікатися на маркетинг. Художники, співаки, музиканти та творчі люди почали викладати свої роботи в інтернет. Щоб заробити на цьому гроші, було створено багато цифрових платформ, на які ви можете підписатися, щоб переглянути унікальний вміст [11].

Таким чином, художник може монетизувати результати своєї роботи. Так було раніше, і цей період підійшов до кінця. Технологія не стоїть на місці, і тепер є платформа для продажу NFT (рис. 1.1).

Рисунок 1.1 – NFT зображення

Художник відвідує такі Сайти, підключає електронний гаманець, завантажує свою фотографію, вводить ціну в криптовалюту (наприклад, 10-е місце в блокчейні Ethereum) і виплачує комісію платформі. Після цих простих кроків кожен може придбати цю роботу. Це не означає, що на ринку існує лише 1 цифрова версія зображення nft.

Подібно до того, як оригінальні художні принти виготовляються, використовуються, купуються та продаються, копія nft все ще є дійсною частиною блокчейну, але не має такої ж цінності, як оригінал. Крім того, не думайте, що ви зламали свою систему, клацнувши правою кнопкою миші на зображенні NFT і зберігаючи його. Це не робить вас мільйонером, оскільки завантажений файл не містить жодної інформації, яка була б частиною блокчейну Ethereum [12].

Важливо розуміти, що коли хтось купує NFT, пов'язаний з бізнесом, він не купує основні права інтелектуальної власності, пов'язані з цим бізнесом. Власник авторських прав має певні ексклюзивні права на відтворення твору, створення, виконання, демонстрацію та розповсюдження похідних творів. Як правило,

придбання твору мистецтва не передає покупцеві всі авторські права на такі твори.

Наприклад, коли хтось купує картину в художній галереї для свого будинку, він отримує найбільш фізичну картину, яку може виставити, але не всі такі картини, другорядні роботи тощо можуть бути виставлені на загальний огляд. У них немає репродукцій. Насправді основне авторське право передається лише в тому випадку, якщо власник авторських прав письмово доводить, що він має намір передати ці права разом із копією твору [13].

Якщо власник nft не отримає явного дозволу від продавця, власник NFT автоматично не отримає законного права фотографувати творчі роботи, прикріплені до nft, і створювати футболки та листівки для продажу. За відсутності інших документів покупець NFT в результаті цієї покупки має ліцензію на розміщення відповідних носіїв у своєму маркетинговому гаманці виключно в особистих цілях, але не гарантує, що nft є основним авторським правом на відповідний контент або що ці носії можуть відображатися в продуктах сторонніх виробників, на веб-сайтах або платформи.

Якщо хтось купує NFT, пов'язаний з творчою роботою, сторони можуть і в ідеалі повинні укласти контракт. Ліцензія NFT відокремлює фактичний маркер NFT від" мистецтва " (основного зображення, музики, звуку або їх поєднання), пов'язаного з NFT. Ліцензія описує, як покупці NFT отримують особисту ліцензію на використання та демонстрацію творів мистецтва, пов'язаних з nft, та комерційну ліцензію на виробництво товарів, що відображають мистецтво, пов'язане з nft, із загальним лімітом доходу 100 000 фунтів стерлінгів на рік. Насправді Ліцензія NFT-це запропонована угода, яка може бути використана лише в тому випадку, якщо покупець і продавець вважають за потрібне. Наприклад, коли платформа YellowHeart оголосила про продаж свого нового альбому Kings Of Leon як nft, умови обслуговування платформи YellowHeart включали деякі елементи ліцензії NFT, але відбулися значні зміни. Відповідно до умов використання Yellowheart, придбання nft не включає елементи мистецтва (визначені як обкладинки альбомів та зображення), пов'язані з nft, та комерційні ліцензії Nft, оскільки покупець володіє

nft і призначений лише для особистих цілей, а умови Yellowheart включають ряд статей, що обмежують комерційну діяльність, пов'язану з nft. за творчість, що стоїть за nft. Це гарна ідея [14].

Наприклад, він забороняє використання пов'язаних творів мистецтва або товарів, що містяться в них, у продуктах третіх осіб, фільмах чи інших засобах масової інформації. Багато учасників ринку стверджують, що nft можна використовувати для перевірки автентичності.

Фактично, Nft може підтвердити право власності на токен і унікальну історію того, як ці токени були розроблені і пов'язані з творчістю work.In у загальнодоступному блокчейні кожен може бачити адресу гаманця власника та пов'язані з ним метадані.Інформація доступна у вигляді загальнодоступного запису. Однак простий NFT сам по собі не може зіставити творця або власника NFT з реальною людиною у фізичному світі і не може підтвердити, що творець nft має основне право пов'язувати цей nft з конкретною творчою роботою. Підробка-це проблема творів мистецтва NFT. Легко змусити колекціонерів замислитися, чи справді якесь Цифрове мистецтво, на яке вони витратили тисячі доларів, є справжнім твором Бенксі чи просто підробкою [15].

Щоб вирішити цю проблему, деякі платформи використовують старомодний метод автентифікації творця вмісту на платформі. Наприклад, SuperRare дозволяє художникам, які бажають виставити свої цифрові роботи на продаж, спочатку переглядати такі імена, електронні листи, добірки творів мистецтва на своїй платформі, і, таким чином, SuperRare дозволяє колекціонерам отримувати справжні твори мистецтва від шанованих художників з відповідними правами на основні твори мистецтва. Інші ринки NFT намагаються обійти проблему шахрайства, використовуючи широкі обмеження. Наприклад, якщо користувач намагається щось придбати через atomicassets marketplace, відображається таке повідомлення, і користувач повинен прийняти його, щоб продовжити покупку. "Будь-хто може створювати атомні ресурси NFT і може вільно вибирати атрибути, такі як імена та зображення, що містять підроблені версії існуючого nft або викрадену

інтелектуальну власність [16].

### 1.3 Аналіз існуючих аналогів

Передача NFT зображень із застосуванням стеганографії та шифрування може підвищити безпеку та конфіденційність транзакцій. Інструменти, такі як Steghide, OpenPuff, SilentEye та Crypture, можуть використовуватися для приховування ключової інформації всередині зображень, що забезпечує додатковий рівень захисту.

Steghide дозволяє приховувати інформацію у JPEG та BMP файлах, що робить його зручним для роботи з NFT зображеннями [17].

Використання:

- створення стеганографічного ключа. перед генерацією nft зображення створюється стеганографічний ключ, який буде прихований всередині зображення;
- приховування ключа. наприклад, можна використати команду steghide embed -cf image.jpg -ef key.txt -p password;
- шифрування ключа за допомогою пароля;
- передача;
- витягування ключа. одержувач використовує steghide для вилучення ключа за допомогою команди steghide extract -sf image.jpg -p password.

Переваги:

- легкість у використанні;
- підтримка поширених форматів зображень.

Недоліки:

- обмежена підтримка форматів файлів.

openpuff підтримує широкий спектр форматів файлів, що дозволяє працювати з різними типами мультимедіа.

Використання:

- створення стеганографічного ключа. підготовка файлу із стеганографічним ключем;

− приховування ключа. наприклад, через графічний інтерфейс обирається файл-носій та файл із ключем;

− шифрування: встановлюється пароль для захисту;

− передача: передайте зображення nft із прихованим ключем;

− витягування ключа: одержувач використовує openpuff для вилучення ключа з файлу-носія.

Переваги:

− підтримка багатьох форматів файлів;

− висока стійкість до атак.

Недоліки:

− складність у використанні для новачків;

silenteye має зручний графічний інтерфейс для приховування інформації у зображеннях та аудіофайлах;

− створення стеганографічного ключа: підготовка текстового файлу із ключем;

− приховування ключа: обирається файл із ключем та застосовується шифрування;

− шифрування. обирається алгоритм шифрування (наприклад, aes) ;

− передача;

− витягування ключа: одержувач використовує silenteye для вилучення ключа, відкривши зображення та вказавши пароль.

Переваги:

− легкість у використанні завдяки графічному інтерфейсу;

− вбудовані алгоритми шифрування.

Недоліки:

− обмежена кількість підтримуваних форматів файлів;

crypture - компактний інструмент командного рядка для стеганографії.

використання;

− створення стеганографічного ключа: текстовий файл із ключем;

− приховування ключа: наприклад, командою crypture -e image.png key.txt.

Шифрування:

− передача. передача зображення nft з прихованим ключем;

− витягування ключа. одержувач використовує crypture для вилучення ключа, застосовуючи команду crypture -d image.png.

Переваги:

− легкий і швидкий інструмент командного рядка;

− підтримка різних форматів.

Недоліки:

− відсутність графічного інтерфейсу.

Використання стеганографічних інструментів, таких як Steghide, OpenPuff, SilentEye та Crypture, у передачі зображень може значно підвищити рівень безпеки та конфіденційності транзакцій. Кожен з цих інструментів має свої переваги та недоліки [18].

### 1.4 Висновки та постановка задач

У першому розділі було здійснено теоретичний аналіз предметної області з метою дослідження методів та підходів у рамках заданої теми.

Описано основні поняття та терміни, що стосуються досліджуваної сфери, включаючи визначення ключових концепцій та їх взаємозв'язків. Наведено детальний опис предметної області, що дозволяє зрозуміти контекст та специфіку досліджуваної проблеми.

Проведено аналіз важливості та актуальності розробки в сучасних умовах. Показано, чому досліджувана тема є критично важливою та які проблеми вона покликана вирішити. Зазначено, як сучасні виклики та потреби впливають на необхідність даної розробки.

# 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

У даному розділі було здійснено проектування програмного додатку. Обрано найбільш ефективні та безпечні методи для передачі та захисту даних у програмному додатку. Проведено аналіз технологій і протоколів з метою визначення оптимальних рішень для забезпечення цілісності та конфіденційності інформації. Створено архітектуру сервісу, яка враховує всі необхідні компоненти та їх взаємодію. Описано структуру системи, визначено ключові модулі та їх функції, що забезпечує масштабованість і надійність роботи програмного додатку. Розглянуто та обрано алгоритми шифрування, які будуть використані для захисту даних. Проаналізовано різні криптографічні методи з точки зору їхньої безпеки та ефективності, що дозволило вибрати найкраще рішення для розробки модуля шифрування.

## 2.1 Обґрунтування методів передачі і захисту даних

Curve25519 є однією з найпопулярніших еліптичних кривих, яка забезпечує високу безпеку та ефективність. Порівняно з іншими методами шифрування, вона має кілька ключових переваг. Наприклад, RSA, один з найпоширеніших традиційних алгоритмів шифрування, вимагає набагато більших ключів для досягнення подібного рівня безпеки, що призводить до значного збільшення обсягу обчислень і пам'яті. У той час як для досягнення безпеки, еквівалентної 128-бітному симетричному шифруванню, RSA потребує ключів довжиною близько 3072 біт, Curve25519 забезпечує це за допомогою лише 256-бітових ключів. Це робить Curve25519 значно швидшою та менш вимогливою до ресурсів [19].

Порівняно з іншими еліптичними кривими, такими як P-256, Curve25519 також має свої переваги. Вона була спеціально спроектована для уникнення певних видів атак і має спрощену реалізацію, що знижує ризики помилок у коді та вразливостей. Її параметри обрані так, щоб мінімізувати ризик сторонніх каналів атаки, що підвищує загальну безпеку. Крім того, вона є більш ефективною в

обчисленнях, що робить її привабливою для використання в мобільних пристроях і інших обмежених середовищах.

Іншим популярним методом є ECDSA (Elliptic Curve Digital Signature Algorithm), який також базується на еліптичних кривих і використовується для цифрових підписів. Хоча ECDSA забезпечує високий рівень безпеки, його реалізація може бути складнішою і менш оптимізованою порівняно з Curve25519. Це може призвести до більшої кількості помилок та вразливостей в практичних реалізаціях [20].

Curve25519 була створена Деніелом Бернштейном у 2005 році і призначена для забезпечення високої безпеки та продуктивності. Ось основні аспекти цього методу шифрування:

− еліптична крива. curve25519 є конкретною еліптичною кривою над полем простих чисел модулю 2255 − 192^{255} – 192255 − 19. це поле обрано за його криптографічну безпеку та ефективність обчислень;

− прості арифметичні операції. завдяки обраному модулю 2255 − 192^{255} – 192255 − 19, арифметичні операції над цією кривою виконуються дуже ефективно. це робить curve25519 особливо придатною для обмежених ресурсами пристроїв, таких як мобільні телефони або iot пристрої;

− висока безпека. крива забезпечує рівень безпеки, еквівалентний 128-бітовому симетричному шифруванню. це означає, що для злому curve25519 потрібно приблизно 21282^{128}2128 операцій, що є практично недосяжним для сучасних технологій;

− використання в протоколах. curve25519 широко використовується в різних криптографічних протоколах, таких як tls (transport layer security), signal protocol, та інших. вона підтримується багатьма бібліотеками, включаючи openssl, boringssl, та libsodium [21].

Робота методу:

1. Генерація ключів:

− приватний ключ. генерується випадкове число довжиною 256 біт. для забезпечення додаткової безпеки і коректності, зазвичай застосовуються деякі коригування (наприклад, встановлення певних бітів у визначені значення) ;

− публічний ключ. обчислюється шляхом множення фіксованої точки генератора кривої на приватний ключ.

2. Встановлення спільного секрету:

- два учасники, кожен з яких має свою пару приватний/публічний ключ, можуть обчислити спільний секрет. кожен учасник множить свій приватний ключ на публічний ключ іншого учасника;
- результат буде однаковим для обох учасників, утворюючи спільний секрет, який можна використовувати для шифрування подальшого обміну даними.

приклад використання:

1. Аліса генерує свої ключі:

- приватний ключ: aaa;
- публічний ключ: a=a·ga = a \cdot ga=a·g, де ggg — фіксована точка на кривій.

2. Боб генерує свої ключі:

- приватний ключ: bbb;
- публічний ключ: b=b·gb = b \cdot gb=b·g.

3. Встановлення спільного секрету:

- аліса обчислює sa=a·bs_a = a \cdot bsa=a·b;
- боб обчислює sb=b·as_b = b \cdot asb=b·a.

Оскільки SAS_ASA і SBS_BSB є однаковими (за властивостями еліптичних кривих), вони стають спільним секретом.

Curve25519 забезпечує швидкі та безпечні обчислення, роблячи її ідеальним вибором для сучасної криптографії.

Метод вирізняється своєю високою безпекою, ефективністю і простотою реалізації, що робить її привабливим вибором у порівнянні з традиційними

методами, такими як RSA, та іншими еліптичними кривими, такими як P-256 і ECDSA [15].

ChaCha20 – це сучасний поточний шифр, розроблений Деніелом Бернштейном у 2008 році як поліпшення його попереднього шифру Salsa20. Цей алгоритм призначений для швидкого і безпечного шифрування даних. Він має кілька важливих характеристик, які роблять його привабливим для використання в сучасних криптографічних системах.

ChaCha20 використовує 256-бітовий ключ і 64-бітовий початковий вектор (nonce), що дозволяє йому забезпечити високий рівень безпеки. Шифр працює шляхом генерування псевдовипадкової послідовності байтів (потоку), яка потім використовується для шифрування або дешифрування даних шляхом застосування операції XOR з відкритим текстом або шифрованим текстом.

Один з ключових аспектів ChaCha20 полягає у використанні чотириблочного структури, де кожен блок обробляється серією арифметичних операцій, циклічних зсувів і побітових операцій XOR. Це робить шифр швидким та ефективним для програмного забезпечення, оскільки ці операції легко реалізуються на більшості сучасних процесорів.

ChaCha20 має кілька переваг перед іншими поточними шифрами, такими як RC4, який вважається менш безпечним через численні вразливості. Порівняно з AES (Advanced Encryption Standard) в режимі потоку, ChaCha20 має перевагу у швидкості на системах без апаратної підтримки AES, таких як деякі мобільні пристрої. AES у режимі потоку зазвичай використовує режим лічильника (CTR), який також забезпечує високу безпеку, але може бути повільнішим без апаратного прискорення [12].

Іншим важливим аспектом є стійкість ChaCha20 до атак через сторонні канали. Завдяки своїй конструкції, яка мінімізує залежність часу виконання від вхідних даних, ChaCha20 є більш стійким до атак через аналіз часу або споживаної енергії, що є важливим для забезпечення безпеки в реальних умовах.

ChaCha20 часто використовується в поєднанні з алгоритмом аутентифікації Poly1305, що утворює комбінований режим шифрування і аутентифікації, відомий як ChaCha20-Poly1305. Цей режим забезпечує одночасно конфіденційність і цілісність даних, що робить його ідеальним для використання в сучасних мережевих протоколах, таких як TLS (Transport Layer Security).

Список переваг і недоліків шифру ChaCha20.

Переваги:

− висока швидкість. chacha20 є дуже швидким шифром, особливо на пристроях без апаратного прискорення aes;

− ефективність реалізації. чотириблочна структура і прості арифметичні операції роблять chacha20 легким для реалізації на різних платформах;

− висока безпека. використання 256-бітового ключа забезпечує високий рівень безпеки шифрування;

− стійкість до атак через сторонні канали. мінімізація залежності часу виконання від вхідних даних робить chacha20 стійким до атак через аналіз часу або споживаної енергії;

− простота розуміння і використання. чистий та простий дизайн робить chacha20 привабливим для використання у відкритих інтерфейсах програмування.

Недоліки:

− відсутність широкого прийняття. хоча chacha20 стає все більш популярним, деякі старіші системи можуть підтримувати тільки стандартні методи шифрування, такі як aes;

− відносно молодий алгоритм. оскільки chacha20 був розроблений лише у 2008 році, він може не мати такого ж рівня перевірки часом, як деякі інші стандартні методи шифрування;

− потреба у безпечному обміні ключами. як і в усіх сучасних системах шифрування, безпечність chacha20 залежить від безпечного обміну ключами між вузлами;

− відносно низька стійкість до атаки підбору ключа. хоча 256-бітовий ключ забезпечує велику стійкість, низька ймовірність атаки на підбор ключа все ж може залишати певний ризик [23].

Переваги ChaCha20 переважають його недоліки, зокрема швидкість, ефективність та висока безпека, що робить його привабливим вибором для багатьох сучасних криптографічних застосувань.

Загалом, ChaCha20 – це швидкий, безпечний і ефективний поточний шифр, який знаходить широке застосування в різних криптографічних системах завдяки своїм перевагам перед іншими методами [24].

## 2.2 Розроблення архітектури сервісу

Розробку архітектури сервісу можна почати з блок-схем з кількох причин:

− визначення логіки процесу;
− візуалізація алгоритму;
− виявлення проблем;
− спільна робота;
− документація.

Блок-схема допомагає визначити послідовність дій, необхідних для досягнення поставленої мети. Вона визначає, які кроки потрібно виконати і в якому порядку.

Блок-схема дає змогу візуалізувати алгоритм роботи програми. Вона допомагає розробникам і всім зацікавленим сторонам краще зрозуміти, як працює програма і які кроки вона виконує [25].

Під час розробки блок-схема може допомогти виявити потенційні проблеми або недоліки у логіці програми до того, як вони виникнуть на рівні коду. Вона є чудовим інструментом для спільної роботи між розробниками та зацікавленими сторонами. Вона може допомогти узгодити розуміння процесу між усіма учасниками проекту.

Блок-схема служить як документація для програмного продукту. Вона може

бути використана для навчання нових розробників, а також для аналізу та відстеження змін у процесі розробки.

Початок розробки з блок-схем допомагає структурувати і уточнити концепцію програми, зменшує кількість помилок на пізніших етапах розробки і полегшує спільну роботу над проектом [16].

Звичайна блок-схема складається з різних геометричних фігур, таких як прямокутники (або квадрати), ромби, стрілки та текстові блоки. Кожна фігура має своє визначення та призначення:

− прямокутники використовуються для позначення конкретних дій або кроків у процесі. Кожен прямокутник містить текст, який описує дію.

− ромби використовуються для позначення рішень або умов у програмі. Приклади: "чи дані вірні?", "чи задовільняє умова?". Умови, які можуть бути визначені у ромбах, зазвичай мають два можливих варіанти відповіді: "так" і "ні";

− стрілки вказують напрямок потоку у програмі. Вони з'єднують фігури і показують порядок виконання дій. Наприклад: стрілка, що виходить з одного блоку і вказує на наступний, показує послідовність виконання дій;

− текстові блоки використовуються для підпису фігур, опису їх функцій або пояснення логіки процесу. Наприклад: "початок", "кінець", "пояснення".

Спочатку визначається логіка процесу, який ви хочете зобразити у блок-схемі. Це може бути послідовність кроків або умови прийняття рішень. Для кожного кроку або умови вибирається відповідну фігуру: прямокутник для дії, ромб для умови. Додається текст до кожної фігури, щоб пояснити, що саме відбувається у кожному кроці або умові. Підключення стрілок відповідно до послідовності виконання кроків. Якщо потрібно, додаються текстові блоки для пояснення функцій або логіки процесу.

Далі може розглянути блок схему алгоритму роботи системи (рис. 2.1).

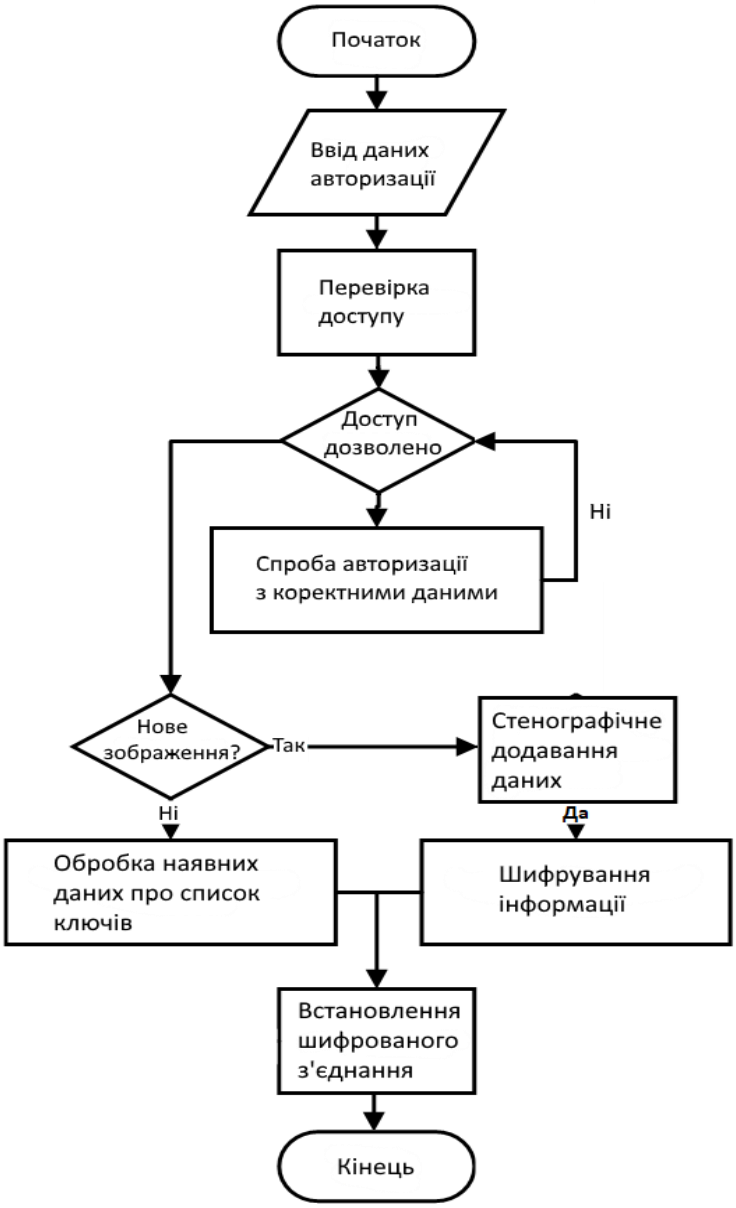



Рисунок 2.1 – Блок схема роботи системи

Крок 1. Початок.

Процес починається з початкового блоку, що позначає початок алгоритму.

Крок 2. Ввід даних авторизації.

Користувач вводить свої дані для авторизації. Це може включати ім'я користувача, пароль або інші необхідні облікові дані.

Крок 3. Перевірка доступу.

Система перевіряє введені дані для визначення доступу. Це може включати перевірку наявності користувача в базі даних, правильності пароля та інших критеріїв.

Крок 4. Доступ дозволено?

Умовний блок перевіряє, чи відповідають введені дані критеріям доступу:

– Якщо доступ дозволено, алгоритм продовжується.

– Якщо доступ не дозволено, користувач отримує відмову у доступі, і процес повертається до кроку 2 для повторного введення даних.

Крок 5. Спроба авторизації з коректними даними.

Система намагається авторизувати користувача, використовуючи коректні дані.

Крок 6. Нове з'єднання?

Умовний блок перевіряє, чи потрібно створити нове з'єднання:

– Якщо нове з'єднання не потрібне, алгоритм переходить до обробки наявних даних ключів.

– Якщо нове з'єднання потрібне, процес переходить до наступного кроку.

Крок 7. Оброблення наявних даних ключів.

Система обробляє наявні дані ключів, що можуть включати перевірку, оновлення або інші операції з ключами.

Крок 8. Встановлення з'єднання.

Система встановлює з'єднання, необхідне для подальшої роботи користувача.

Крок 9. Кінець.

Процес завершується, коли всі необхідні дії виконано, і користувач отримує доступ до системи або отримує відповідний результат.

Основна мета Use Case діаграми полягає в ідентифікації функціональних вимог до системи та визначенні, як користувачі будуть взаємодіяти з системою для досягнення своїх цілей. Вона допомагає команді розробників розуміти, як система повинна працювати з точки зору користувачів і які вимоги до неї повинні бути

враховані під час розробки [27].

Use Case діаграма корисна для розробки з кількох причин (рис. 2.2):

- уточнення вимог;
- визначення функціональності;
- забезпечення повноти;
- визначення границь системи;
- орієнтація на користувача.

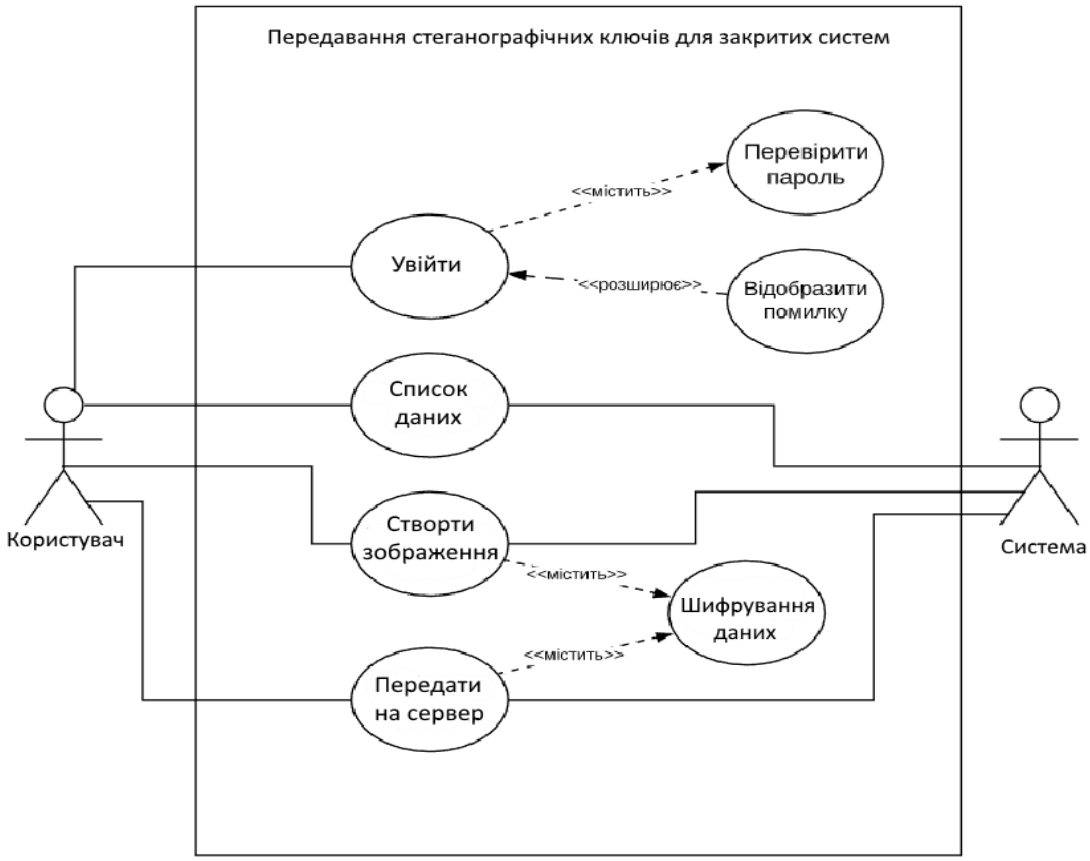



Рисунок 2.2 – Use Case діаграма

Вона допомагає команді розробників краще зрозуміти, як користувачі будуть взаємодіяти з системою та які функціональні вимоги до системи існують. Варіанти використання допомагають ідентифікувати конкретні сценарії взаємодії між користувачами та системою, що потрібно реалізувати. Вона допомагає визначити всі можливі способи взаємодії користувачів з системою, щоб не пропустити жодної

важливої функціональності, дозволяє чітко визначити, хто є акторами, що взаємодіють з системою, і які саме функції системи надаються.

Також допомагає розробникам більше зосередитися на потребах користувачів та зрозуміти, як систему слід розробляти, щоб забезпечити оптимальний досвід користувача.

Може бути використана для спілкування з зацікавленими сторонами, такими як клієнти або менеджмент, для показу того, як система буде працювати та які функціональність буде включена.

Use Case діаграма є важливим інструментом для розробки, оскільки вона допомагає зрозуміти, як система повинна функціонувати з точки зору користувачів та визначити, які функції системи повинні бути реалізовані.

Також важлтвою частиною проектування архітектури є вибір структури коду. Основними компонентами програмної реалізації шифрованого каналу передавання стеганографічних ключів є:

− генератор ключів створює стеганографічні ключі для приховування інформації;

− модуль шифрування забезпечує шифрування ключів перед передаванням;

− стеганографічний модуль приховує зашифровані ключі у цифрових контейнерах;

− комунікаційний модуль здійснює передавання цифрових контейнерів між користувачами.

Принцип роботи системи:

− генерація та шифрування ключів. Генератор ключів створює стеганографічні ключі, які потім шифруються модулем шифрування, використовуючи обраний алгоритм;

− стеганографічне вбудовування. Зашифровані ключі вбудовуються в цифрові контейнери за допомогою стеганографічного модуля;

− передавання контейнерів. Комунікаційний модуль передає цифрові контейнери між користувачами через захищений канал.

Розроблення цих методів буде відбуватися на основі шаблону проектування MVC. Це архітектурний шаблон, що використовується в розробці програмного забезпечення для розділення компонентів програми на три основні частини: модель, вид та контролер.

Модель (Model) представляє дані та бізнес-логіку програми. Модель відповідає за доступ до даних, їх обробку та збереження. Вона не повинна залежати від вигляду або контролера, що дозволяє їй бути переносимою та повторно використовуваною.

Вид (View) представляє інтерфейс користувача програми. Він відображає дані, які надає модель, та реагує на дії користувача. Вид не має знати про бізнес-логіку, але може співпрацювати з контролером для обробки подій користувача.

Контролер (Controller) обробляє взаємодію між користувачем, моделлю та видом. Він приймає введення від користувача, виконує необхідні дії (наприклад, вибірка даних з моделі або оновлення її стану) та відповідає за відображення відповідного вигляду користувачу. Контролер також не має залежати від конкретного вигляду, що дозволяє легко змінювати вигляди без зміни логіки програми [28].

Якщо порівняти MVC з іншими можливими архітектурами, такими як MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel) можна побачити його переваги і простоту використання (рис. 2.3).

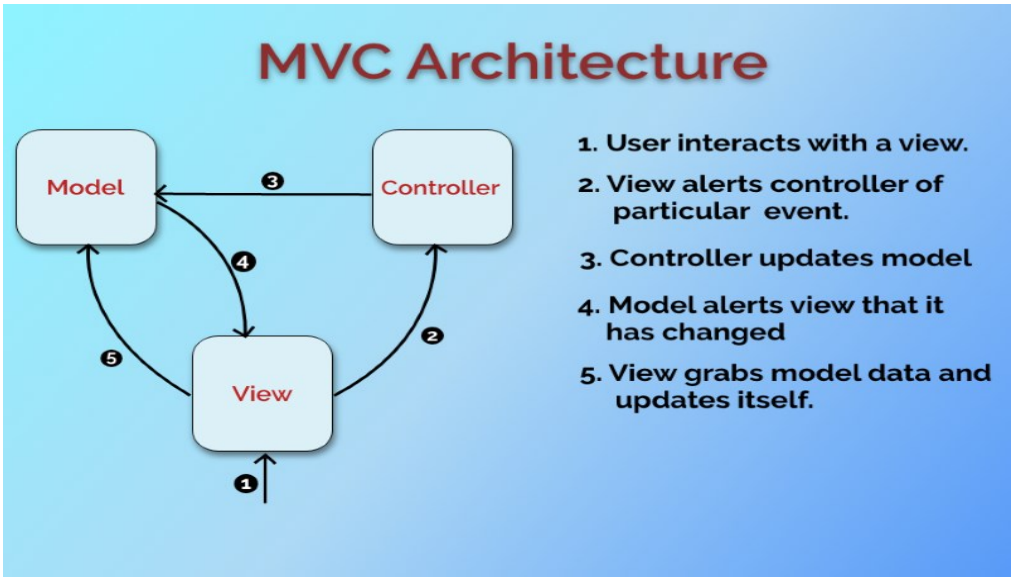



Рисунок 2.3 – MVC архітектура

MVP (Model-View-Presenter):

У MVP присутня схожа ідея розділення на компоненти: модель, вигляд та презентер.

Основна відмінність полягає у тому, що презентер виконує більш активну роль у взаємодії з виглядом, контролюючи його поведінку та оновлення.

У MVC контролер є слабшим посередником між моделлю та видом, в той час як у MVP презентер має більш активну роль у визначенні того, які дані потрібно відображати та як відповідати на взаємодію користувача [20].

MVVM (Model-View-ViewModel):

У MVVM також присутня ідея розділення на компоненти: модель, вид та ViewModel (рис. 2.4).

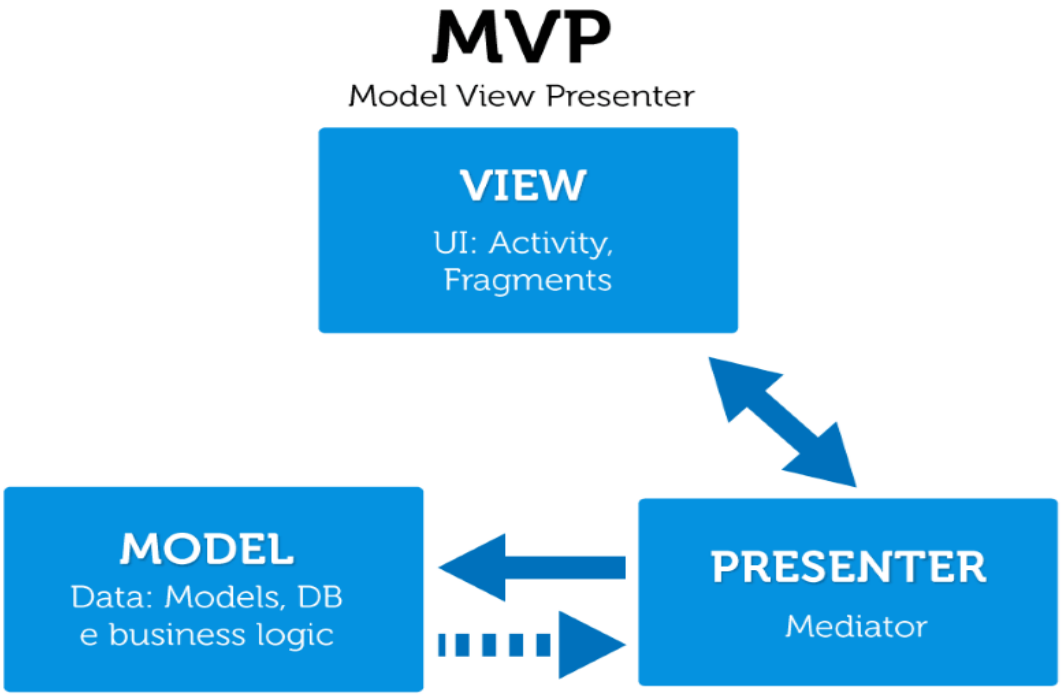



Рисунок 2.4 – MVP архітектура

ViewModel подібний до презентера в MVP, але має додаткові можливості, такі як прив'язка даних. Він відповідає за преобразування даних моделі в формат, який може бути легко відображений у вигляді.

У порівнянні з MVC, MVVM дозволяє зберігати більше логіки в ViewModel, що спрощує відображення даних та реагування на зміни (рис. 2.5).

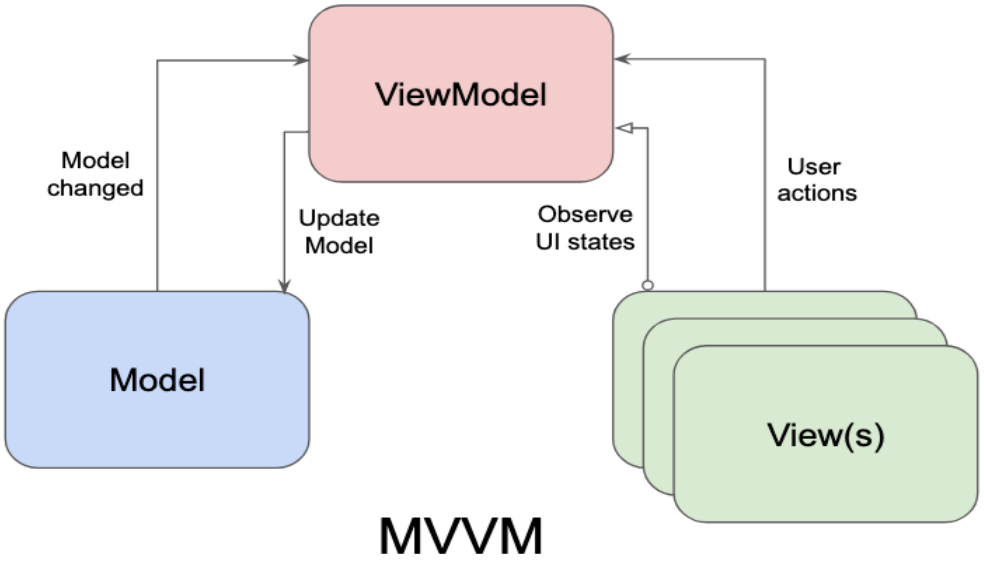



Рисунок 2.5 – MVVM архітектура

Схема, ілюструє архітектурний патерн Model-View-ViewModel, або MVVM. Цей патерн використовується в розробці програмного забезпечення, особливо для створення користувацьких інтерфейсів.

Основна мета – розділити розробку графічного інтерфейсу користувача і бізнес-логіку програми, що дозволяє створювати більш структурований і підтримуваний код.

Патерн MVVM складається з трьох основних компонентів: Model (Модель), View (Подання) та ViewModel (Модель подання). Модель представляє дані та бізнес-логіку програми. Вона відповідає за отримання, зберігання та обробку даних, а також за взаємодію з базою даних чи іншими джерелами даних. Модель не знає нічого про інтерфейс користувача і не взаємодіє з ним безпосередньо.

Подання, або View, представляє собою користувацький інтерфейс. Воно відображає дані та отримує користувацькі дії, такі як натискання кнопок чи введення тексту. View безпосередньо пов’язане з користувачем і відповідає за візуальне представлення інформації. Однак, воно не містить логіки для обробки даних.

Модель подання, служить посередником між моделлю та поданням. Вона отримує дані від моделі та підготовлює їх для відображення у Поданні. Вона також обробляє дії користувача, які надходять з подання, та оновлює Модель відповідно. У цьому патерні модель подання спостерігає за станом інтерфейсу користувача і, у разі змін у моделі, оновлює подання.

На схемі зображено, як модель змінюється, і ці зміни передаються до моделі подання, яка потім оновлює модель відповідно. З іншого боку, дії користувача, що відбуваються у поданні, також передаються до моделі подання, яка спостерігає за станом інтерфейсу користувача і реагує на ці дії.

Таким чином, патерн MVVM забезпечує чітке розділення відповідальностей між компонентами програми, що робить її більш гнучкою, масштабованою та легкою для тестування. Кожен компонент може бути розроблений та протестований окремо, що значно спрощує процес розробки та обслуговування програми.

### 2.3 Проєктування бази даних, захист даних та моделювання алгоритмів роботи програмного засобу

Питання безпеки передачі інформації є надзвичайно актуальним. Зростання обсягів даних, що передаються через різноманітні мережі, вимагає розробки надійних методів захисту інформації від несанкціонованого доступу та втручання. Один із таких методів – це шифрування, яке дозволяє захистити дані шляхом їх перетворення в недоступний для розуміння вигляд. Однак, навіть зашифровані дані можуть привертати увагу потенційних зловмисників. Тому, щоб зробити передачу інформації ще більш безпечною, використовуються методи стеганографії, що дозволяють приховувати сам факт передачі секретних даних.

Модуль шифрування покликаний забезпечити конфіденційність, цілісність та автентичність переданих даних. У рамках цієї розробки розглянуто використання сучасних криптографічних алгоритмів, таких як Curve25519 для обміну ключами та ChaCha20 для симетричного шифрування [29].

Розроблення модуля окреслює основні аспекти безпеки передачі інформації, наголошує на важливості використання криптографічних та стеганографічних методів, а також встановлює загальний контекст розробки модуля шифрування. У цьому алгоритмі описано теоретичні та аспекти реалізації цього модуля, включаючи генерацію ключів, обмін ними, шифрування та дешифрування даних, а також методи стеганографії для прихованого вбудовування ключів у зображення.

Модуль шифрування забезпечує безпечну передачу даних між двома сторонами, використовуючи сучасні криптографічні алгоритми. В контексті передачі стеганографічних ключів через зображення, модуль шифрування складається з наступних компонентів:

1. Генерація ключів

− кожна сторона генерує пару ключів (приватний та публічний) за допомогою асиметричного алгоритму curve25519;

− приватний ключ зберігається в безпечному місці і не передається, тоді як публічний ключ передається іншій стороні.

2. Обмін публічними ключами

− Обидві сторони обмінюються своїми публічними ключами через безпечний канал зв'язку або попередньо узгоджений спосіб.

3. Генерація спільного секрету

− кожна сторона використовує свій приватний ключ та публічний ключ іншої сторони для генерування спільного секрету за допомогою алгоритму diffie-hellman;

− спільний секрет використовується як основа для шифрування та дешифрування даних.

4. Вбудовування секрету в зображення (Стеганографія)

− спільний секрет вбудовується у зображення за допомогою стеганографічного методу, наприклад, lsb (least significant bit);

− зображення модифікується таким чином, що зміни не помітні для людського ока, але можуть бути відновлені програмно.

5. Шифрування зображення

− зображення, в яке вбудовано секрет, шифрується за допомогою симетричного алгоритму chacha20;

− chacha20 забезпечує швидке та безпечне шифрування.

6. Передача зашифрованого зображення

− зашифроване зображення передається одержувачу;

− одержувач отримує зашифроване зображення.

Етапи теоретичної розробки

Вибір криптографічних примітивів:

− асиметричний алгоритм: curve25519 для обміну ключами;

− симетричний алгоритм: chacha20-poly1305 для шифрування даних;

− стеганографічний метод: lsb для вбудовування секрету в зображення.

Аналіз безпеки:

− переконатися, що використані алгоритми є стійкими до відомих атак;

− оцінити безпеку каналу передачі публічних ключів;

− переконатися, що вбудовування секрету в зображення не порушує його цілісність і не призводить до виявлення сторонніми.

Розроблення алгоритму:

− створити чітку послідовність кроків для кожного етапу модуля шифрування;

− забезпечити, що всі криптографічні операції виконуються коректно і без помилок.

Реалізація:

− реалізувати модуль шифрування в обраній мові програмування;

− переконатися, що дані коректно шифруються і дешифруються, а секрет правильно вбудовується і витягується з зображення.

Документація:

− описати всі функції та алгоритми, що використовуються в модулі;

− забезпечити наявність інструкцій для користувачів щодо використання модуля.

Модуль шифрування для передачі стеганографічних ключів у зображеннях забезпечує високий рівень безпеки завдяки використанню сучасних криптографічних алгоритмів та методів стеганографії. Теоретична розроблення модуля включає вибір криптографічних примітивів, аналіз безпеки, розробку алгоритму, реалізацію і тестування, а також документацію [30].

## 2.4 Висновки до розділу 2

У цьому розділі було здійснено проектування програмного додатку.

Обрано найбільш ефективні та безпечні методи для передачі та захисту даних у програмному додатку. Проведено аналіз технологій і протоколів, з метою визначення оптимальних рішень для забезпечення цілісності та конфіденційності інформації.

Створено архітектуру сервісу, яка враховує всі необхідні компоненти та їх взаємодію. Описано структуру системи, визначено ключові модулі та їх функції, що забезпечує масштабованість і надійність роботи програмного додатку.

Розглянуто та обрано алгоритми шифрування, які будуть використані для захисту даних. Проаналізовано різні криптографічні методи з точки зору їхньої безпеки та ефективності, що дозволило вибрати найкраще рішення для розробки модуля шифрування.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ШИФРОВАНОГО КАНАЛУ ПЕРЕДАВАННЯ СТЕГАНОГРАФІЧНИХ КЛЮЧІВ ДЛЯ ЗАХИСТУ СИСТЕМ

У даному розділі описано практичну реалізацію програмного додатку. Розроблено дизайн користувацького інтерфейсу, орієнтований на зручність та інтуїтивність використання, враховуючи сучасні тенденції в UX/UI дизайні. Виконано розробку та налаштування шифрованого каналу для безпечної передачі даних, реалізовано необхідні протоколи та алгоритми шифрування для захисту інформації від несанкціонованого доступу та перехоплення. Здійснено програмну реалізацію основних функцій сервісу, написано код для роботи всіх модулів та компонентів, проведено тестування та налагодження системи для забезпечення її стабільної та надійної роботи. Створено детальну інструкцію користувача, яка містить всі необхідні відомості для ефективного використання програмного додатку, описано основні функції, налаштування та можливості додатку, а також надано рекомендації щодо вирішення можливих проблем під час його експлуатації.

## 3.1 Розроблення інтерфейсу

Створення гарного інтерфейсу для веб-проектів на HTML та CSS включає дотримання певних правил і принципів. Ось ключові з них:

− семантичні теги. Використовуйте теги html5 для структурування контенту. це допоможе пошуковим системам і допоміжним технологіям краще розуміти ваш сайт;

− чистий код. Дотримуйтесь принципів чистого коду, використовуйте відступи та коментарі для полегшення читання та підтримки коду;

− гнучкі сітки (grids). Використовуйте гнучкі сітки та системи верстки, такі як flexbox або css grid, для створення адаптивного макета;

− медіа-запити. Використовуйте медіа-запити для адаптації стилів під різні розміри екранів;

– стилі. Використовуйте однакові стилі для схожих елементів. використовуйте css-класи для забезпечення консистентності;

– мінімалізм. Використовуйте мінімалістичний підхід до дизайну. уникайте перевантаження сторінки великою кількістю елементів або кольорів;

– простір. Використовуйте відступи (padding та margin) для забезпечення простору між елементами;

– альтернативний текст. Додавайте атрибути alt до зображень для покращення доступності;

– aria атрибути. Використовуйте aria-атрибути для покращення доступності динамічних елементів;

– шрифти та розмір тексту. Використовуйте різні розміри шрифтів та їх стиль для створення ієрархії;

– кольори. Використовуйте контрастні кольори для важливих елементів, щоб привернути до них увагу;

– інтерактивність. Забезпечте чіткий зворотний зв'язок для інтерактивних елементів (наприклад, кнопок). Використовуйте псевдокласи, такі як :hover, :focus та :active;

– кросбраузерне тестування. Перевіряйте інтерфейс у різних браузерах, щоб забезпечити однаковий вигляд та функціональність;

– оптимізація швидкості. Мінімізуйте файли css та html, використовуйте кешування та оптимізуйте зображення для швидкого завантаження сторінки.

Створення форми авторизації:

```
<div class="fixed-bg">
<div class="darkness-bg">
<div class="wrap enter-p">

<div class="left">
<a class="logo" href="#">BCH<span>24</span></a>

<div class="center">
<div class="url">
<p>Please check that you are<br> visiting the correct URL</p>

<div class="link-bg">
<a href="#">https://bch24.io</a>
</div>
```

```
</div>

<div class="note">
<span>Note:</span> Ad-Blockers, VPNs/Proxies, Bots/Scripts, Multiple
Accounts are NOT allowed and will result in permanent account suspension!
</div>
</div>

<img class="robot" src="/img/robot-reg.png">
</div>

<div class="right">
<div class="column">
<form action="/main/authorize" method="POST">
<span>Login</span>
<input name="login" type="text" >
<span>Password</span>
<input name="pass" type="password" >

<input class="input-btn" type="submit" value="Увійти">
</form>
</div>
</div>
</div>
</div>
</div>

<div class="wrap">
<div class="login">
<form class="login-form" action="/main/authorize" method="POST">

</form>
</div>
</div>
```

Після створення форми авторизації на HTML та CSS важливо звернути увагу на декілька ключових аспектів, щоб забезпечити її безперебійну роботу та зручність для користувачів.

Слід подбати про валідацію форми. Це важливий крок, який забезпечує коректність введених користувачами даних. На стороні клієнта можна використовувати JavaScript для перевірки правильності введення, наприклад, формату електронної пошти або мінімальної довжини паролю. Але не варто покладатися тільки на клієнтську валідацію, оскільки її можна обійти. Тому важливо реалізувати також і серверну валідацію, яка повторно перевіряє дані після їх надсилання на сервер.

Необхідно подбати про безпеку даних користувачів. Всі дані, які передаються з форми авторизації, повинні бути захищені за допомогою HTTPS, що забезпечить

шифрування інформації під час її передачі між клієнтом і сервером. Також варто звернути увагу на захист від атак типу CSRF, що можна реалізувати за допомогою токенів CSRF.

Не менш важливо забезпечити добру доступність форми авторизації. Використання семантичних тегів HTML5 і ARIA-атрибутів допоможе зробити форму зрозумілою для користувачів з обмеженими можливостями та допоміжних технологій. Наприклад, додавання атрибутів aria-label для полів вводу або role="form" для самої форми підвищить її доступність.

Ще один важливий аспект – це зворотний зв'язок для користувача. Після натискання кнопки «Увійти» користувач повинен отримати чітке повідомлення про успішну авторизацію або про помилки, які виникли під час спроби входу. Це можуть бути повідомлення про неправильний логін або пароль, або інші проблеми, які можуть виникнути.

Забезпечення адаптивного дизайну форми також є важливим. Використання медіа-запитів дозволить зробити форму зручною для користувачів на різних пристроях – від комп'ютерів до смартфонів. Адаптивна верстка з використанням Flexbox або CSS Grid допоможе створити макет, який буде гарно виглядати та функціонувати на будь-якому розмірі екрану.

І нарешті, не забувайте про тестування. Проводьте кросбраузерне тестування, щоб переконатися, що форма коректно відображається і працює в різних браузерах. Також важливо перевірити форму на різних пристроях, щоб забезпечити максимальну зручність для всіх користувачів.

Таким чином, створення форми авторизації – це не тільки написання HTML та CSS коду, але й забезпечення її функціональності, безпеки, доступності та зручності для кінцевих користувачів. Дотримуючись цих рекомендацій, ви зможете створити форму, яка буде надійною та ефективною, що забезпечить позитивний досвід користувачів вашого веб-сайту.

### 3.2 Обґрунтування головних методів класів

Розроблення шифрованого каналу передачі даних є ключовим елементом забезпечення безпеки в сучасних інформаційних системах. Основна мета шифрованого каналу – це забезпечення конфіденційності, цілісності та автентичності даних, що передаються між сторонами. Використання криптографічних алгоритмів та стеганографічних методів у цьому процесі дозволяє не лише захистити інформацію від несанкціонованого доступу, але й приховати сам факт її передачі.

Архітектура шифрованого каналу:

– кожна сторона генерує пару ключів (приватний та публічний) за допомогою асиметричного алгоритму curve25519;

– сторони обмінюються публічними ключами через безпечний канал;

– спільний секрет генерується за допомогою приватного ключа однієї сторони та публічного ключа іншої сторони, що забезпечує захищений обмін даними;

– спільний секрет вбудовується в зображення за допомогою методу стеганографії lsb (least significant bit) ;

– зображення, що містить секрет, шифрується за допомогою алгоритму chacha20 для забезпечення додаткового рівня захисту;

– зашифроване зображення передається через канал зв'язку;

– відновлення спільного секрету з зображення для подальшого використання.

Розробити шифрований канал для передачі стеганографічних ключів можна використовуючи бібліотеки, які підтримують криптографічні алгоритми Curve25519 і ChaCha20. Ми скористаємося розширеннями libsodium (рис. 3.1).

```
sudo apt-get install php-sodium
```

Рисунок 3.1 – Встановлення розширення

Далі реалізуємо основні функції для взаємодії елементів.

Генерація пари ключів Curve25519:

```
function generate_keypair() {
   $private_key = sodium_crypto_box_keypair();
   $public_key = sodium_crypto_box_publickey($private_key);
   return ['private_key' => $private_key, 'public_key' => $public_key];
}
```

Генерація спільного секрету:

```
function generate_shared_secret($private_key, $peer_public_key) {
   return sodium_crypto_scalarmult(sodium_crypto_box_secretkey($private_key), $peer_public_key);
}
```

Шифрування повідомлення за допомогою ChaCha20:

```
function encrypt_message($message, $key, $nonce) {
   return sodium_crypto_aead_chacha20poly1305_ietf_encrypt($message, '', $nonce, $key);
}
```

Дешифрування повідомлення за допомогою ChaCha20-Poly1305:

```
function decrypt_message($encrypted_message, $key, $nonce) {
   return sodium_crypto_aead_chacha20poly1305_ietf_decrypt($encrypted_message, '', $nonce, $key);
}
```

На основі описаних функцій, процес реалізації шифрованого каналу виглядає наступним чином.

Генерація ключів:

```
$a_keys = generate_keypair();
$b_keys = generate_keypair();
```

Обмін ключами:

```
$a_public_key = $a_keys['public_key'];
$b_public_key = $b_keys['public_key'];
```

Генерація спільного секрету:

```
$a_shared_secret = generate_shared_secret($a_keys['private_key'], $b_public_key);
$b_shared_secret = generate_shared_secret($b_keys['private_key'], $a_public_key);
```

Вбудовування секрету в зображення:

```
$secret = bin2hex($a_shared_secret);
embed_secret_in_image('input.png', $secret);
```

Шифрування зображення:

```
$nonce = random_bytes(SODIUM_CRYPTO_AEAD_CHACHA20POLY1305_IETF_NPUBBYTES);
$encrypted_image = encrypt_image('output.png', $a_shared_secret, $nonce);
```

Передача зашифрованого зображення.

### 3.3 Проєктування структури бази даних

Щоб передати стеганографічний ключ у зображенні, потрібно використовувати стеганографічний метод для вбудовування секретного ключа в зображення. У PHP можна використовувати прості методи стеганографії, такі як LSB для вбудовування даних у зображення.

Алгоритм:

- генерація ключів і обмін ними;
- генерація спільного секрету;
- вбудовування секрету в зображення;
- шифрування зображення;
- передача зображення;
- витягування секрету із зображення;
- дешифрування зображення.

Вбудовування секрету в зображення:

```
function embed_secret_in_image($image_path, $secret) {
  $image = imagecreatefrompng($image_path);
  $width = imagesx($image);
  $height = imagesy($image);

  $secret_bin = '';
  for ($i = 0; $i < strlen($secret); $i++) {
    $secret_bin .= str_pad(decbin(ord($secret[$i])), 8, '0', STR_PAD_LEFT);
  }

  $index = 0;
  for ($y = 0; $y < $height; $y++) {
    for ($x = 0; $x < $width; $x++) {
      $rgb = imagecolorat($image, $x, $y);
      $r = ($rgb >> 16) & 0xFF;
      $g = ($rgb >> 8) & 0xFF;
```

```
            $b = $rgb & 0xFF;

            if ($index < strlen($secret_bin)) {
                $b = ($b & 0xFE) | $secret_bin[$index];
                $index++;
            }

            $color = imagecolorallocate($image, $r, $g, $b);
            imagesetpixel($image, $x, $y, $color);
        }
    }
}
```

Шифрування зображення:

```
function encrypt_image($image_path, $key, $nonce) {
    $image_data = file_get_contents($image_path);
    return sodium_crypto_aead_chacha20poly1305_ietf_encrypt($image_data, '', $nonce, $key);
}

function decrypt_image($encrypted_image_data, $key, $nonce) {
    return sodium_crypto_aead_chacha20poly1305_ietf_decrypt($encrypted_image_data, '', $nonce, $key);
}
```

Витягування секрету із зображення:

```
function extract_secret_from_image($image_path, $secret_length) {
    $image = imagecreatefrompng($image_path);
    $width = imagesx($image);
    $height = imagesy($image);

    $secret_bin = '';
    for ($y = 0; $y < $height; $y++) {
        for ($x = 0; $x < $width; $x++) {
            $rgb = imagecolorat($image, $x, $y);
            $b = $rgb & 0xFF;

            $secret_bin .= $b & 0x01;

            if (strlen($secret_bin) >= $secret_length * 8) {
                break 2;
            }
        }
    }

    $secret = '';
    for ($i = 0; $i < strlen($secret_bin); $i += 8) {
        $secret .= chr(bindec(substr($secret_bin, $i, 8)));
    }

    imagedestroy($image);
    return $secret;
}
```

Для аналізу ефективності розробленого модуля шифрування можна використовувати діаграми, які показують продуктивність і безпеку на різних етапах

процесу. Можна порівняти час, необхідний для генерації ключів, шифрування, дешифрування та вбудовування/витягування секрету з зображення.

На діаграмі показано час (в мілісекундах), необхідний для двох важливих етапів криптографічного процесу: генерації ключів та обміну ключами. Як видно з діаграми, генерація ключів займає набагато більше часу, ніж обмін ключами. Це пояснюється тим, що генерація ключів є більш складним і ресурсомістким процесом, ніж обмін вже згенерованими ключами (рис. 3.2).

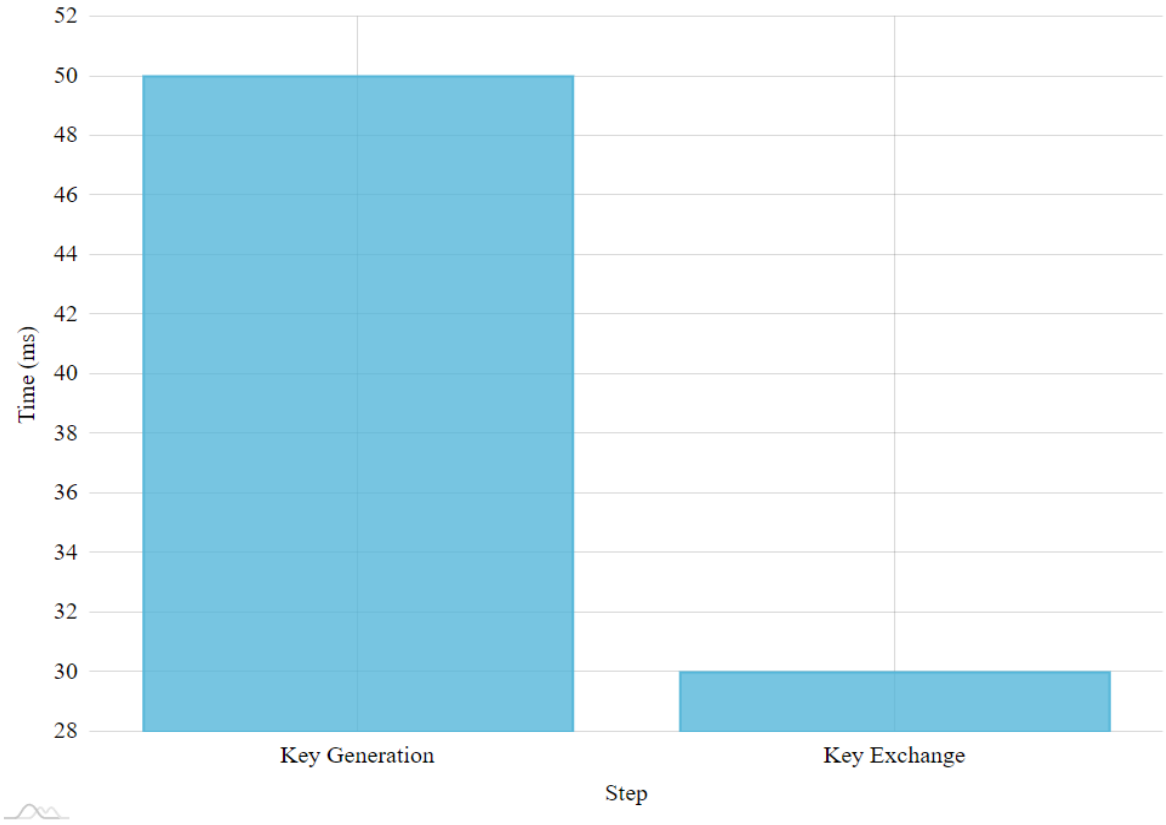



Рисунок 3.2 – Швидкість генерації і вставки

На діаграмі представлено час, необхідний для генерації спільного секрету і вбудовування цього секрету в зображення. Генерація спільного секрету займає значно менше часу, ніж процес вбудовування секрету в зображення. Це можна пояснити тим, що вбудовування секрету в зображення вимагає додаткових обчислень для забезпечення його непомітності та інтеграції в графічну інформацію (рис. 3.3).

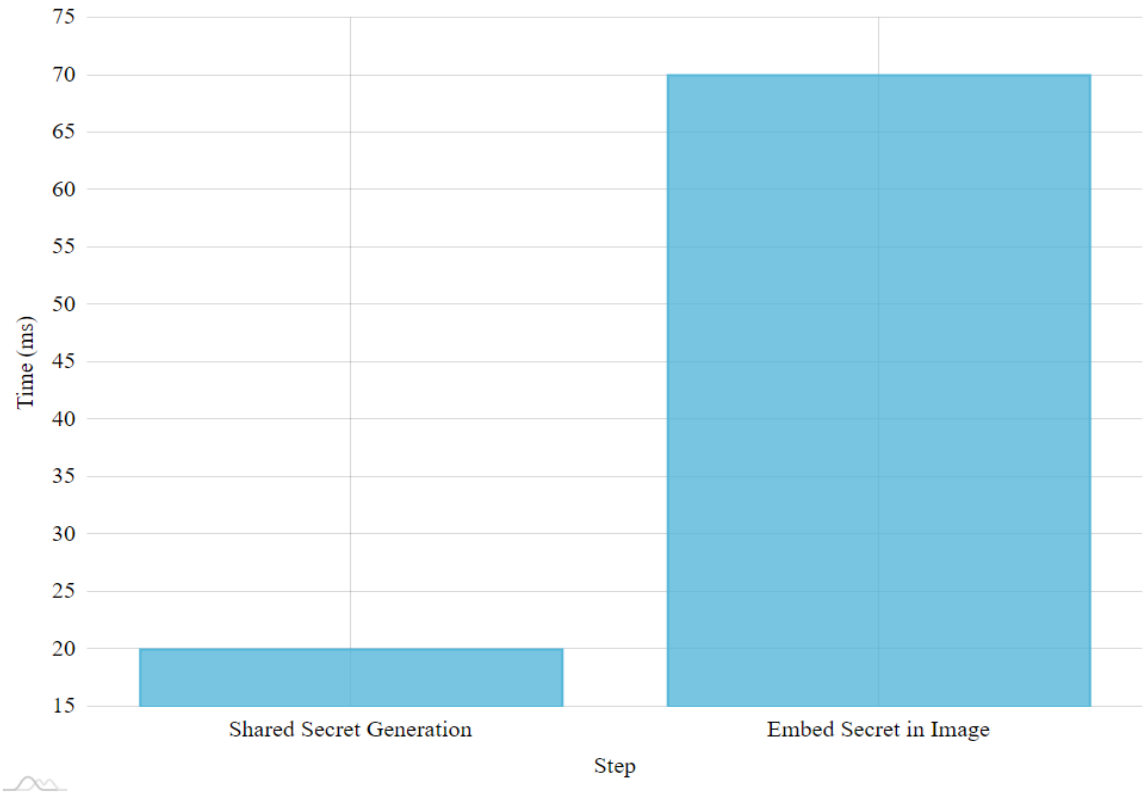



Рисунок 3.3 – Швидкість обробки даних

На діаграма демонструє час, необхідний для трьох основних операцій: шифрування зображення, дешифрування зображення та витягування секрету з зображення. Як показано, шифрування та дешифрування зображення займають приблизно однаковий час, тоді як процес витягування секрету з зображення займає значно більше часу. Це пояснюється складністю процесу аналізу та витягування прихованої інформації з графічного контенту (рис. 3.4).

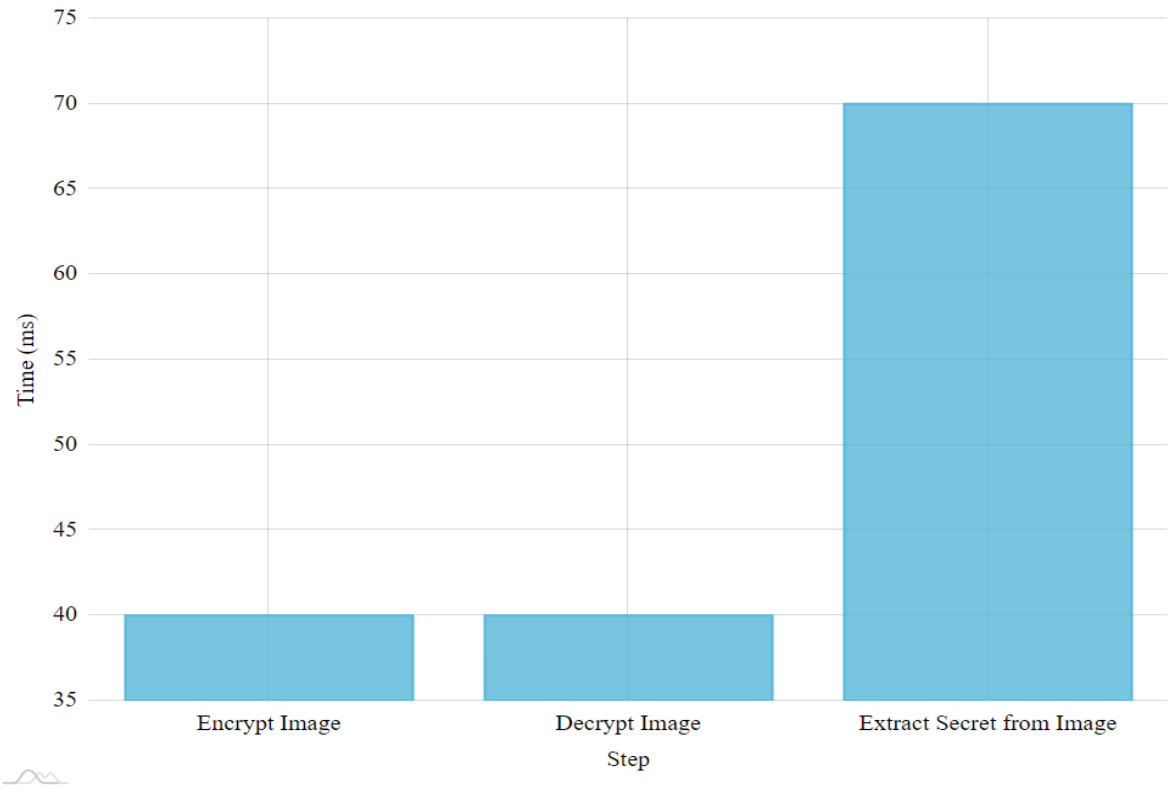



Рисунок 3.4 – Швидкість перевірки

З аналізу представлених діаграм можна зробити висновок, що час, необхідний для різних криптографічних операцій, варіюється в залежності від складності та обчислювальних ресурсів, необхідних для їх виконання. Найбільш ресурсомісткими є операції, пов'язані з генерацією ключів та вбудовуванням/витягуванням секретної інформації з зображення, тоді як обмін ключами, шифрування та дешифрування зображень займають менше часу. Це важливо враховувати при розробці та оптимізації криптографічних систем для забезпечення балансу між безпекою та продуктивністю.

### 3.4 Інструкція користувача

На головній сторінці знаходиться кнопка «login panel» для авторизації користувача у системі. Після натискання на неї користувач потрапляє до форми авторизації (рис. 3.4).

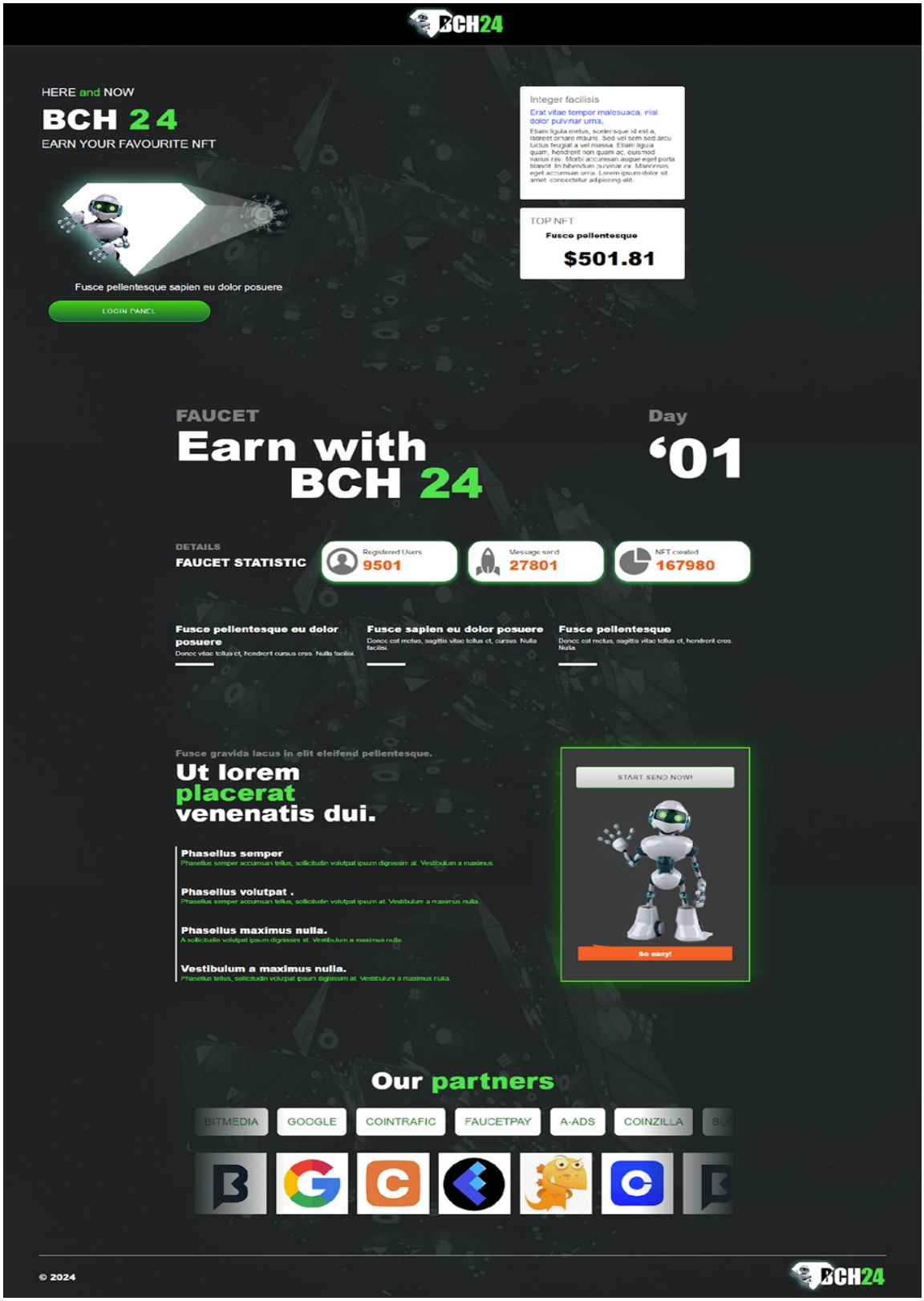



Рисунок 3.4 – Головна сторінка

Якщо дані авторизації вірні користувач отримує доступ до гловного функціоналу сайту (рис. 3.5).

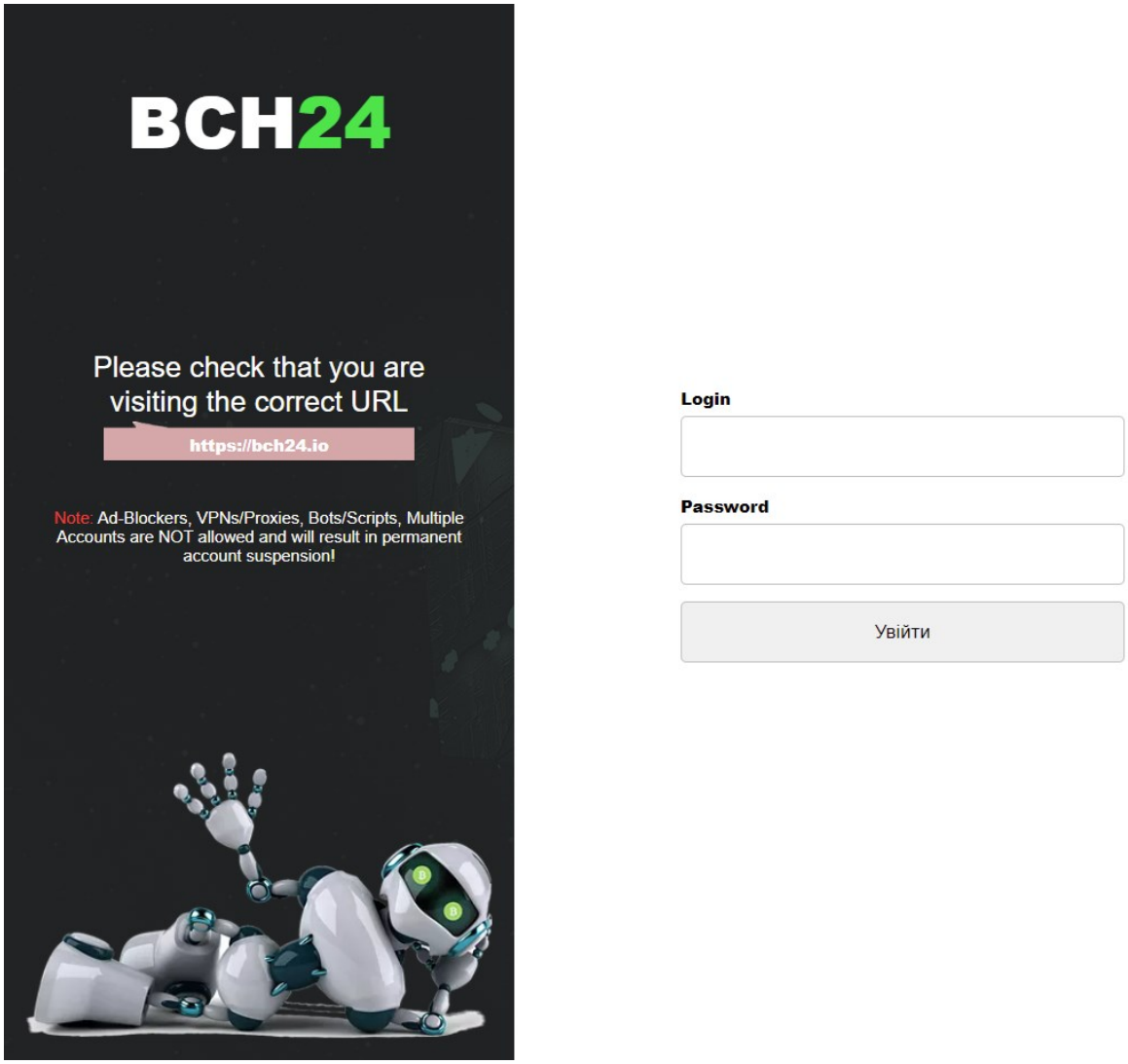



Рисунок 3.5 – Форма авторизації

Для створення зображення необхідно натиснути кнопку «generate img». Буде автоматично створений нове випадкове зображення в яке буде інтегровано підпис користувача і відправлено на сервер.

В низу сторінки розміщено список з переданими даними до серверу і інформацією про них (рис. 3.6).

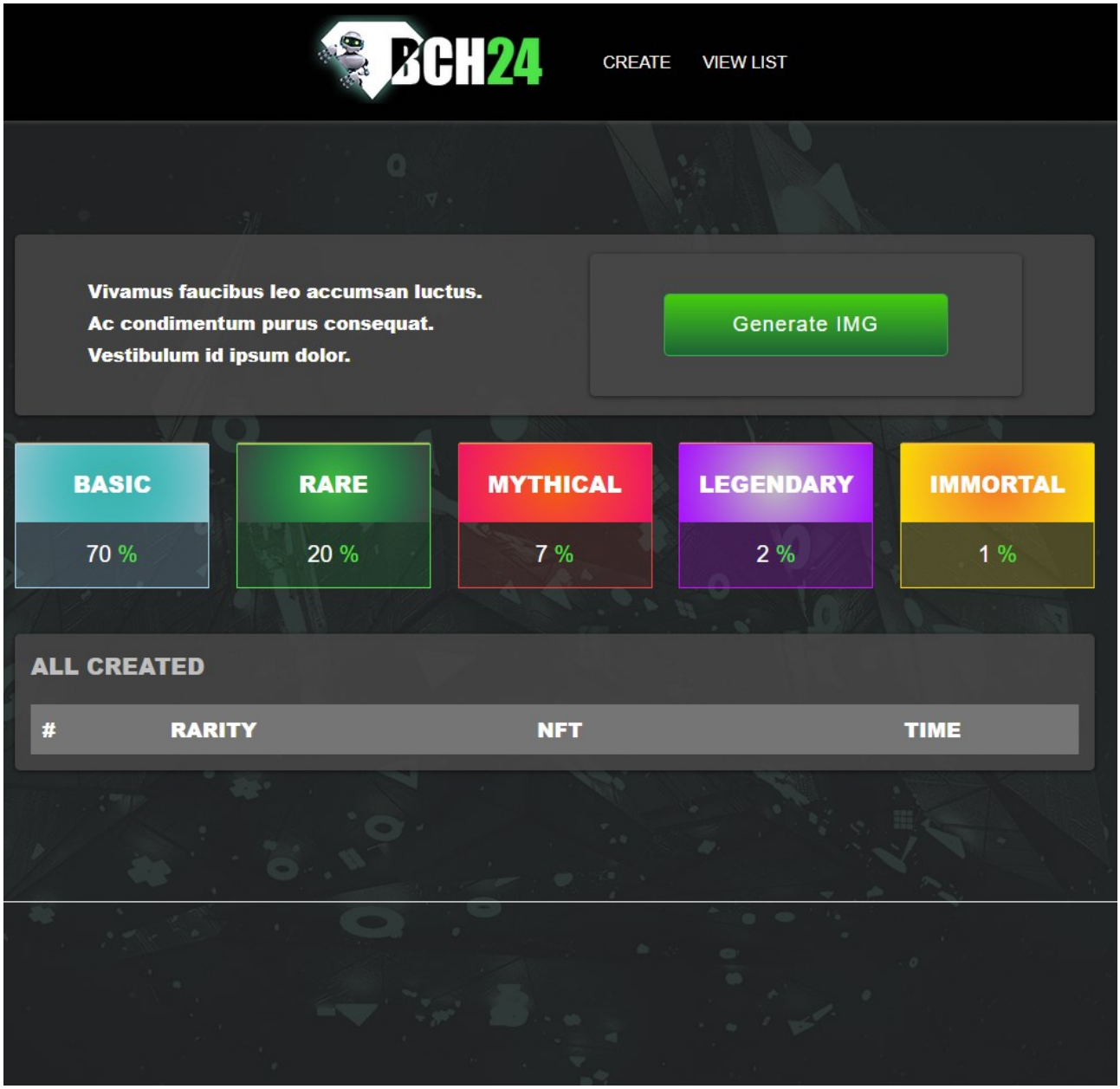



Рисунок 3.6 – Створення NFT зображення

Також в меню можна відкрити сторінку для зручного перегляду всіх зображень (рис. 3.7, 3.8).

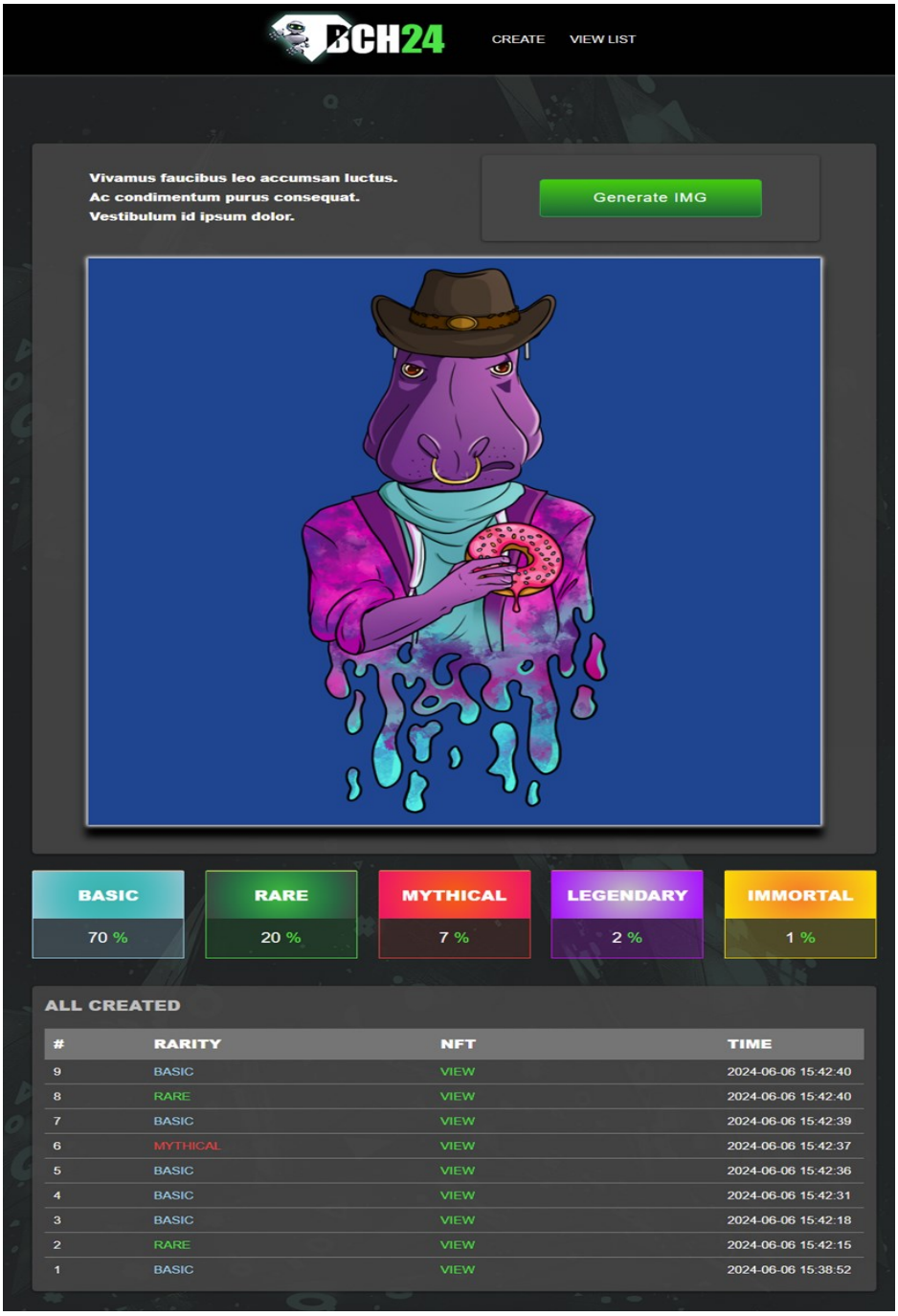



Рисунок 3.7 – Швидкість перевірки

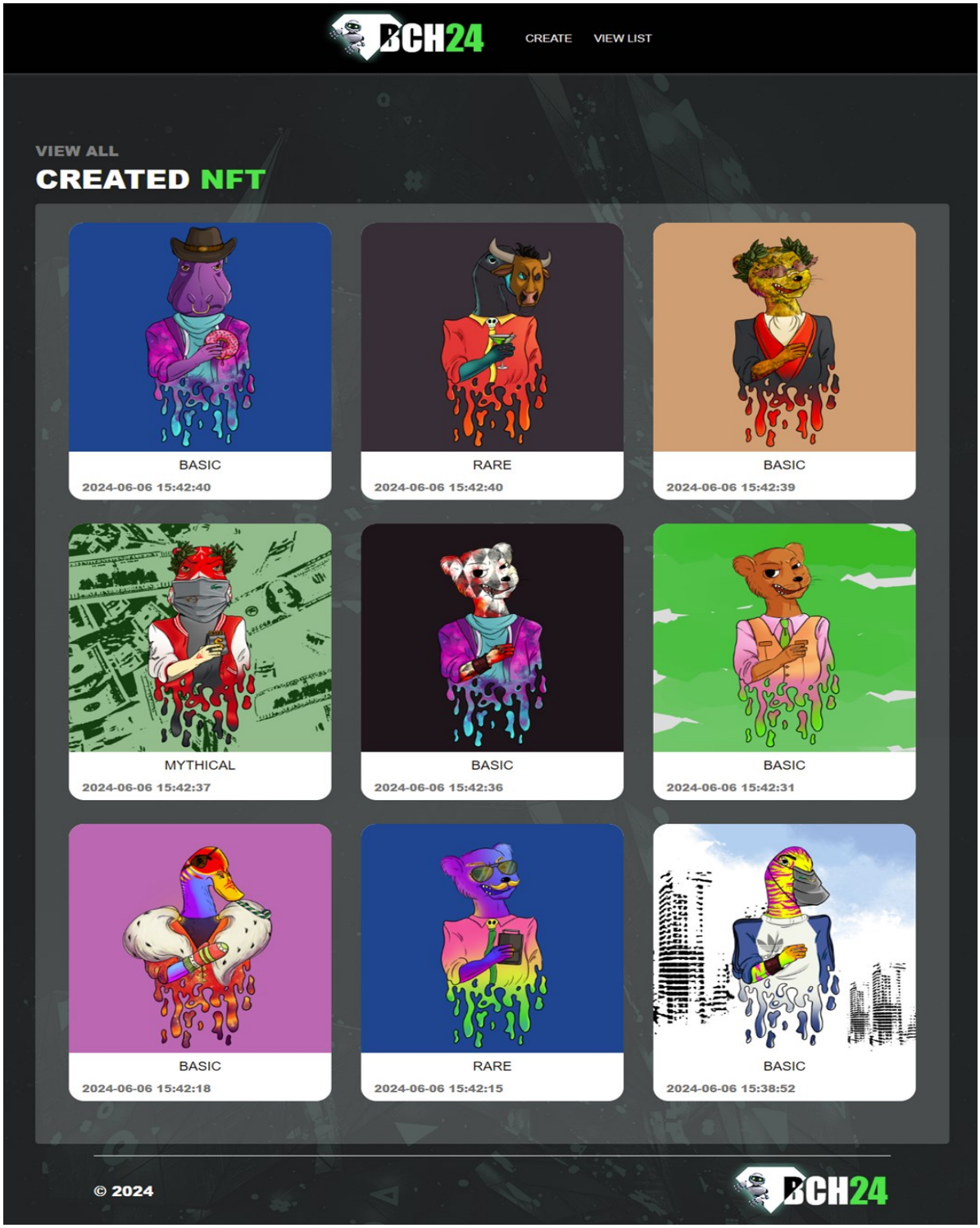



Рисунок 3.8 – Перелік зображень швидкості перевірки

Веб-сайт BCH 24 надає користувачам можливість легко авторизуватися та користуватися основними функціями, такими як створення зображень та перегляд переданих даних. Інтуїтивно зрозумілий інтерфейс дозволяє швидко адаптуватися та ефективно використовувати всі можливості платформи. Всі дії, від авторизації

до створення та перегляду зображень, виконуються просто та зручно, що робить сайт доступним для широкого кола користувачів. ВСН 24 забезпечує надійний та ефективний інструмент для генерації та управління NFT зображеннями, що підтверджує його цінність для користувачів.

### 3.5 Висновки до розділу 3

У цьому розділі описано практичну реалізацію програмного додатку.

Розроблено дизайн користувацького інтерфейсу, орієнтований на зручність та інтуїтивність використання. Враховано сучасні тенденції в UX/UI дизайні, щоб забезпечити користувачам комфортний досвід взаємодії з додатком.

Виконано розробку та налаштування шифрованого каналу для безпечної передачі даних. Реалізовано необхідні протоколи та алгоритми шифрування, що забезпечують захист інформації від несанкціонованого доступу та перехоплення.

Здійснено програмну реалізацію основних функцій сервісу. Написано код, який забезпечує роботу всіх модулів та компонентів, описаних у проектуванні. Проведено тестування та налагодження системи для забезпечення її стабільної та надійної роботи.

Створено детальну інструкцію користувача, яка містить всі необхідні відомості для ефективного використання програмного додатку. Описано основні функції, налаштування та можливості додатку, а також надано рекомендації щодо вирішення можливих проблем під час його експлуатації.

# ВИСНОВКИ

У дипломній роботі було здійснено всебічне дослідження та розробку програмного додатку, який забезпечує безпечну передачу даних.

Предметна область досліджуваної сфери була чітко окреслена. Визначено, що дослідження стосується розробки програмного забезпечення для захищеної передачі даних, що є критично важливим у сучасному інформаційному суспільстві.

Аналіз актуальності розробки показав, що з розвитком інформаційних технологій зростає необхідність у забезпеченні безпеки даних під час їх передачі. Розроблення даного програмного продукту є актуальною і має високий попит на ринку.

Огляд існуючих аналогів продемонстрував, що на ринку вже існують рішення для безпечної передачі даних, проте вони мають певні недоліки. Це підкреслює важливість розробки нового продукту, який би усунув ці недоліки і забезпечив вищий рівень безпеки.

Вибір методів передачі і захисту даних було здійснено на основі ретельного аналізу сучасних методів та технологій. Обрано найбільш ефективні та безпечні способи передачі і захисту даних.

Розроблення архітектури сервісу включала створення детального плану структурної організації програмного забезпечення, що забезпечує його ефективне функціонування та можливість масштабування.

Проектування інтерфейсу було здійснено з урахуванням принципів зручності для користувача. Інтерфейс розроблений таким чином, щоб користувачі могли легко і швидко отримати доступ до всіх функцій програми.

Теоретична Розроблення модуля шифрування включала вибір алгоритмів шифрування, що забезпечують високий рівень безпеки при мінімальних витратах ресурсів.

Розроблення шифрованого каналу забезпечила створення захищеного шляху передачі даних між користувачами, що гарантує конфіденційність і цілісність

інформації.

Програмна реалізація сервісу включала написання коду, тестування та налагодження програмного продукту. Результатом є стабільно функціонуючий сервіс, готовий до впровадження.

Інструкція користувача створена для полегшення роботи з програмою. Вона містить детальні вказівки щодо встановлення, налаштування та використання програмного додатку.

Загалом, виконана дипломна робота дозволила вирішити поставлені завдання та досягти мети дослідження. Розроблений програмний додаток забезпечує надійний захист даних при їх передачі, що підтверджується результатами аналізу. Цей додаток може знайти широке застосування у різних сферах діяльності, де важлива безпека інформації.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. – Wiley. – 2015. URL: https://www.wiley.com/en-us/Applied+Cryptography:+Protocols,+Algorithms,+and+Source+Code+in+C-p-9781119096726 (дата звернення 17.05.2024).

2. Stallings, W. Cryptography and Network Security: Principles and Practice. – Pearson. – 2017. (дата звернення 17.05.2024).

3. Menezes, A., van Oorschot, P., & Vanstone, S. Handbook of Applied Cryptography. – CRC Press. – 2018. (дата звернення 17.05.2024).

4. Katz, J., & Lindell, Y. Introduction to Modern Cryptography. – CRC Press. – 2020. URL: https://www.crcpress.com/Introduction-to-Modern-Cryptography/Katz-Lindell/p/book/9780367331586 (дата звернення 20.05.2024).

5. Provos, N., & Honeyman, P. Hide and Seek: An Introduction to Steganography. – IEEE Security & Privacy. – 1 (3). – 2020. – 32-44 (дата звернення 21.05.2024).

6. Johnson, N. F., Duric, Z., & Jajodia, S. Information Hiding: Steganography and Watermarking - Attacks and Countermeasures. – Springer. – 2021. (дата звернення 21.05.2024).

7. Kessler, G. C. An Overview of Steganography for the Computer Forensics Examiner. – Forensic Science Communications. – 6 (3). – 2018. (дата звернення 23.05.2024).

8. Rivest, R. L., Shamir, A., & Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. – Communications of the ACM. – 21 (2). – 1017. – 120-126 (дата звернення 24.05.2024).

9. Daemen, J., & Rijmen, V. The Design of Rijndael: AES - The Advanced Encryption Standard. – Springer. – 2020. (дата звернення 24.05.2024).

10. Paar, C., & Pelzl, J. Understanding Cryptography: A Textbook for Students and Practitioners. – Springer. – 2020. (дата звернення 24.05.2024).

11. Bishop, M. Computer Security: Art and Science. – Addison-Wesley. – 2022. (дата звернення 24.05.2024).

12. Ferguson, N., Schneier, B., & Kohno, T. Cryptography Engineering: Design Principles and Practical Applications. – Wiley. – 2019. URL: https://www.wiley.com/en-us/Cryptography+Engineering%3A+Design+Principles+and+Practical+Applications-p-9780470474242 (дата звернення 24.05.2024).

13. Anderson, R. Security Engineering: A Guide to Building Dependable Distributed Systems. – Wiley. – 2018. (дата звернення 24.05.2024).

14. Kahn, D. The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. – Scribner. – 2016. URL: https://www.simonandschuster.com/books/The-Codebreakers/David-Kahn/9780684831305 (дата звернення 24.05.2024).

15. Zhang, K., & Li, X. Principles of Secure Communication Systems. – Springer. – 2019. (дата звернення 24.05.2024).

16. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms. – MIT Press. – 2019 (дата звернення 24.05.2024).

17. Pfitzmann, B., & Sadeghi, A. R. Advanced Topics in Cryptography. – Springer. – 2020. (дата звернення 24.05.2024).

18. Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. Information Hiding - A Survey. – Proceedings of the IEEE. – 87 (7). – 2019. – 1062-1078. (дата звернення 1.06.2024).

19. Shamir, A. How to Share a Secret. – Communications of the ACM. – 22 (11). – 2019. – 612-613. (дата звернення 1.06.2024).

20. Wheeler, D., & Needham, R. TEA, a Tiny Encryption Algorithm. – Proceedings of the Fast Software Encryption Workshop. – 2018. (дата звернення 1.06.2024).

21. Bruce, S., & Ferguson, N. Practical Cryptography. – Wiley. – 2023. (дата звернення 1.06.2024).

22. Diffie, W., & Hellman, M. New Directions in Cryptography. – IEEE Transactions on Information Theory. – 22 (6). – 2016. – 644-654. (дата звернення 1.06.2024).

23. Goldwasser, S., & Micali, S. Probabilistic Encryption. – Journal of Computer and System Sciences. – 28 (2). – 2018. – 270-299. (дата звернення 1.06.2024).

24. Boneh, D., & Shoup, V. A Graduate Course in Applied Cryptography. – Cambridge University Press. – 2020. (дата звернення 1.06.2024).

25. Nechvatal, J., et al. Report on the Development of the Advanced Encryption Standard (AES). – Journal of Research of the National Institute of Standards and Technology. – 106 (3). – 2021. – 511-576. (дата звернення 3.06.2024).

26. Schneier, B. Secrets and Lies: Digital Security in a Networked World. – Wiley. – 2020. (дата звернення 3.06.2024).

27. Yang, C. Data Hiding Techniques and Applications. – Springer. – 2018 (дата звернення 3.06.2024).

28. Li, X., & Wang, X. Steganography and Steganalysis in Digital Multimedia: Hiding Information in Images. – Information Science Reference. – 2019. (дата звернення 3.06.2024).

29. Mitra, R. Cryptography and Network Security. – PHI Learning Pvt. Ltd. – 2016. (дата звернення 3.06.2024).

30. Preneel, B. Cryptographic Hash Functions. – Springer. – 2019. (дата звернення 3.06.2024).

# ДОДАТКИ

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

**ЗАТВЕРДЖУЮ**

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

_____Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

# ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

Програмна реалізація шифрованого каналу передавання стеганографічних ключів для закритих систем
08-72.БДР.016.00.074.ТЗ

Керівник бакалаврської дипломної роботи
к.т.н., доцент

_____Адлер О.О._____
"_____" _____

Вінниця – 2024 р.

**1. Найменування та область застосування**

Програмна реалізація шифрованого каналу передавання стеганографічних ключів для закритих систем. Область застосування: шифрований канал передавання стеганографічних ключів для закритих систем.

**2.Підстава для розробки**

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

**3. Мета та призначення розробки**

3.1 Мета розробки: розробка шифрованого каналу передавання стеганографічних ключів для закритих систем.

3.2 Призначення: розроблений програмний засіб шифрування каналу передавання стеганографічних ключів для закритих систем.

**4. Джерела розробки**

4.1. Васюра А. С. Метод шаблонного вбудовування даних у вейвлет-коефіцієнти на основі критерію стеганографічної стійкості / А.С. Васюра; В.В Лукічов // Інформаційні технології та комп’ютерна техніка. — Наукові праці ВНТУ.—2009.-№ 1.-8с

4.2. Васюра А. С. Підвищення ефективності методу шаблонного збудовування даних у зображення / А.С. Васюра; В.В. Лукічов // Інформаційні технології та комп'ютерна техніка. — Наукові праці ВНТУ.— 2008. — № 3. — 10 с.

4.3. K. Hempstalk Digital Invisible Ink Toolkit: Documentation and FAQs [Електронний ресурс] // University of Waikato — 2005. — Режим доступу http://diit.sourceforge.net/doco.html

**5. Вимоги до програми**

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

**6. Вимоги до програмної документації**

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

**7. Вимоги до технічного захисту інформації**

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

**8. Техніко-економічні показники**

8.1 Цінність результатів використання даного проєкту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

## 9. Стадії та етапи розробки

| № з/п | Назва етапів бакалаврської дипломної роботи | Початок | Закінчення |
|---|---|---|---|
| 1. | Визначення напрямку бакалаврської роботи, формулювання теми | | |
| 2. | Аналіз предметної області обраної теми | | |
| 3. | Розробка алгоритму роботи | | |
| 4. | Написання бакалаврської роботи на основі розробленої теми | | |
| 5. | Передзахист бакалаврської дипломної роботи | | |
| 6. | Виправлення, уточнення, корегування бакалаврської дипломної роботи | | |
| 7. | Захист бакалаврської дипломної роботи | | |

## 10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

– ПЗ до бакалаврської дипломної роботи;

- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Михайлина Б.В.

# Додаток Б. Лістинг функцій Php-sodium

```
<?php

/** @generate-function-entries */

function sodium_crypto_aead_aes256gcm_is_available
  (): bool {}

#ifdef HAVE_AESGCM
function sodium_crypto_aead_aes256gcm_decrypt
  (string $ciphertext, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_aes256gcm_encrypt
  (string $message, string $additional_data, string $nonce, string $key) : string {}

function sodium_crypto_aead_aes256gcm_keygen
  () : string {}

# ifdef HAVE_AEAD_DETACHED
function sodium_crypto_aead_aes256gcm_decrypt_detached
  (string $ciphertext, string $mac, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_aes256gcm_encrypt_detached
  (string $message, string $additional_data, string $nonce, string $key) : array {}
# endif
#endif

function sodium_crypto_aead_chacha20poly1305_decrypt
  (string $ciphertext, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_chacha20poly1305_encrypt
  (string $message, string $additional_data, string $nonce, string $key) : string {}

function sodium_crypto_aead_chacha20poly1305_keygen
  () : string {}

function sodium_crypto_aead_chacha20poly1305_ietf_decrypt
  (string $ciphertext, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_chacha20poly1305_ietf_encrypt
  (string $message, string $additional_data, string $nonce, string $key) : string {}

function sodium_crypto_aead_chacha20poly1305_ietf_keygen
  () : string {}

#ifdef HAVE_AEAD_DETACHED
function sodium_crypto_aead_chacha20poly1305_decrypt_detached
  (string $ciphertext, string $mac, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_chacha20poly1305_encrypt_detached
  (string $message, string $additional_data, string $nonce, string $key) : array {}

function sodium_crypto_aead_chacha20poly1305_ietf_decrypt_detached
  (string $ciphertext, string $mac, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_chacha20poly1305_ietf_encrypt_detached
```

```
  (string $message, string $additional_data, string $nonce, string $key) : array {}
#endif

#ifdef crypto_aead_xchacha20poly1305_IETF_NPUBBYTES
function sodium_crypto_aead_xchacha20poly1305_ietf_decrypt
  (string $ciphertext, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_xchacha20poly1305_ietf_keygen
  () : string {}

function sodium_crypto_aead_xchacha20poly1305_ietf_encrypt
  (string $message, string $additional_data, string $nonce, string $key) : string {}

# ifdef HAVE_AEAD_DETACHED
function sodium_crypto_aead_xchacha20poly1305_ietf_decrypt_detached
  (string $ciphertext, string $mac, string $additional_data, string $nonce, string $key) : string | FALSE {}

function sodium_crypto_aead_xchacha20poly1305_ietf_encrypt_detached
  (string $message, string $additional_data, string $nonce, string $key) : array {}
# endif
#endif

function sodium_crypto_auth
  (string $message, string $key) : string {}

function sodium_crypto_auth_keygen
  () : string {}

function sodium_crypto_auth_verify
  (string $mac, string $message, string $key) : bool {}

function sodium_crypto_box
  (string $message, string $nonce, string $key_pair) : string {}

function sodium_crypto_box_keypair
  () : string {}

function sodium_crypto_box_seed_keypair
  (string $seed) : string {}

function sodium_crypto_box_keypair_from_secretkey_and_publickey
  (string $secret_key, string $public_key) : string {}

function sodium_crypto_box_open
  (string $ciphertext, string $nonce, string $key_pair) : string | FALSE {}

function sodium_crypto_box_publickey
  (string $key_pair) : string {}

function sodium_crypto_box_publickey_from_secretkey
  (string $secret_key) : string {}

function sodium_crypto_box_seal
  (string $message, string $key_pair) : string {}

function sodium_crypto_box_seal_open
  (string $ciphertext, string $key_pair) : string | FALSE {}
```

```
function sodium_crypto_box_secretkey
  (string $key_pair) : string {}

#ifdef crypto_core_ristretto255_HASHBYTES
function sodium_crypto_core_ristretto255_add(string $p, string $q): string {}

function sodium_crypto_core_ristretto255_from_hash(string $s): string {}

function sodium_crypto_core_ristretto255_is_valid_point(string $s): bool {}

function sodium_crypto_core_ristretto255_random(): string {}

function sodium_crypto_core_ristretto255_scalar_add(string $p, string $q): string {}

function sodium_crypto_core_ristretto255_scalar_complement(string $s): string {}

function sodium_crypto_core_ristretto255_scalar_invert(string $s): string {}

function sodium_crypto_core_ristretto255_scalar_mul(string $x, string $y): string {}

function sodium_crypto_core_ristretto255_scalar_negate(string $s): string {}

function sodium_crypto_core_ristretto255_scalar_random(): string {}

function sodium_crypto_core_ristretto255_scalar_reduce(string $s): string {}

function sodium_crypto_core_ristretto255_scalar_sub(string $p, string $q): string {}

function sodium_crypto_core_ristretto255_sub(string $p, string $q): string {}
#endif

function sodium_crypto_kx_keypair
  () : string {}

function sodium_crypto_kx_publickey
  (string $key_pair) : string {}

function sodium_crypto_kx_secretkey
  (string $key_pair) : string {}

function sodium_crypto_kx_seed_keypair
  (string $seed) : string {}

function sodium_crypto_kx_client_session_keys
  (string $client_key_pair, string $server_key) : array {}

function sodium_crypto_kx_server_session_keys
  (string $server_key_pair, string $client_key) : array {}

function sodium_crypto_generichash
  (string $message, string $key = "", int $length = SODIUM_CRYPTO_GENERICHASH_BYTES) : string {}

function sodium_crypto_generichash_keygen
  () : string {}

function sodium_crypto_generichash_init
  (string $key = "", int $length = SODIUM_CRYPTO_GENERICHASH_BYTES) : string {}
```

```
function sodium_crypto_generichash_update
  (string &$state, string $message) : bool {}

function sodium_crypto_generichash_final
  (string &$state, int $length = SODIUM_CRYPTO_GENERICHASH_BYTES) : string {}

function sodium_crypto_kdf_derive_from_key
  (int $subkey_length, int $subkey_id, string $context, string $key) : string {}

function sodium_crypto_kdf_keygen
  () : string {}

#ifdef crypto_pwhash_SALTBYTES
function sodium_crypto_pwhash
  (int $length, string $password, string $salt, int $opslimit, int $memlimit, int $algo =
SODIUM_CRYPTO_PWHASH_ALG_DEFAULT) : string {}

function sodium_crypto_pwhash_str
  (string $password, int $opslimit, int $memlimit) : string {}

function sodium_crypto_pwhash_str_verify
  (string $hash, string $password) : bool {}
#endif

#if SODIUM_LIBRARY_VERSION_MAJOR > 9  ||  (SODIUM_LIBRARY_VERSION_MAJOR == 9 &&
SODIUM_LIBRARY_VERSION_MINOR >= 6)
function sodium_crypto_pwhash_str_needs_rehash
  (string $password, int $opslimit, int $memlimit) : bool {}
#endif

#ifdef crypto_pwhash_scryptsalsa208sha256_SALTBYTES
function sodium_crypto_pwhash_scryptsalsa208sha256
  (int $length, string $password, string $salt, int $opslimit, int $memlimit) : string {}

function sodium_crypto_pwhash_scryptsalsa208sha256_str
  (string $password, int $opslimit, int $memlimit) : string {}

function sodium_crypto_pwhash_scryptsalsa208sha256_str_verify
  (string $hash, string $password) : bool {}
#endif

function sodium_crypto_scalarmult
  (string $n, string $p) : string {}

#ifdef crypto_core_ristretto255_HASHBYTES
function sodium_crypto_scalarmult_ristretto255(string $n, string $p): string {}

function sodium_crypto_scalarmult_ristretto255_base(string $n): string {}
#endif

function sodium_crypto_secretbox
  (string $message, string $nonce, string $key) : string {}

function sodium_crypto_secretbox_keygen
  () : string {}

function sodium_crypto_secretbox_open
  (string $ciphertext, string $nonce, string $key) : string | FALSE {}
```

```
#ifdef crypto_secretstream_xchacha20poly1305_ABYTES
function sodium_crypto_secretstream_xchacha20poly1305_keygen
  () : string {}

function sodium_crypto_secretstream_xchacha20poly1305_init_push
  (string $key) : array {}

function sodium_crypto_secretstream_xchacha20poly1305_push
  (string &$state, string $message, string $additional_data = "", int $tag  =
SODIUM_CRYPTO_SECRETSTREAM_XCHACHA20POLY1305_TAG_MESSAGE) : string {}

function sodium_crypto_secretstream_xchacha20poly1305_init_pull
  (string $header, string $key) : string {}

function sodium_crypto_secretstream_xchacha20poly1305_pull
  (string &$state, string $ciphertext, string $additional_data = "") : array | FALSE {}

function sodium_crypto_secretstream_xchacha20poly1305_rekey
  (string &$state) : void {}
#endif

function sodium_crypto_shorthash
  (string $message, string $key) : string {}

function sodium_crypto_shorthash_keygen
  () : string {}

function sodium_crypto_sign
  (string $message, string $secret_key) : string {}

function sodium_crypto_sign_detached
  (string $message, string $secret_key) : string {}

function sodium_crypto_sign_ed25519_pk_to_curve25519
  (string $public_key) : string {}

function sodium_crypto_sign_ed25519_sk_to_curve25519
  (string $secret_key) : string {}

function sodium_crypto_sign_keypair
  () : string {}

function sodium_crypto_sign_keypair_from_secretkey_and_publickey
  (string $secret_key, string $public_key) : string {}

function sodium_crypto_sign_open
  (string $ciphertext, string $public_key) : string | FALSE {}

function sodium_crypto_sign_publickey
  (string $key_pair) : string {}

function sodium_crypto_sign_secretkey
  (string $key_pair) : string {}

function sodium_crypto_sign_publickey_from_secretkey
  (string $secret_key) : string {}
```

```
function sodium_crypto_sign_seed_keypair
  (string $seed) : string {}

function sodium_crypto_sign_verify_detached
  (string $signature, string $message, string $public_key) : bool {}

function sodium_crypto_stream
  (int $length, string $nonce, string $key) : string {}

function sodium_crypto_stream_keygen
  () : string {}

function sodium_crypto_stream_xor
  (string $message, string $nonce, string $key) : string {}

function sodium_crypto_stream_xchacha20
  (int $length, string $nonce, string $key) : string {}

function sodium_crypto_stream_xchacha20_keygen
  () : string {}

function sodium_crypto_stream_xchacha20_xor
  (string $message, string $nonce, string $key) : string {}

/* ----- helpers ----- */

function sodium_add
  (string &$string1, string $string2) : void {}

function sodium_compare
  (string $string1, string $string2) : int {}

function sodium_increment
  (string &$string) : void {}

function sodium_memcmp
  (string $string1, string $string2) : int {}

function sodium_memzero
  (string &$string) : void {}

function sodium_pad
  (string $string, int $length) : string {}

function sodium_unpad
  (string $string, int $block_size) : string {}

/* ----- codecs ----- */

function sodium_bin2hex
  (string $string) : string {}

function sodium_hex2bin
  (string $string, string $ignore = "") : string {}

#ifdef sodium_base64_VARIANT_ORIGINAL
function sodium_bin2base64
  (string $string, int $id) : string {}
```

```
function sodium_base642bin
  (string $string, int $id, string $ignore = "") : string {}
#endif

/* ----- aliases ----- */

/** @alias sodium_crypto_box_publickey_from_secretkey */
function sodium_crypto_scalarmult_base
  (string $secret_key) : string {}
```

# Додаток В. Лістинг файлу index.php

```
<div class="fixed-bg">
	<div class="darkness-bg">
		<div class="wrap main-p">

			<div class="menu">
				<div class="left">
					<a href="#"><img src="/img/logo.webp" alt="Logo" ></a>
				</div>
			</div>

			<div class="earn">
				<div class="left">
					<h1 class="title">
						<p class="small">HERE <span>and</span> NOW</p>
						<p class="big">BCH <span class="tik-one">2</span> <span
class="tik-two">4</span></p>
					</h1>

					<div class="like-img">
						<h2>EARN YOUR FAVOURITE NFT</h2>
						<img src="img/earn-bg.webp" >
						<h3>Fusce pellentesque sapien eu dolor posuere</h3>
					</div>

					<a href="/main/login" class="btn">LOGIN PANEL <i></i></a>
				</div>

				<div class="right">
					<div class="column less">
						<div class="item">
							<h4 class="title">Integer facilisis</h4>
							<a class="news" >Erat vitae tempor malesuada, nisl dolor
pulvinar urna,</a>
							<p class="text">
Etiam ligula metus, scelerisque id est a, laoreet ornare mauris. Sed vel sem sed arcu luctus feugiat a vel massa.
Etiam ligula quam, hendrerit non quam ac, euismod varius nisi. Morbi accumsan augue eget porta blandit.
In bibendum pulvinar ex. Maecenas eget accumsan urna. Lorem ipsum dolor sit amet, consectetur adipiscing elit..
							</p>
						</div>

						<div class="item">
							<h4 class="title">TOP NFT</h4>

							<div class="line">
								<p class="one">Fusce pellentesque</p>
							</div>
							<p class="price">$501.81</p>
						</div>

					</div>
				</div>
			</div>

			<div class="stat wrap-center">
				<div class="day">
```

```
                <div class="left">
                    <p class="gray">FAUCET</p>
                    <p class="with">Earn with</p>
                    <p class="bch">BCH <span>24</span></p>
                </div>
                <div class="right">
                    <p class="gray">Day</p>
                    <p class="date">‘01</p>
                </div>
            </div>

            <div class="details">
                <div class="item">
                    <p class="gray">DETAILS</p>
                    <p class="white">FAUCET STATISTIC</p>
                </div>
                <div class="item">
                    <div class="img">
                        <img src="img/i1.png">
                    </div>
                    <div class="text">
                        <p class="about">Registered Users</p>
                        <p class="qty">9501</p>
                    </div>
                </div>
                <div class="item">
                    <div class="img">
                        <img src="img/i2.png">
                    </div>
                    <div class="text">
                        <p class="about">Message send</p>
                        <p class="qty">27801</p>
                    </div>
                </div>
                <div class="item">
                    <div class="img">
                        <img src="img/i3.png">
                    </div>
                    <div class="text">
                        <p class="about">NFT created</p>
                        <p class="qty">167980</p>
                    </div>
                </div>
            </div>


            <div class="about">
                <div class="item">
                    <div class="top">
                        <h3>Fusce pellentesque eu dolor posuere</h3>
                        <p>Donec vitae tellus et, hendrerit cursus eros. Nulla
facilisi.</p>
                    </div>
                    <img src="img/white-line.png">
                </div>
                <div class="item">
                    <div class="top">
                        <h3>Fusce sapien eu dolor posuere</h3>
```

```
                                        <p>Donec est metus, sagittis vitae tellus et, cursus. Nulla
facilisi.</p>
                                    </div>
                                    <img src="img/white-line.png">
                                </div>
                                <div class="item">
                                    <div class="top">
                                        <h3>Fusce pellentesque</h3>
                                        <p>Donec est metus, sagittis vitae tellus et, hendrerit eros.
Nulla.</p>
                                    </div>
                                    <img src="img/white-line.png">
                                </div>
                            </div>
                        </div>


                        <div class="robot wrap-center">
                            <div class="left">
                                <h2>Fusce gravida lacus in elit eleifend pellentesque. </h2>
                                <h3>Ut lorem <br> <span>placerat</span> <br> venenatis dui.</h3>

                                <div class="text">
                                    <p class="title">Phasellus semper  </p>
                                    <p class="about">Phasellus semper accumsan tellus, sollicitudin
volutpat ipsum dignissim at. Vestibulum a maximus. </p>

                                    <p class="title">Phasellus volutpat . </p>
                                    <p class="about">Phasellus semper accumsan tellus, sollicitudin
volutpat ipsum at. Vestibulum a maximus nulla. </p>

                                    <p class="title">Phasellus maximus nulla. </p>
                                    <p class="about">A sollicitudin volutpat ipsum dignissim at.
Vestibulum a maximus nulla. </p>

                                    <p class="title"> Vestibulum a maximus nulla. </p>
                                    <p class="about">Phasellus tellus, sollicitudin volutpat ipsum
dignissim at. Vestibulum a maximus nulla. </p>

                                </div>
                            </div>

                            <div class="right">
                                <div class="start">
                                    <a class="btn">START SEND NOW!</a>
                                    <img src="img/robot.png">
                                </div>

                                <div class="url">
                                    <div class="link-bg">
                                        <a>So easy!</a>
                                    </div>
                                </div>
                            </div>
                        </div>

                        <div class="sliders">
                            <p class="title">Our <span>partners</span></p>
```

```
<div class="slider-one">
        <div class="slide-track">
                <div class="slide">
                        <p>BITMEDIA</p>
                </div>
                <div class="slide">
                        <p>GOOGLE</p>
                </div>
                <div class="slide">
                        <p>COINTRAFIC</p>
                </div>
                <div class="slide">
                        <p>FAUCETPAY</p>
                </div>
                <div class="slide">
                        <p>A-ADS</p>
                </div>
                <div class="slide">
                        <p>COINZILLA</p>
                </div>
                <div class="slide">
                        <p>BLOCKCHAIN</p>
                </div>
                <div class="slide">
                        <p>CRYPTOCOINSAD</p>
                </div>
                <div class="slide">
                        <p>BITMEDIA</p>
                </div>
                <div class="slide">
                        <p>GOOGLE</p>
                </div>
                <div class="slide">
                        <p>COINTRAFIC</p>
                </div>
                <div class="slide">
                        <p>FAUCETPAY</p>
                </div>
                <div class="slide">
                        <p>A-ADS</p>
                </div>
                <div class="slide">
                        <p>COINZILLA</p>
                </div>
                <div class="slide">
                        <p>BLOCKCHAIN</p>
                </div>
                <div class="slide">
                        <p>CRYPTOCOINSAD</p>
                </div>
        </div>
</div>

<div class="slider-two">
        <div class="slide-track">
                <img src="https://i.ibb.co/FmzKHsR/bitmedia.jpg" >
                <img src="https://i.ibb.co/rZCdXqL/google.jpg" >
                <img src="https://i.ibb.co/dczDnSc/cointrafic.jpg" >
```

```
                                        <img src="https://i.ibb.co/1K9CV0f/faucet.jpg" >
                                        <img src="https://i.ibb.co/TW67xGP/zilla.jpg" >
                                        <img src="https://i.ibb.co/GF89B65/crypto.jpg" >
                                        <img src="https://i.ibb.co/vVL2yqH/a-ads.jpg"  >

                                        <img src="https://i.ibb.co/FmzKHsR/bitmedia.jpg" >
                                        <img src="https://i.ibb.co/rZCdXqL/google.jpg" >
                                        <img src="https://i.ibb.co/dczDnSc/cointrafic.jpg" >
                                        <img src="https://i.ibb.co/1K9CV0f/faucet.jpg" >
                                        <img src="https://i.ibb.co/TW67xGP/zilla.jpg" >
                                        <img src="https://i.ibb.co/GF89B65/crypto.jpg" >
                                        <img src="https://i.ibb.co/vVL2yqH/a-ads.jpg"  >
                                    </div>
                                </div>
                            </div>

                            <div class="footer">


                                <div class="copy">
                                    <div class="left">
                                        <p class="year">&amp;#169; 2024 </p>
                                    </div>
                                    <div class="right">
                                        <a href="#"><img src="img/logo.webp" alt="Logo" ></a>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
```

# Додаток Г. Лістинг загального алгоритму

```
// Генерація пари ключів Curve25519
function generate_keypair() {
   $private_key = sodium_crypto_box_keypair();
   $public_key = sodium_crypto_box_publickey($private_key);
   return ['private_key' => $private_key, 'public_key' => $public_key];
}

// Генерація спільного секрету
function generate_shared_secret($private_key, $peer_public_key) {
   return sodium_crypto_scalarmult(sodium_crypto_box_secretkey($private_key), $peer_public_key);
}

// Шифрування повідомлення за допомогою ChaCha20
function encrypt_message($message, $key, $nonce) {
   return sodium_crypto_aead_chacha20poly1305_ietf_encrypt($message, '', $nonce, $key);
}

// Дешифрування повідомлення за допомогою ChaCha20
function decrypt_message($encrypted_message, $key, $nonce) {
   return sodium_crypto_aead_chacha20poly1305_ietf_decrypt($encrypted_message, '', $nonce, $key);
}

// Вбудовування секрету в зображення
function embed_secret_in_image($image_path, $secret) {
   $image = imagecreatefrompng($image_path);
   $width = imagesx($image);
   $height = imagesy($image);

   $secret_bin = '';
   for ($i = 0; $i < strlen($secret); $i++) {
      $secret_bin .= str_pad(decbin(ord($secret[$i])), 8, '0', STR_PAD_LEFT);
   }

   $index = 0;
   for ($y = 0; $y < $height; $y++) {
      for ($x = 0; $x < $width; $x++) {
         $rgb = imagecolorat($image, $x, $y);
         $r = ($rgb >> 16) & 0xFF;
         $g = ($rgb >> 8) & 0xFF;
         $b = $rgb & 0xFF;

         if ($index < strlen($secret_bin)) {
            $b = ($b & 0xFE) | $secret_bin[$index];
            $index++;
         }

         $color = imagecolorallocate($image, $r, $g, $b);
         imagesetpixel($image, $x, $y, $color);
      }
   }

   imagepng($image, 'output.png');
   imagedestroy($image);
```

```
}

// Витягування секрету з зображення
function extract_secret_from_image($image_path, $secret_length) {
   $image = imagecreatefrompng($image_path);
   $width = imagesx($image);
   $height = imagesy($image);

   $secret_bin = '';
   for ($y = 0; $y < $height; $y++) {
      for ($x = 0; $x < $width; $x++) {
         $rgb = imagecolorat($image, $x, $y);
         $b = $rgb & 0xFF;

         $secret_bin .= $b & 0x01;

         if (strlen($secret_bin) >= $secret_length * 8) {
            break 2;
         }
      }
   }

   $secret = '';
   for ($i = 0; $i < strlen($secret_bin); $i += 8) {
      $secret .= chr(bindec(substr($secret_bin, $i, 8)));
   }

   imagedestroy($image);
   return $secret;
}

// Шифрування зображення
function encrypt_image($image_path, $key, $nonce) {
   $image_data = file_get_contents($image_path);
   return sodium_crypto_aead_chacha20poly1305_ietf_encrypt($image_data, '', $nonce, $key);
}

// Дешифрування зображення
function decrypt_image($encrypted_image_data, $key, $nonce) {
   return sodium_crypto_aead_chacha20poly1305_ietf_decrypt($encrypted_image_data, '', $nonce, $key);
}

$alice_keys = generate_keypair();
$bob_keys = generate_keypair();

// Взаємний обмін публічними ключами
$alice_public_key = $alice_keys['public_key'];
$bob_public_key = $bob_keys['public_key'];

// Генерація спільного секрету
$alice_shared_secret = generate_shared_secret($alice_keys['private_key'], $bob_public_key);
$bob_shared_secret = generate_shared_secret($bob_keys['private_key'], $alice_public_key);

if ($alice_shared_secret !== $bob_shared_secret) {
   die('Shared secrets do not match!');
}

// Вбудовування спільного секрету в зображення
```

```
$original_image = 'input.png'; // Оригінальне зображення
$secret = bin2hex($alice_shared_secret); // Перетворення секрету в рядок
embed_secret_in_image($original_image, $secret);

// Шифрування зображення
$nonce = random_bytes(SODIUM_CRYPTO_AEAD_CHACHA20POLY1305_IETF_NPUBBYTES);
$encrypted_image = encrypt_image('output.png', $alice_shared_secret, $nonce);

// Передача зашифрованого зображення (пропускається)

// Дешифрування зображення
$decrypted_image_data = decrypt_image($encrypted_image, $bob_shared_secret, $nonce);

// Збереження дешифрованого зображення
file_put_contents('decrypted_output.png', $decrypted_image_data);

// Витягування секрету з зображення
$extracted_secret = extract_secret_from_image('decrypted_output.png', strlen($secret));
```

**Додаток Д. Ілюстративний матеріал**

# Програмна реалізація шифрованого каналу передавання стеганографічних ключів для закритих систем

Виконав студент групи 1КІТС-20б Михайлина Б.В.
Науковий керівник к.т.н.,доц.,зав. каф. МБІС Адлер О.О.

## Актуальність теми

Актуальність проблеми полягає в необхідності забезпечення надійного шифрування та передачі даних в умовах зростаючої кількості кібератак. Стеганографія дозволяє приховувати факт передачі секретної інформації, що додатково захищає дані.

## Мета роботи

Метою дослідження є розробка та реалізація шифрованого каналу передавання стеганографічних ключів для захищених систем. Це забезпечить подвійний захист передаваних даних, поєднуючи шифрування та стеганографію.

## Новизна

Новизна роботи полягає в інтеграції методів шифрування та стеганографії для створення шифрованого каналу передачі ключів, що забезпечує вищий рівень безпеки порівняно з традиційними методами. Це новий підхід до захисту даних, який поєднує переваги двох різних технологій.

# Об’єкт та предмет дослідження

Об'єктом дослідження є процеси шифрування та стеганографії в інформаційних системах, включаючи вивчення існуючих методів шифрування даних та способів приховування інформації.

Предметом дослідження є методи та засоби шифрованого передавання стеганографічних ключів, розробка та оцінка алгоритмів для надійної передачі ключів через шифровані канали зв'язку.

# Теоретичні основи криптографії

Криптографія є основою для забезпечення конфіденційності та цілісності інформації в сучасних інформаційних системах. Основні принципи включають симетричне та асиметричне шифрування.

**Симетричне шифрування** використовує один ключ для шифрування і розшифрування даних, що забезпечує швидкість операцій, але вимагає безпечного обміну ключами.

**Асиметричне шифрування** використовує пару ключів: публічний (для шифрування) та приватний (для розшифрування). Цей підхід забезпечує безпеку обміну ключами, але вимагає більших обчислювальних ресурсів.

# Запропонований метод

Пропонований метод передбачає використання алгоритмів curve25519 для обміну ключами та ChaCha20 для шифрування даних в захищених системах. Основні аспекти методу включають:

Використання curve25519 для генерації асиметричних ключів, що забезпечує високу швидкість обчислень та стійкість до криптографічних атак.

Використання ChaCha20 для симетричного шифрування даних, забезпечуючи високу швидкість шифрування та дешифрування.

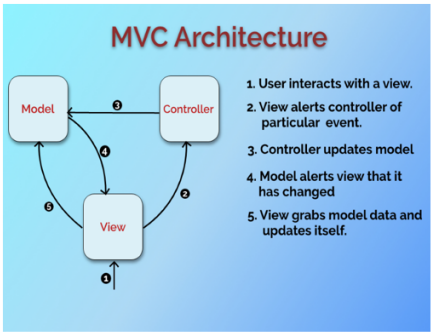

# Архітектура застосунку

Розробка цих методів буде відбуватися на основі шаблону проектування MVC. Це архітектурний шаблон, що використовується в розробці програмного забезпечення для розділення компонентів програми на три основні частини: модель (Model) , вид (View) та контролер (Controller).

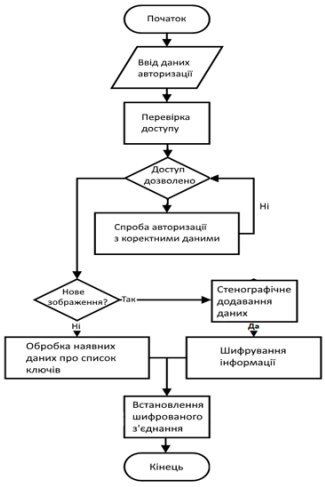
Модель (Model) представляє дані та бізнес-логіку програми. Модель відповідає за доступ до даних, їх обробку та збереження.

Вид (View) представляє інтерфейс користувача програми.

Контролер (Controller) обробляє взаємодію між користувачем, моделлю та видом.



Представлено блок-схему роботи системи

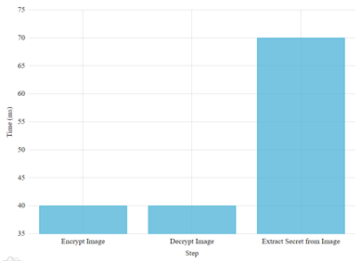



# Аналіз ефективності розробленого модуля

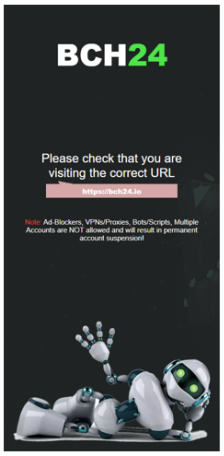
Діаграми, які показують продуктивність і безпеку на різних етапах процесу. Можна порівняти час, необхідний для генерації ключів, шифрування, дешифрування та вбудовування/витягування секрету з зображення.

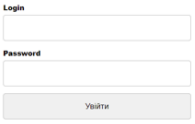






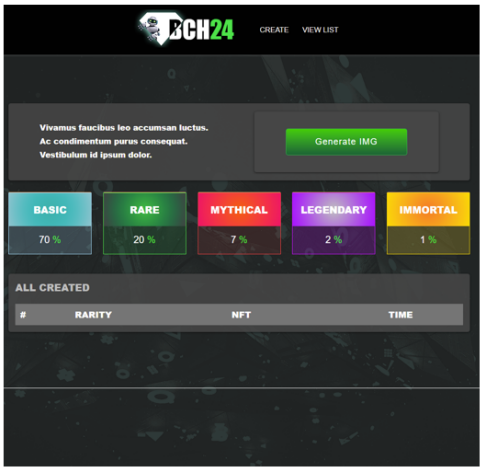
# Форма авторизації веб-сервісу




Якщо дані авторизації вірні користувач отримує доступ до головного функціоналу сайту

# Генерація зображення




Для створення зображення необхідно натиснути кнопку «generate img». Буде автоматично створений нове випадкове зображення в яке буде інтегровано підпис користувача і відправлено на сервер.

В низу сторінки розміщено список з переданими даними до серверу і інформацією про них

# Дійсні аналоги

OpenStego

**Основне призначення**: Вбудовування конфіденційної інформації в медіафайли.

**Особливості**: Відкритий код, підтримка різних типів медіафайлів (зображення, звуки).

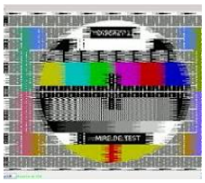
**Використання для ключів**: Можливість зашифрувати стеганографічні ключі та вбудувати їх у зображення для безпечної передачі.


Steghide

**Основне призначення**: Сховання даних у медіафайли (особливо в зображення).

**Особливості**: Висока ефективність стеганографічного вбудовування, підтримка шифрування.

**Використання для ключів**: Шифрування стеганографічних ключів і вбудовування їх у зображення для забезпечення конфіденційності.



CryptImage

**Основне призначення**: Захист конфіденційної інформації, вбудовування її у зображення.

**Особливості**: Простота використання, фокус на безпеці і конфіденційності.

**Використання для ключів**: Зашифрування і вбудовування стеганографічних ключів у зображення для безпечної передачі.

# Переваги та недоліки розробленого застосунку

Переваги:

- Висока швидкість
- Ефективність реалізкації
- Висока безпека
- Стійкість до атак через стороні канали
- Простота розуміння і використання

Недоліки:

- Відсутність широкого прийняття
- Відносно молодий алгоритм
- Потреба у безпечному обміні ключами
- Відносно низька стійкість до брутфорсу

# Висновок

У цьому дослідженні розглядається проблема забезпечення передачі стеганографічних ключів у захищеній системі за допомогою використання зашифрованих каналів. Проведено аналіз існуючих криптографічних і стеганографічних методів для виявлення їх переваг і обмежень. Основним результатом дослідження стала розробка та впровадження методу на основі алгоритму Curve25519 для обміну ключами та алгоритму ChaCha20 для симетричного шифрування.Експериментальна оцінка підтверджує високу ефективність і стійкість нашого підходу за різних умов навантаження та типів переданих даних.Отримані результати демонструють можливість успішної реалізації розробленого методу в практичній сфері захисту конфіденційної інформації в мережах зв'язку та інформаційних системах.

Це дослідження відкриває перспективу подальших досліджень для вдосконалення алгоритму шифрування та його інтеграції зі стеганографічними методами для забезпечення ще більш високого рівня безпеки.

ДЯКУЮ ЗА УВАГУ

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Програмна реалізація шифрового каналу передавання стеганографічних ключів для закритих систем

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 96 % Схожість 4 %

Аналіз звіту подібності (відмітити потрібне):

**1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Коваль Н.П. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Михайлина Б.В. (прізвище, ініціали)

Керівник роботи _____ (підпис) Адлер О.О. (прізвище, ініціали)