МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

САПІЖУК ІВАН ТАРАСОВИЧ

Алгоритми захисту конфіденційності та автентичності Інтернет-речей з використанням стеганографії / Algorithms for protection of confidentiality and authenticity of Internet of things using steganography

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБм -21
І.Т. Сапіжук

Науковий керівник
к.т.н., доцент Якименко І. З.

Кваліфікаційну роботу
Допущено до захисту:

«____»_____ ___2024р.

Завідувач кафедри **В.В.Яцків**

ТЕРНОПІЛЬ – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 – Кібербезпека та захист інформації
освітньо-професійна програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
«____» _____ 2023 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

САПІЖУК Іван Тарасович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Алгоритми захисту конфіденційності та автентичності Інтернет-речей з використанням стеганографії / Algorithms for protection of confidentiality and authenticity of Internet of things using steganography
керівник роботи к.т.н., доцент Якименко І. З.
затверджені наказом по університету від 12 грудня 2023 року № 753

2. Строк подання студентом закінченої кваліфікаційної роботи 12 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- Проаналізувати базові концепції та принципи роботи IoT систем;
- Дослідити існуючі методи забезпечення безпеки в IoT;
- Проаналізувати сучасні протоколи та стандарти безпеки IoT;
- Дослідити методи стеганографічного захисту інформації;
- Проаналізувати стеганографічні методи забезпечення конфіденційності;
- Дослідити стеганографічні методи автентифікації;
- Проаналізувати методи інтеграції стеганографії в IoT;
- Розробити стеганографічний метод захисту даних;
- Впровадити стеганографічний метод в IoT систему;

5. Перелік графічного матеріалу у роботі
– Алгоритм передачі даних в стеганографії;
– Схема роботи DWT;

– Схема роботи розширення спектру;
– Схема роботи методу LSB;

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.2023 р. – 03.2024 р.	
2	Аналіз алгоритмів стеганографії в IOT.	03.2024 р. – 06.2024 р.	
3	Розробка методів захисту даних Інтернет-речей.	06.2024 р. – 11.2024 р.	

Студент _____ Сапіжук І. Т.
(підпис)

Керівник роботи _____ к.т.н., доцент Якименко І. З.
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритми захисту конфіденційності та автентичності Інтернет-речей з використанням стеганографії » на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека та захист інформації» освітньо-професійної програми «Кібербезпека» написана обсягом 110 сторінки і містить 26 ілюстрацій, 12 таблиць, 3 додатки та 53 джерел за переліком посилань.

Метою дослідження є аналіз алгоритмів захисту конфіденційності та автентичності даних в IoT з використанням стеганографічних методів. Дослідження спрямоване на огляд ефективних та надійних механізмів безпеки, які враховують специфіку архітектури та протоколів IoT.

Було успішно реалізовано метод стеганографії на основі дискретного вейвлет-перетворення (DWT) для роботи із зображеннями. Також реалізовано додатковий метод для роботи з текстовими даними, що розширює можливості системи. Обидва методи включають механізми кодування та декодування з використанням випадкових позицій для підвищення безпеки. Створено повноцінне середовище для роботи IoT пристроїв. Розроблена архітектура базується на MQTT протоколі, що забезпечує надійну асинхронну комунікацію між компонентами системи. Реалізовано механізми реєстрації пристроїв, обробки команд та даних, а також систему відновлення після збоїв. Створена структура топіків дозволяє ефективно маршрутизувати різні типи повідомлень між компонентами системи.

Ключові слова: Стеганографія, IOT, LSB, DWT, MQTT.

ABSTRACT

The qualification work on the topic "Algorithms for protecting the confidentiality and authenticity of Internet of Things using steganography" for obtaining the educational degree "Master" in specialty 125 "Cybersecurity and information protection" of the educational and professional program "Cybersecurity" is written in the volume of 110 pages and contains 26 illustrations, 12 tables, 3 appendices and 53 sources according to the list of references.

The purpose of the study is to analyze algorithms for protecting the confidentiality and authenticity of data in IoT using steganographic methods. The study is aimed at reviewing effective and reliable security mechanisms that take into account the specifics of the IoT architecture and protocols.

A steganography method based on the discrete wavelet transform (DWT) was successfully implemented for working with images. An additional method for working with text data was also implemented, which expands the capabilities of the system. Both methods include encoding and decoding mechanisms using random positions to increase security. A full-fledged environment for IoT devices has been created. The developed architecture is based on the MQTT protocol, which provides reliable asynchronous communication between system components. Mechanisms for device registration, command and data processing, as well as a disaster recovery system have been implemented. The created topic structure allows for efficient routing of various types of messages between system components.

Keywords: Steganography, IOT, LSB, DWT, MQTT.

ЗМІСТ

СПИСОК ВИКОРИСТАНИХ СКОРОЧЕНЬ

IOT (IoT) — Internet of Things.

IERC — European Research Cluster on the Internet of Things.

ITU — International Telecommunication Union.

AES — Advanced Encryption Standard.

CLEFIA — Block Cipher Developed by Sony.

ECC — Elliptic Curve Cryptography.

RSA — Rivest–Shamir–Adleman.

ECMQV — Elliptic Curve Menezes-Qu-Vanstone.

POF — Proof of Functionality.

PUF — Physical Unclonable Function.

ABAC — Attribute-Based Access Control.

IDS — Intrusion Detection System.

IPS — Intrusion Prevention System.

NTRU — Nth-degree Truncated Polynomial Ring Unit.

CSMA — Carrier Sense Multiple Access.

LwM2M — Lightweight Machine to Machine.

DDS — Data Distribution Service.

ВСТУП

Актуальність теми. Актуальність теми цього дослідження зумовлена стрімким розвитком технологій Інтернету речей (IoT) і зростаючою потребою в захисті конфіденційності та достовірності даних, переданих між IoT-пристроями. Забезпечення безпеки і конфіденційності в системі IoT стикається з новими проблемами зі збільшенням кількості під'єднаних пристроїв та обсягу даних, які вони генерують. Традиційні криптографічні методи, як-от шифрування і цифрові підписи, не завжди підходять для IoT-додатків через обмежені обчислювальні ресурси та енергоспоживання пристроїв. Перспективним напрямком у забезпеченні безпеки IoT є використання методів стеганографії. Стеганографія дає змогу приховувати конфіденційну інформацію серед інших типів даних, таких як зображення, голос і мережевий трафік, не привертаючи уваги зловмисників. Поєднання стеганографії та криптографії забезпечує додатковий рівень захисту, який ускладнює виявлення та розшифрування прихованих даних. Архітектура та специфіка протоколів є актуальними питаннями.

Зростаючий інтерес наукового центру та індустрії IoT до розроблення надійних та ефективних механізмів безпеки зумовлює важливість досліджень з цієї тематики. Дослідження в цій галузі спрямовані на подолання технічних проблем, пов'язаних з обмеженими ресурсами IoT-пристроїв, забезпечення масштабованості та сумісності рішень безпеки з різними протоколами і стандартами IoT. Розвиток стеганографічних технологій та їхня адаптація до вимог IoT відкриває нові можливості для захисту конфіденційності та автентичності даних без серйозної шкоди для продуктивності та енергоспоживання пристроїв.

Мета і завдання дослідження. Метою дослідження є аналіз алгоритмів захисту конфіденційності та автентичності даних в IoT з використанням стеганографічних методів. Дослідження спрямоване на огляд ефективних та

надійних механізмів безпеки, які враховують специфіку архітектури та протоколів IoT.

Зазначена мета передбачає вирішення таких завдань:

- Проаналізувати базові концепції та принципи роботи IoT систем.
- Дослідити існуючі методи забезпечення безпеки в IoT.
- Проаналізувати сучасні протоколи та стандарти безпеки IoT.
- Дослідити методи стеганографічного захисту інформації.
- Проаналізувати стеганографічні методи забезпечення конфіденційності.
- Дослідити стеганографічні методи автентифікації.
- Проаналізувати методи інтеграції стеганографії в IoT.
- Розробити стеганографічний метод захисту даних.
- Створити тестове IoT середовище.
- Впровадити стеганографічний метод в IoT систему.

Об'єктом дослідження є процеси захисту конфіденційності та автентичності даних в середовищі Інтернету речей.

Предметом дослідження є алгоритми захисту конфіденційності та автентичності даних в IoT з використанням стеганографічних методів.

Методи дослідження: методи стеганографії, криптографії, аналіз алгоритмів, моделювання, експериментальні дослідження.

Наукова новизна: Наукова новизна дослідження полягає у розробці алгоритмів захисту конфіденційності та автентичності даних в IoT з використанням стеганографічних методів та їх реалізації. Запропоновано оригінальні підходи до стеганографічного приховування даних та забезпечення автентичності, що дозволяють підвищити рівень безпеки та приватності в екосистемі IoT без суттєвого впливу на продуктивність та енергоспоживання пристроїв.

Практичне значення: Практичне значення отриманих результатів дослідження полягає у відтворенні алгоритмів захисту конфіденційності та автентичності даних в IoT з використанням стеганографії, які можуть бути безпосередньо впроваджені у розробку систем та пристроїв IoT. Запропоновані

алгоритми та рекомендації щодо їх використання мають значний потенціал для підвищення рівня безпеки та приватності в різних сферах застосування IoT, таких як промисловість, охорона здоров'я, розумні будинки та міста.

Результати дослідження: Результати даного дослідження сприяють розвитку методів та алгоритмів захисту конфіденційності та автентичності даних в IoT з використанням стеганографії.

Публікації та апробація до магістерської роботи.

1. І. Сапіжук, В. Драпак. Використання блокчейну для забезпечення автентичності та цілісності даних в інтернеті речей. Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. С. 11-18

2. І. Сапіжук. Б. Бараннік, П. Билень. Застосування стеганографії для забезпечення коніфденційності та автентичності даних в інтернеті речей. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. С. 140-142

1 ТЕОРЕТИЧНІ ОСНОВИ IOT

1.1. Основні поняття та концепції IOT

Інтернет речей (IoT) - це концепція, що описує мережу фізичних об'єктів, пристроїв, транспортних засобів, будівель та інших елементів, які мають вбудовані електронні компоненти, програмне забезпечення, датчики та мережеві з'єднання, що дають змогу цим об'єктам збирати й обмінюватись даними [1] IoT є розширенням Інтернету та “забезпечує під'єднання і взаємодію широкого спектра повсякденних об'єктів, а не лише традиційних обчислювальних пристроїв” [2]. Ця концепція революціонізує способи взаємодії фізичних об'єктів із цифровим світом, відкриваючи нові можливості для оптимізації процесів, підвищення ефективності та створення інноваційних послуг.

Існує декілька визначень IoT, які підкреслюють різні аспекти цієї концепції. Наприклад, Міжнародний союз електрозв'язку (ITU) визначає IoT як “глобальну інфраструктуру для інформаційного суспільства, що забезпечує розширені послуги шляхом з'єднання (фізичних і віртуальних) речей на основі існуючих та розвиваючих інформаційно-комунікаційних технологій” [3]. Інше визначення, запропоноване Європейською дослідницькою кластерною ініціативою IoT (IERC), описує IoT як “динамічну глобальну мережеву інфраструктуру з можливостями самоналаштування, засновану на стандартних і сумісних протоколах зв'язку, де фізичні та віртуальні “речі” мають ідентифікатори, фізичні атрибути і віртуальні особистості, використовують інтелектуальні інтерфейси та безшовно інтегровані в інформаційну мережу” [4].

Архітектури IoT зазвичай складаються з трьох основних рівнів: рівня сприйняття, мережевого рівня та рівня додатків [5]. Рівень сприйняття охоплює різні датчики, виконавчі механізми та інші пристрої, які збирають дані про навколишнє середовище і взаємодіють з ним. Мережевий рівень відповідає за передачу та обробку зібраних даних і забезпечує зв'язок між пристроями та хмарними сервісами. “Прикладний рівень містить програмне забезпечення та інтерфейси для взаємодії користувачів з IoT-пристроями та отримання корисної

інформації” [6]. Така багаторівнева архітектура забезпечує ефективне функціонування систем IoT і дає змогу збирати, передавати та обробляти дані з різних джерел.

Основними компонентами IoT є “датчики та виконавчі пристрої, мережеві протоколи, шлюзи та хмарні платформи” [7]. Датчики та виконавчі механізми - це пристрої, які збирають дані про навколишнє середовище та виконують певні дії. Мережеві протоколи, як-от Wi-Fi, Bluetooth і ZigBee, забезпечують передачу даних між пристроями IoT. Шлюзи - це пристрої, що забезпечують зв'язок між датчиками, виконавчими пристроями та хмарними сервісами. Хмарні платформи - це сервіси для зберігання, оброблення та аналізу даних, отриманих від IoT-пристроїв [8]. Разом ці компоненти утворюють єдину екосистему IoT, яка забезпечує ефективний збір, передачу та обробку даних.

Додатки IoT охоплюють широкий спектр галузей, «включно з промисловістю, охороною здоров'я, транспортом, розумними будинками та містами» [9]. У промисловості IoT дає змогу контролювати виробничі процеси, оптимізувати використання ресурсів і підвищувати ефективність виробництва. В охороні здоров'я IoT використовується для віддаленого моніторингу стану пацієнтів, управління прийомом ліків і підвищення якості медичного обслуговування. У транспортній галузі IoT використовується для управління рухом, моніторингу стану транспортних засобів та оптимізації логістичних процесів. Розумні будинки і міста використовують IoT для автоматизації побутових процесів, управління енергоспоживанням і підвищення безпеки [10]. Ці приклади ілюструють величезний потенціал IoT для поліпшення багатьох аспектів нашого життя.

Переваги IoT включають в себе “підвищення ефективності та продуктивності, зниження витрат, поліпшення якості обслуговування і створення нових бізнес-моделей” [11]. Завдяки IoT компанії можуть отримувати більше даних про свої продукти і послуги, що дає їм змогу ухвалювати більш обґрунтовані рішення і покращувати якість обслуговування клієнтів. Це дає їм змогу ухвалювати більш обґрунтовані рішення і покращувати якість

обслуговування клієнтів. IoT також сприяє “автоматизації процесів, знижуючи потребу в ручній праці та зводячи до мінімуму людські помилки” [12]. Крім того, IoT відкриває нові можливості для створення інноваційних продуктів і послуг, які можуть зробити революцію в цілих галузях і поліпшити якість життя.

1.2. Огляд існуючих алгоритмів захисту конфіденційності та автентичності в IoT

Захист конфіденційності та автентичності в системах Інтернету речей (IoT) реалізується за допомогою спеціальних алгоритмів і протоколів безпеки. що впливає на вибір відповідних криптографічних методів [13].

Для забезпечення конфіденційності в IoT широко використовуються алгоритми симетричного шифрування. основним вибором залишається Advanced Encryption Standard (AES) завдяки оптимальному співвідношенню безпеки та продуктивності. AES-128 є найбільш часто використовуваним і забезпечує достатній рівень захист, за менших обчислювальних витрат, ніж AES-256 [14]. AES-218 спеціально розроблений для обмежених пристроїв, як-от PRESENT і CLEFIA, має меншу продуктивність, але потребує менше ресурсів.

Алгоритм Advanced Encryption Standard (AES) - це симетричний блоковий шифр, стандартизований Національним інститутом стандартів і технологій США (NIST). AES обробляє дані в 128-бітних блоках і підтримує 128, 192 і 256-бітові ключі. Структура алгоритму заснована на мережі SP, що поєднує підстановку і перестановку. Процес шифрування складається з раундів перетворення, кількість яких залежить від довжини ключа: 10 раундів для AES-128, 12 раундів для AES-192 і 14 раундів для AES-256 [15].

В IoT-системах найчастіше використовується AES-128 завдяки оптимальному співвідношенню безпеки та обчислювальних витрат. Кожен раунд включає операції SubBytes (нелінійна заміна байтів), ShiftRows (циклічний зсув рядків), MixColumns (змішування стовпчиків) і AddRoundKey (раундове додавання ключів). апаратна реалізація AES-128 вимагає близько 2400 логічних

вентилів, тактова частота - 100 МГц, забезпечуючи пропускну здатність до 2,6 Гбіт/с [16].

PRESENT - полегшений блоковий шифр, розроблений спеціально для пристроїв з обмеженими можливостями. Алгоритм працює з 64-бітними блоками даних і підтримує 80- або 128-бітові ключі. В основі структури лежить 31 раунд перетворення з використанням простої мережі SP. Кожен раунд містить додаткову операцію з раундовим ключем, шар підстановки з 4-бітним S-блоком і перетворення з перестановкою бітів. Апаратна реалізація PRESENT вимагає всього 1570 логічних вентилів, що робить його ідеальним для IoT-пристроїв з низьким ресурсом [17].

CLEFIA - блоковий шифр, розроблений корпорацією Sony. Алгоритм підтримує розмір блоку 128 біт і довжину ключа 128, 192 і 256 біт. Архітектура заснована на узагальненій мережі Фейстеля з чотирма гілками. Кількість раундів становить 18, 22 і 26 для відповідних довжин ключів. CLEFIA досягає високої криптографічної стійкості при помірних апаратних вимогах, що становлять близько 2500 логічних вентилів [18], використовуючи два типи 8-бітних S-блоків і дві дифузійні матриці. Детальніше про ці методи описано в таблицях 1.1, 1.2 і 1.3.

Таблиця 1.1 – Порівняльні характеристики алгоритмів шифрування

Алгоритм	Розмір блоку (біт)	Розмір ключа (біт)	Кількість раундів	Логічних вентилів
AES-128	128	128	10	24 000
PRESENT-80	64	80	31	1 570
CLEFIA-128	128	128	18	2 500

Серед представлених алгоритмів, розмір блоку (біт) - це кількість біт

даних, які алгоритм обробляє за одну операцію шифрування, розмір ключа (біт) визначає рівень криптографічної стійкості, кількість раундів показує скільки разів повторюються криптографічні перетворення, кількість логічних вентилів відображає складність апаратної реалізації.

Таблиця 1.2 – Показники продуктивності на 8-бітному мікроконтролері

Алгоритм	Розмір коду (байт)	Оперативна пам'ять (байт)	Такти/байт	Енергія/байт (нДж)
AES-128	1504	1764	161	4.2
PRESENT-80	936	961	594	1.7
CLEFIA-128	2104	2163	451	2.5

В свою чергу, криптографічні властивості алгоритмів представлені в таблиці 1.3.

Таблиця 1.3 – Криптографічні властивості алгоритмів

Алгоритм	Диференціальний криптоаналіз	Лінійний криптоаналіз	Атаки на реалізацію
AES-128	>2^128	>2^128	Вразливий до атак через кеш
PRESENT-80	>2^80	>2^80	Стійкий
CLEFIA-128	>2^128	>2^128	Помірно стійкий

Асиметрична криптографія в IoT здебільшого представлена криптографічними алгоритмами на еліптичних кривих (ECC): порівняно з RSA, ECC забезпечує той самий рівень безпеки за меншого розміру ключа. Найпоширенішими є крива NIST P-256 і крива Curve25519.

Остання продемонструвала кращу продуктивність і стійкість до атак через сторонні канали [19]. Для створення цифрових підписів використовуються алгоритми ECDSA і EdDSA, засновані на властивостях еліптичних кривих.

Крива NIST P-256, стандартизована Національним інститутом стандартів і технологій США, описується рівнянням $y^2 = x^3 - 3x + b$ над простим полем, де b - спеціально підібраний параметр.

Крива Curve25519, розроблена Деніелом Бернштейном, базується на кривій Монтгомері $y^2 = x^3 + 486662x^2 +$ x над полем 2^255 - 19 та демонструє кращу продуктивність завдяки ефективнішим формулам для групових операцій [20].

Алгоритм цифрового підпису ECDSA (Elliptic Curve Digital Signature Algorithm) генерує підписи розміром 512 біт для кривої P-256.

EdDSA (Edwards-curve Digital Signature Algorithm) використовує скручені криві Едвардса та забезпечує підписи розміром 512 біт з вищою швидкодією та кращою захищеністю від помилок реалізації [21].

Хеш-функції в свою чергу гарантують цілісність даних у системах IoT. Стандартними варіантами є SHA-2 і SHA-3, але для обмежених пристроїв було розроблено і полегшені альтернативи: BLAKE2s оптимізовано для 32-бітних платформ, що зберігає криптографічну стійкість і високу [22]. Функція SHAKE на основі SHA-3 дає змогу гнучко обирати довжину вихідного значення.

SHA-2 базується на структурі Меркля-Дамгарда, де повідомлення розбивається на блоки фіксованого розміру. Кожен блок послідовно обробляється з використанням функції стиснення, а результат попереднього блоку використовується як вхід для обробки наступного. SHA-2 активно використовує операції додавання за модулем 2^32, побітові логічні функції та

циклічні зсуви для формування хешу.

SHA-3 використовує зовсім інший підхід - губкову конструкцію Keccak, яка має внутрішній стан розміром 1600 біт. Робота алгоритму складається з двох основних фаз: поглинання вхідних даних та вижимання вихідних. SHA-3 використовує 5 нелінійних функцій перемішування (θ, ρ, π, χ, ι), що забезпечує кращу стійкість до атак на основі довжини повідомлення.

Функція SHAKE дозволяє отримувати хеш-значення довільної довжини завдяки розширювальному виходу губкової конструкції [23]. Також всі дані порівняння представлені в таблицях 1.4 та 1.5 відповідно до того, як було це представлено в симетричних системах криптування.

Тобто, в таблиці 1.4 представлено порівняння вищезгаданих асиметричних криптосистем на основі їх розміру ключа, розміру підпису та операцій в секунду з виділенням пам’яті.

Таблиця 1.4 – Порівняння асиметричних криптосистем

Алгоритм	Розмір ключа (біт)	Розмір підпису (біт)	Операцій/с	Пам'ять (КБ)
RSA-3072	3072	3072	1.2	384
NIST P-256	256	512	8.4	32

В таблиці 1.5. представлені характеристики хеш-функцій з їх основними параметрами.

Таблиця 1.5 – Характеристики хеш-функцій

Алгоритм	Розмір блоку (біт)	Розмір виходу (біт)	Б/с	Пам'ять (байт)
SHA-256	512	256	26	278
SHA-3	1088	256	08	384

Продовження таблиці 1.5

BLAKE2s	512	256	95	256
SHAKE128	1088	Змінний	47	384

Протоколи аутентифікації IoT повинні враховувати обмеження пристроїв і специфікації мережевої взаємодії. Полегшений протокол аутентифікації OAuth 2.0 адаптовано до IoT за допомогою профілю Device Flow. Цей протокол дозволяє пристроям з обмеженим інтерфейсом отримувати токени доступу [24]. DTLS (Datagram Transport Layer Security) забезпечує безпечний транспортний рівень для UDP-комунікацій, характерних для IoT.

Для групової аутентифікації розроблено спеціальну схему, засновану на еліптичних кривих. Це дає змогу пристроям підтверджувати приналежність до довіреної групи без необхідності встановлювати індивідуальні безпечні з'єднання [25]. Протоколи групового підпису гарантують анонімність окремих пристроїв і водночас зберігають можливість відкликання скомпрометованих членів.

Управління ключами в IoT-системах здійснюється за допомогою полегшених протоколів: протокол ECMQV (Elliptic Curve Menezes-Qu-Vanstone) забезпечує узгодження ключів з автентифікацією на основі еліптичної кривої. Протокол SPAKE2 дає змогу використовувати простий пароль, що може бути використаний для встановлення безпечного з'єднання.

В свою чергу алгоритми Physical Unclonable Functions (PUF) використовуються для захисту від атак на фізичному рівні. PUF використовують унікальні фізичні характеристики чипа для генерації криптографічних ключів. Поширеними реалізаціями є PUF з кільцевим осцилятором і PUF з арбітром, які забезпечують достатню ентропію за мінімальних апаратних витрат [26].

Конфіденційність на рівні даних забезпечується гомоморфним шифруванням і схемами поділу секретів. Полегшена версія часткового

гомоморфного шифрування дозволяє виконувати обмежений набір операцій над зашифрованими даними [27]. Схема поділу секретів Шаміра була адаптована для розподіленого зберігання критично важливих даних між IoT-пристроями.

Технологія блокчейн адаптується для використання в системах IoT: полегшені протоколи консенсусу, як-от Proof of Stake і Delegated Proof of Stake, потребують менше обчислювальних ресурсів, ніж класичний Proof of Work [28]. Смарт-контракти на основі блокчейна забезпечують автоматизований контроль доступу та аудит транзакцій.

Протоколи безпечної маршрутизації в мережах IoT використовують криптографічні примітиви для захисту від атак на рівні мережі. Протокол безпечної маршрутизації (SRP) забезпечує автентифікацію маршрутної інформації та захист від атак “червоточин” [29]. Географічна безпечна маршрутизація (GSPR) використовує геолокаційні дані для перевірки маршрутів.

Механізми управління доступом в IoT засновані на підходах, заснованих на атрибутах і ролях. Управління доступом на основі атрибутів (ABAC) дає змогу гнучко визначати політики доступу на основі атрибутів пристрою і користувача [30]. Контроль доступу на основі можливостей використовує токени можливостей для делегування прав доступу між пристроями. Системи виявлення і запобігання вторгнень (IDS/IPS) в IoT використовують легкі алгоритми машинного навчання. Методи виявлення аномалій засновані на статистичних моделях і малорозмірних нейронних мережах [31]. Розподілені IDS дають змогу розподіляти навантаження між пристроями і забезпечують масштабованість.

Протоколи безпечної агрегації даних дають змогу об'єднувати інформацію від декількох датчиків IoT без розкриття окремих значень. Агрегація на основі гомоморфного шифрування забезпечує конфіденційність під час обчислення статистичних показників [32]. Агрегація даних зі збереженням конфіденційності захищає окремі вузли від компрометації за допомогою схем поділу секретів. Квантовостійкі алгоритми поступово впроваджуються в системи IoT. Ґратчаста криптографія, зокрема схеми на основі NTRU і LWE, забезпечують постквантову безпеку за прийнятних обчислювальних витрат [33]. Криптографія Ring-LWE

оптимізована для обмеженого числа пристроїв і має високу продуктивність.

Протоколи Secure Boot і Software Update використовують ланцюжки довіри та цифрові підписи. Secure Boot забезпечує цілісність прошивки під час завантаження [34], а Over-the-Air (OTA) оновлення реалізують диференціальні оновлення та механізми аварійного відновлення.

1.3. Протоколи безпеки та стандарти захисту IoT пристроїв

Варто також розглянути протоколи безпеки для захисту IoT пристроїв, серед яких є стандартні - базові протоколи, так і нестандартні, які почали використовувати відносно недавно або через їх кращу реалізацію.

Першим таким є Message Queuing Telemetry Transport (MQTT) - це базовий протокол передавання даних для IoT-пристроїв, що працюють за принципом публікації/підписки. “Протокол забезпечує три рівні якості обслуговування (QoS): QoS 0 гарантує максимум одну доставку, QoS 1 - мінімум одну доставку, а QoS 2 - рівно одну доставку. MQTT використовує TLS для шифрування і підтримує автентифікацію за ім'ям користувача/паролем або сертифікатами X.509.” [35]. В таблиці 1.6 наведено характеристики рівнів QoS.

Таблиця 1.6 – Характеристики рівнів QoS в протоколі MQTT

Характеристика	QoS 0 (At most once)	QoS 1 (At least once)	QoS 2 (Exactly once)
Гарантія доставки	Не гарантована	Гарантована з можливістю дублювання	Гарантована одноразова
Підтвердження	Немає	PUBACK	PUBREC, PUBREL, PUBCOMP
Накладні витрати	Мінімальні	Середні	Максимальні

Продовження таблиці 1.6

Швидкість передачі	Найвища	Середня	Найнижча
Використання пам'яті	Мінімальне	Середнє	Максимальне
Типові застосування	Телеметрія сенсорів	Логування подій	Фінансові транзакції

Протокол обмежених додатків (Constrained Application Protocol, CoAP) розроблено як полегшений аналог HTTP для обмежених пристроїв. Протокол реалізує архітектуру REST і працює поверх UDP, що робить його ефективним для пристроїв з обмеженими ресурсами.

CoAP забезпечує безпеку за DTLS і підтримує режими попереднього поділу ключів (PSK), необробленого відкритого ключа (RPK) і сертифіката. Протокол також включає механізми виявлення ресурсів і моніторингу стану. В таблиці 1.7 наведено порівняльну характеристику режимів безпеки.

Таблиця 1.7 – Характеристики режимів CoAP

Характеристика	PSK режим	RPK режим	Сертифікатний режим
Тип ключа	Симетричний	Асиметричний	Асиметричний + PKI
Розмір ключа	128-256 біт	2048-4096 біт	2048-4096 біт
Вимоги до пам'яті	Низькі	Середні	Високі
Обчислювальні витрати	Низькі	Середні	Високі

Продовження таблиці 1.7

Масштабованість	Обмежена	Середня	Висока
Управління ключами	Ручне	Напівавтоматичне	Автоматичне
Типове застосування	Сенсори	Актуатори	Шлюзи
Підтримка відкликання	Немає	Обмежена	Повна

IEEE 802.15.4 визначає стандарт малопотужної бездротової персональної мережі (LR-WPAN). Цей стандарт містить специфікації безпеки для MAC-рівня з використанням симетричних шифрів AES-CCM* зі 128-бітними ключами. Цей механізм безпеки забезпечує захист від конфіденційності даних, аутентифікацію джерела та відтворення повідомлень. Стандарт підтримує “чотири типи ключів: головний, мережевий, груповий і парний” [36].

Служба безпеки ZigBee на базі IEEE 802.15.4 розширює можливості безпеки за рахунок додаткових механізмів на рівні мережі та додатків. Протокол використовує центри довіри для централізованого управління безпекою, включно з розподілом ключів і контролем доступу до мережі. ZigBee підтримує оновлення ключів безпеки і механізми виключення скомпрометованих пристроїв з мережі. На рисунку 1.1 зображено процес роботи ZigBee.

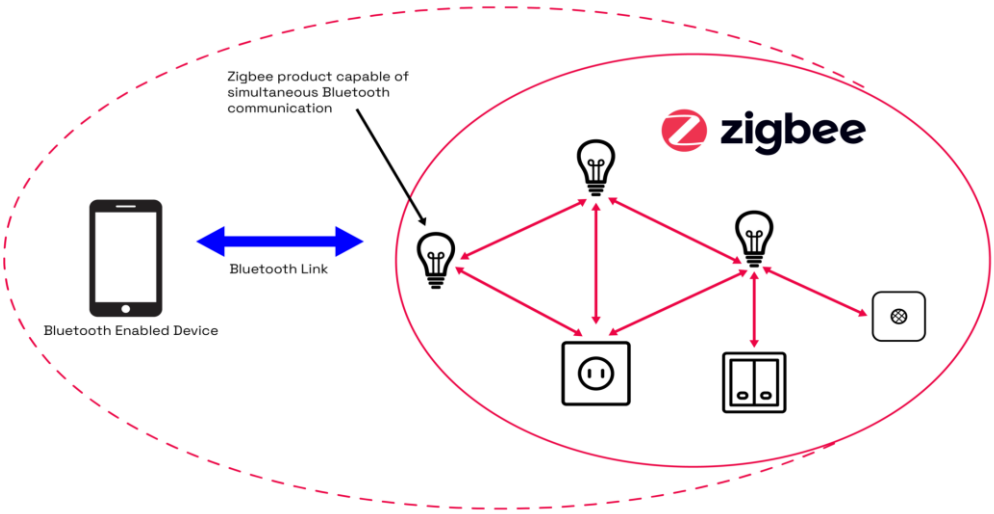

Рисунок 1.1 – ZigBee

Якщо коротко про ZigBee. ZigBee є бездротовим стандартом комунікації, спеціально розробленим для Internet of Things та автоматизації, що забезпечує низьке енергоспоживання та надійну передачу даних у mesh-мережах. Стандарт розроблений та підтримується ZigBee Alliance, нині відомим як Connectivity Standards Alliance.

Мережа ZigBee включає три типи пристроїв: координатор, маршрутизатори та кінцеві пристрої. Координатор є обов'язковим компонентом мережі, який відповідає за її формування, керування та зберігання інформації про безпеку. Маршрутизатори розширюють покриття мережі та забезпечують альтернативні шляхи для передачі даних. Кінцеві пристрої взаємодіють з мережею через батьківські вузли та можуть переходити в режим сну для економії енергії.

Протокол працює в діапазоні 2.4 ГГц та забезпечує швидкість передачі даних до 250 кбіт/с. ZigBee використовує CSMA/CA механізм для доступу до каналу зв'язку, що мінімізує колізії при передачі даних. Технологія підтримує самовідновлювану mesh-топологію, де кожен маршрутизатор може встановлювати з'єднання з іншими маршрутизаторами, забезпечуючи множинні шляхи для передачі даних.

Є ще протокол LoRaWAN, який реалізує комплексну систему безпеки для мереж дальнього радіусу дії. Кожен пристрій має унікальний 128-бітний ключ застосунку (AppKey), на основі якого генеруються два сеансових ключі: мережевий сеансовий ключ (NwkSKey) для автентифікації та сеансовий ключ застосунку (AppSKey) для шифрування даних. Протокол використовує “лічильник повідомлень для захисту від повторного відтворення та підтримує механізми безпечних мережевих з'єднань” [37].

Open Connectivity Foundation (OCF) визначає комплексну архітектуру безпеки для екосистеми IoT. Стандарт включає механізми автентифікації на основі DTLS/TLS, які підтримують різні методи, включно з сертифікатами X.509, спільними ключами та паролями. OCF “реалізує контроль доступу на основі ролей (RBAC) і контроль доступу на основі атрибутів (ABAC) для

забезпечення гнучкого налаштування політики безпеки. конфігурації політики безпеки. Стандарт також визначає безпечну конфігурацію пристрою та процедури оновлення програмного забезпечення” [38].

Протокол Thread забезпечує наскрізне шифрування для домашніх мереж IoT. Безпека реалізована на трьох рівнях: MAC-рівень використовує механізми IEEE 802.15.4, мережевий рівень забезпечує шифрування автентифікації та інформації про маршрутизацію, а прикладний рівень підтримує додаткове шифрування даних застосунку. Потік використовує “ ”комісарів” для безпечного додавання нових пристроїв і підтримує автоматичне відновлення мережі в разі відмови вузла” [39].

Що до стандартів. Стандартизація IoT відіграє ключову роль у забезпеченні сумісності, безпеки та надійності пристроїв і систем Існує кілька великих міжнародних організацій і консорціумів, що розробляють і підтримують стандарти для IoT, включно з ISO/IEC, IEEE, ETSI, ITU-T та галузевими альянсами. До них належать.

Стандартизація IoT систем спрямована на вирішення таких ключових завдань: забезпечення функціональної сумісності між пристроями різних виробників, встановлення єдиних вимог до безпеки та конфіденційності, визначення протоколів обміну даними, специфікація форматів даних та інтерфейсів взаємодії. Важливим аспектом є також стандартизація процесів тестування та сертифікації IoT рішень.

Наприклад, стандарт oneM2M визначає сервісний рівень безпеки для IoT-платформ. oneM2M підтримує різні рівні безпеки та забезпечує узгоджену модель довіри в різних доменах. Стандарт також визначає процедури безпечного завантаження та оновлення програмного забезпечення.

Стандарт Industrial Internet Security Framework (IISF) визначає комплексний підхід до забезпечення безпеки промислових IoT-систем. Рамки ґрунтуються на концепції глибокої оборони і включають рекомендації з безпеки кінцевих точок, комунікацій, моніторингу та управління конфігурацією.

Стандарт IEC 62443 встановлює вимоги безпеки для систем промислової

автоматизації та управління. Стандарт визначає системний підхід до забезпечення безпеки, включно з оцінкою ризиків, зонуванням мережі, контролем доступу та моніторингом безпеки. IEC 62443 також “встановлює вимоги до життєвого циклу розроблення безпечних продуктів і систем” [40].

Протокол Lightweight Machine to Machine (LwM2M), розроблений Open Mobile Alliance, забезпечує ефективне управління пристроями IoT. Протокол підтримує безпечну конфігурацію пристроїв, оновлення програмного забезпечення та моніторинг стану. “LwM2M захищає зв'язок за допомогою DTLS і підтримує різні механізми аутентифікації” [41].

Промисловий протокол OPC UA (Open Platform Communications Unified Architecture) забезпечує безпечну взаємодію між промисловими IoT пристроями та системами управління. Протокол реалізує багаторівневу модель безпеки, що включає автентифікацію на основі сертифікатів X.509, шифрування повідомлень та контроль доступу на рівні об'єктів. “OPC UA підтримує різні профілі безпеки для адаптації до різних вимог та обмежень” [42].

Bluetooth Low Energy (BLE) Security включає механізми парування пристроїв, генерації ключів та шифрування даних. Протокол підтримує чотири режими безпеки та сім рівнів захисту, що дозволяють балансувати між безпекою та енергоспоживанням. “BLE використовує алгоритм AES-CCM для шифрування та механізми захисту від атак типу man-in-the-middle” [43].

Протокол Wirelesshart, призначений для промислових бездротових сенсорних мереж, реалізує комплексні механізми безпеки. Кожен пристрій має унікальний Join Key для початкового приєднання до мережі, після чого використовуються Network Key для мережевої автентифікації та Session Keys для захисту даних. Протокол підтримує “ротацію ключів та blacklist для скомпрометованих пристроїв” [44].

DDS (Data Distribution Service) Security визначає механізми захисту для систем реального часу. Стандарт включає контроль доступу на основі домену безпеки, автентифікацію на основі сертифікатів або симетричних ключів, шифрування даних та захист цілісності повідомлень. “DDS підтримує різні

криптографічні алгоритми та дозволяє налаштовувати політики безпеки для різних потоків даних” [45].

ISA100.11a стандарт безпеки для промислових бездротових мереж визначає багаторівневу архітектуру захисту. Стандарт використовує симетричне шифрування AES-128 для захисту даних, “підтримує різні типи ключів (Join Key, Network Key, Session Keys) та включає механізми управління ключами. ISA100.11a також визначає процедури безпечного приєднання до мережі та оновлення програмного забезпечення” [46].

1.4. Стеганографія як метод захисту інформації

Стеганографія - це наука про те, як приховати факт передавання інформації шляхом вбудовування інформації в цифрові об'єкти, які зазвичай не помічають. На відміну від криптографії, яка робить повідомлення нечитабельними, “стеганографія приховує сам факт існування секретного повідомлення” [47].

Сучасна цифрова стеганографія використовує специфіку подання інформації в комп'ютерних файлах, мережевих протоколах та інших цифрових об'єктах. Основними носіями, використовуваними для приховування інформації, є зображення, аудіо- та відеофайли і мережевий трафік. Найпоширенішим методом є модифікація найменш значущого біта (LSB - Least Significant Bit), заснована на тому, що “зміна останнього біта байта кольору пікселя призводить до настільки малих змін, що вони непомітні для людського ока” [48]. Методи вбудовування в аудіофайли використовують особливості людського слуху та ґрунтуються на явищі частотного та часового маскування. Найефективнішими методами є “спектральне розширення і фазове кодування, які впроваджують інформацію шляхом зміни фазових складових вихідного сигналу” [49].

У випадку з відеофайлами використовуються як просторові методи, що приховують окремі кадри, так і тимчасові, засновані на послідовності кадрів. Особливо ефективним є приховування інформації шляхом “вбудовування додаткових даних у коефіцієнти дискретного косинусного перетворення в процесі стиснення відео, що забезпечує високу стійкість до компресії” [50].

Мережева стеганографія відкриває можливість приховування даних завдяки використанню особливостей протоколів передачі.

Поширені методи включають “модифікацію невикористовуваних полів заголовка TCP/IP або маніпулювання часовим інтервалом між пакетами, що дає змогу створювати приховані канали даних у мережевому трафіку”. [51].

Важливим аспектом стеганографії є забезпечення стійкості до методів стегоаналізу. Статистичний стегоаналіз виявляє приховані повідомлення, порушуючи природні статистичні властивості носія, а візуальний стегоаналіз використовує методи посилення змін для виявлення вбудованих артефактів. Для підвищення стійкості використовуються адаптивні методи вбудовування, що враховують характеристики конкретного носія.

У випадку із зображеннями інформація здебільшого впроваджується в текстуровані ділянки, де зміни менш помітні, і використовується попереднє опрацювання повідомлень, включно з шифруванням і завадостійким кодуванням.

Стеганографія в контексті IoT особливо важлива для безпечного передавання критично важливої інформації. Стеганографія в системах IoT особливо важлива у зв'язку з особливими вимогами до безпеки та обмеженими ресурсами пристроїв.

Основними варіантами використання стеганографії в IoT є прихована передача конфігураційних даних, ключів шифрування, команд управління і критично важливої телеметрії.

Для IoT-пристроїв було розроблено спеціальні полегшені методи стеганографії, що враховують обмеження обчислювальної потужності та пам'яті.

Модифіковані LSB-методи для IoT використовують адаптивний вибір бітів для вбудовування, знижуючи обчислювальне навантаження при збереженні достатнього рівня стійкості.

На рисунку 1.2 зображено як використовується стеганографія для картинок в IoT, а тобто передача інформації за внутрішнєю функцією з секретним повідомленням всередині.

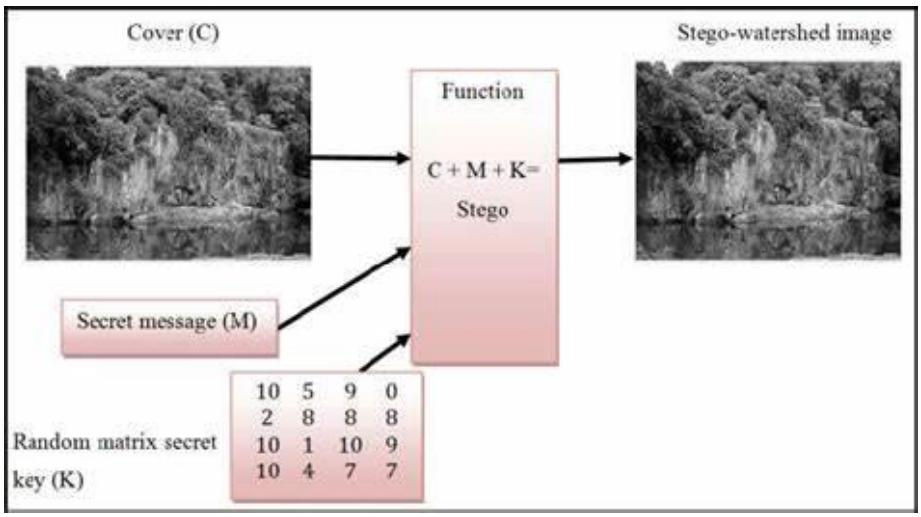

Рисунок 1.2 – Стеганографія в IoT

Як контейнери для стеганографії можна використовувати дані датчиків систем IoT. Температура, вологість, тиск та інші показники можуть бути злегка змінені для вбудовування додаткової інформації. Метод стеганографії в даних датчиків заснований на внесенні контрольованих відхилень у межах допустимих похибок вимірювань і гарантує невидимість для системи моніторингу. Мережева стеганографія в IoT розвивається завдяки використанню спеціальних протоколів. Протокол MQTT дозволяє вбудовувати приховані дані в поля keep-alive, QoS і timestamp, створюючи приховані канали в легітимному трафіку, про що згадано в пункті 1.3.

Промислові IoT-системи використовують стеганографію для безпечного передавання критично важливих команд управління. Вбудовування команд у телеметричні дані дає змогу приховати факт віддаленого управління від потенційних зловмисників, що особливо важливо для критично важливих об'єктів інфраструктури.

У системах “розумного будинку” стеганографія використовується для обміну секретними ключами між пристроями. Дані можуть бути вбудовані у відеопотоки з камер спостереження, аудіосигнали та показання датчиків. Методи розподіленої стеганографії дають змогу створювати захищені пористі мережі IoT-пристроїв із прихованими каналами для обміну службовою інформацією.

Розробляються енергоефективні методи стеганографії для IoT-пристроїв з обмеженим енергоспоживанням. Адаптивне вбудовування враховує поточний заряд батареї і вибирає найкращий метод приховування даних, який мінімізує споживання енергії.

Висновки до розділу 1

1. Досліджено теоретичні основи Інтернету речей (IoT), проаналізовано архітектуру, компоненти та принципи функціонування IoT систем. Розглянуто ключові технології та стандарти, що забезпечують взаємодію IoT пристроїв, а також основні сфери застосування IoT рішень.

2. Проведено комплексний аналіз існуючих алгоритмів захисту в IoT системах, включаючи криптографічні методи, механізми автентифікації та контролю доступу. Досліджено особливості симетричних та асиметричних алгоритмів шифрування, хеш-функцій та цифрових підписів.

3. Проаналізовано сучасні протоколи безпеки та стандарти захисту IoT пристроїв, включаючи MQTT, CoAP, ZigBee та інші. Розглянуто специфікації безпеки від провідних організацій зі стандартизації.

4. Досліджено методи стеганографії як додаткового рівня захисту в IoT системах.

2 АНАЛІЗ АЛГОРИТМІВ СТЕГАНОГРАФІЇ В ІОТ

2.1. Аналіз існуючих стеганографічних методів для захисту конфіденційності в IoT

Модель LSB методу базується на заміні найменш значущих бітів контейнера бітами секретного повідомлення. Базове математичне представлення процесу вбудовування:

$$Si = (Ci - Cimod2) + Mi \quad (2.1)$$

де Si - елемент стегоконтейнера, Ci - елемент контейнера, Mi - біт повідомлення. Принцип роботи методу полягає в тому, що заміна найменш значущого біту призводить до мінімальних змін значення елемента. Для 8-бітного елемента максимальна зміна значення становить 1, що складає менше 0.4% від діапазону можливих значень.

Процес зображений на рисунку 2.1 зображена схема роботи методу.

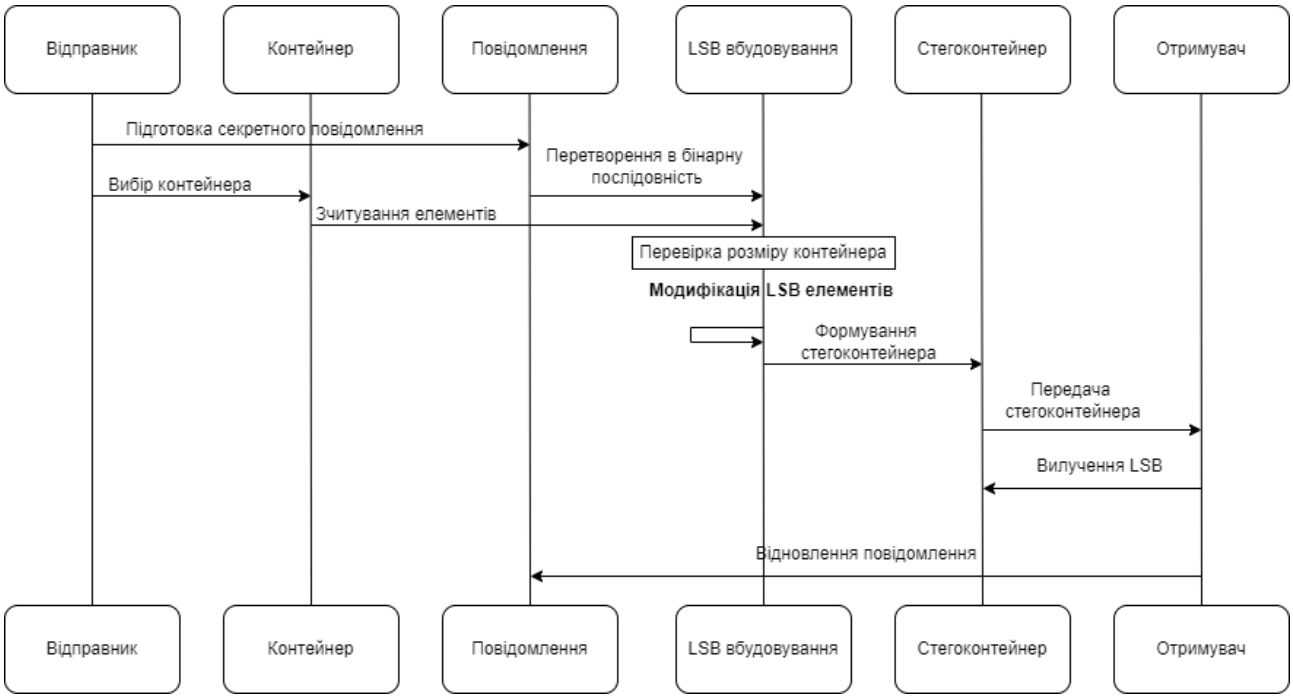

Рисунок 2.1 – Схема роботи методу

Вираз (Ci - Ci mod 2) виконує очищення найменш значущого біту елемента контейнера Ci. Операція mod 2 повертає значення 0, якщо число парне, та 1, якщо

непарне. Віднімання цього залишку від початкового значення встановлює найменш значущий біт в нуль. Після очищення біту виконується додавання біту повідомлення Mi, який може приймати значення 0 або 1. Таким чином, найменш значущий біт результуючого елемента стегоконтейнера Si буде точно відповідати біту повідомлення Mi.

Наведення прикладу. Нехай елемент контейнера *Ci = 157* має бінарне представлення 10011101. Потрібно вбудувати біт повідомлення Mi = 0. Спочатку обчислюється залишок від ділення: *157 mod 2 = 1*, оскільки число непарне. Далі виконується віднімання: 157 - 1 = 156 (бінарне 10011100). На останньому етапі додається біт повідомлення: 156 + 0 = 156. В результаті получається елемент стегоконтейнера *Si = 156* з бінарним представленням 10011100, де найменш значущий біт відповідає біту повідомлення.

Якість вбудовування оцінюється через пікове відношення сигнал/шум. Відношення сигналу/шуму є метрикою оцінки якості вбудовування даних в стеганографії. Для розуміння, можна представити формулу таким чином, де PSNR - відношення сигналу і шуму.

$$PSNR = 10log10\left(MAX_I{}^2 \div MSE\right) \tag{2.2}$$

де множник 10 застосовується для перетворення результату в децибели, після чого логарифм десяти використовується для стиснення широкого діапазону значень відношення, що в дужках. I під кінець MAX_I - представляє собою максимально можливе значення в контейнері. Тобто, для 8-бітних даних MAX_I дорівнює 255, а піднесення до квадрату виконується для нормалізації відносно середньоквадратичної помилки MSE. Стосовно MSE, воно також має свою формулу розрахунку. Зазначається наступним чином:

$$MSE = 1 \div mn \cdot \sum_{i=1}^{m} \quad \sum_{j=1}^{n} \quad [C(i,j) - S(i,j)]^2 \tag{2.3}$$

Середньоквадратична помилка MSE обчислюється як нормалізована сума квадратів різниць між елементами контейнера та стегоконтейнера. Дріб 1/mn виконує нормалізацію суми, де m позначає кількість рядків контейнера, а n -

кількість стовпців контейнера. Перша сума $\sum(i=1 \text{ до } m)$ виконує підсумовування по всім рядкам контейнера від першого до останнього. Друга сума $\sum(j=1 \text{ до } n)$ виконує підсумовування по всім стовпцям контейнера для кожного рядка. Вираз $[C(i,j) - S(i,j)]^2$ обчислює квадрат різниці між відповідними елементами контейнера C(i,j) та стегоконтейнера S(i,j) з координатами i та j.

Для прикладу, застосовується таким метод в IOT і для конфіденційності даних. Нехай існує 8-бітне значення 10110110, що відповідає певному елементу контейнера. В цей елемент необхідно вбудувати один біт секретного повідомлення зі значенням '1'. На першому етапі виконується аналіз молодшого біту вихідного значення. В даному випадку молодший біт дорівнює '0'. За методом LSB цей біт замінюється на біт секретного повідомлення '1'. В результаті отримуємо модифіковане значення 10110111. Різниця між початковим та кінцевим значенням складає лише один біт, що мінімізує спотворення даних. У десятковій системі числення початкове значення 182 змінилося на 183, тобто відхилення становить менше 1%.

Корисно це може бути для передачі показників температури тіла людини в лікарнях. Тобто, на тому самому прикладі, датчик передає значення температури тіла пацієнта як 8-бітне число 36.6°C (10100100 в бінарному форматі). Використовуючи LSB метод, в молодший біт можна вбудувати один біт конфіденційних медичних даних пацієнта. При вбудовуванні біта '1' результуюче значення стане 10100101 (36.7°C), що є допустимим відхиленням для медичних вимірювань, проте зберігає конфіденційність даних, а саме не вказує прямі показники.

Також для захисту конфіденційності такий метод застосовується і в промислових IoT системах, як от захист парамерів виробничих процесів чи приховування налаштуваь систем управління. Окрім промислових чи медичних елементів можна застосовувати і в простому середовищі - системі розумного дому для перехоплення команд управління і захищеної їх передачі, приховування даних систем безпеки чи конфіденційної передачі відеопотоків з камер спостереження.

На рисунку 2.2 зображено алгоритм роботи описаного вище процесу на прикладі лікарні.

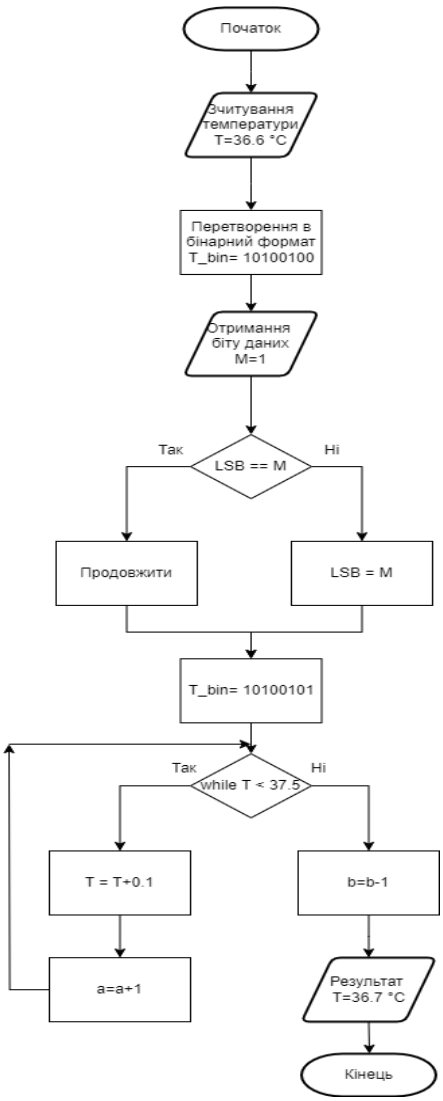

Рисунок 2.2 – Алгоритм роботи LSB на прикладі лікарні

Крім LSB існують ще методи, як от метод розширення спектру. Метод розширення спектру (Spread Spectrum) відрізняється від LSB підходом до приховування даних. Замість заміни окремих бітів контейнера, цей метод розподіляє кожен біт секретного повідомлення по широкому діапазону частот контейнера з використанням псевдовипадкової послідовності.

Принцип роботи базується на розширенні спектру сигналу секретного повідомлення перед його вбудовуванням в контейнер. Процес описується формулою:

$$S(t) = C(t) + \alpha \cdot W(t) \cdot M(t) \quad (2.4)$$

де S(t) представляє результуючий стегосигнал, C(t) є початковим контейнером, W(t) позначає псевдовипадкову послідовність для розширення спектру, M(t) - секретне повідомлення, α - коефіцієнт підсилення, який контролює потужність вбудованого сигналу. На рисунку 2.3 зображено схему роботи методу:

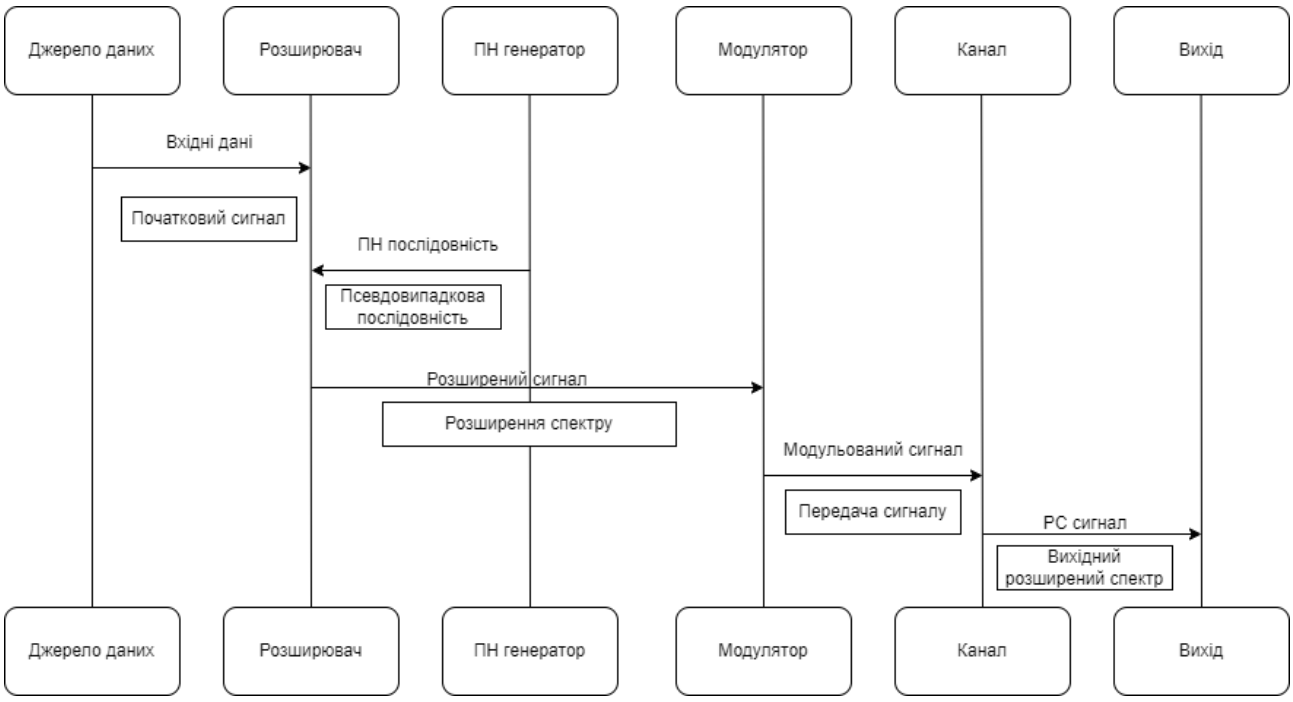

Рисунок 2.3 – Схема роботи розширення спектру

Ключова відмінність від LSB полягає в тому, що Spread Spectrum забезпечує значно вищу стійкість до атак та спотворень. Якщо в LSB зміна одного біту контейнера призводить до втрати біту секретного повідомлення, то в методі розширення спектру інформація розподіляється по багатьом елементам контейнера. Це означає, що навіть при пошкодженні частини контейнера, секретне повідомлення може бути відновлене. Виявлення прихованого повідомлення в методі Spread Spectrum виконується через обчислення коефіцієнта кореляції між стегосигналом та відомою псевдовипадковою послідовністю. Цей процес виражається як формула, описана нижче, де p – коефіцієнт кореляції, а S і W - відповідні значення двох змінних:

$$\rho = \frac{\sum(Si \cdot Wi)}{\sqrt{\sum s_i^2 \cdot \sum W_i^2}} \quad (2.5)$$

На відміну від LSB, який забезпечує максимальну пропускну здатність в 1 біт на елемент контейнера, метод розширення спектру має меншу пропускну здатність через необхідність використання декількох елементів контейнера для кожного біту повідомлення.

Якщо розглядати приклад з лікарнею, то при використанні LSB для приховування даних пацієнта (діагноз, результати аналізів) в показниках датчиків, кожен біт конфіденційної інформації вбудовується в найменш значущий біт телеметричних даних. Наприклад, дані про рівень цукру в крові 5.5 ммоль/л (10110000) можуть містити біт діагнозу в молодшому розряді. При перехопленні такого значення зловмисник може легко виявити наявність прихованих даних через статистичний аналіз молодших бітів.

Також хорошим методом стеганографії для забезпечення конфіденційності вважають вейвлет-перетворення. Цей метод працює з частотними характеристиками сигналу. На рисунку 2.4 зображена схема роботи DWT.

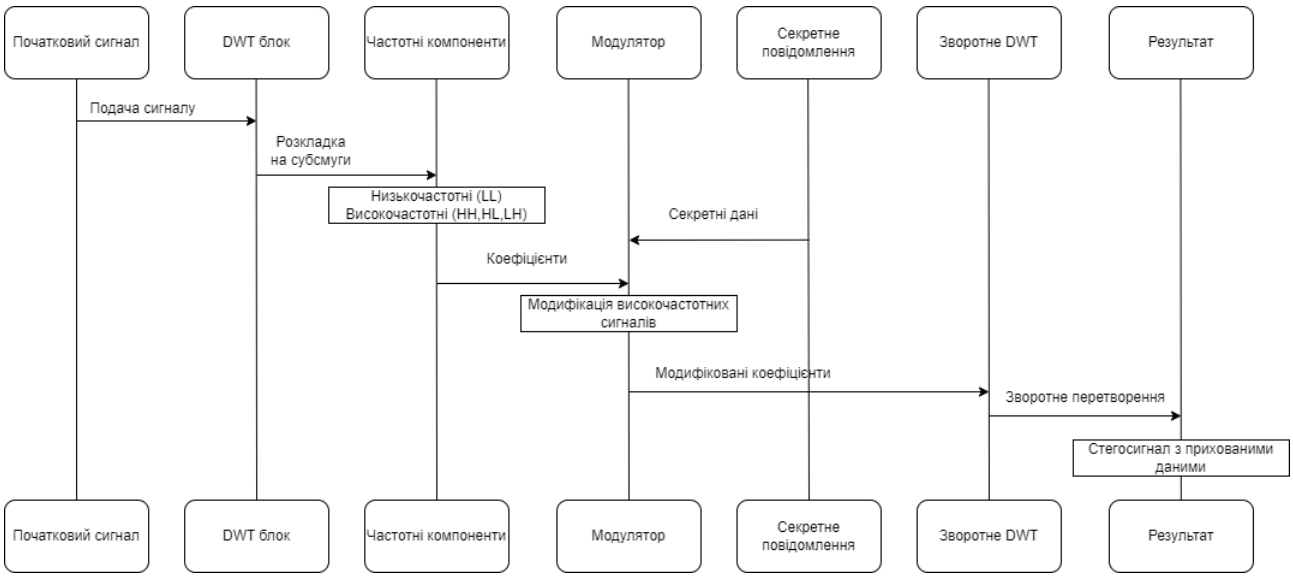

Рисунок 2.4 – Схема роботи DWT

Розглядати алгоритм роботи методу математично можна так:

$$DWT(j,k) = \frac{1}{\sqrt{2^j}} \sum_n \quad x[n]\psi\left(\frac{n-2^j k}{2^j}\right) \tag{2.6}$$

де ψ(t) - материнський вейвлет, j - рівень розкладання, k - зсув. Процес вбудовування конфіденційних даних відбувається через модифікацію вейвлет-коефіцієнтів:

$$S_{coef} = C_{coef} + \alpha \cdot M \tag{2.7}$$

де *S_coef та C_coef* - коефіцієнти вейвлет-перетворення стегоконтейнера та контейнера, M - біти секретного повідомлення, α - коефіцієнт підсилення.

Якщо розгялдати реальний приклад відносно медицини для вищезгаданих методів і сформувати його для DWT.

Якщо температура пацієнта 36.6, то при використанні DWT сигнал температури спочатку розкладається на частотні компоненти (субсмуги) за допомогою вейвлет-перетворення.

Для температурного сигналу отримуємо низькочастотні коефіцієнти, що відповідають за основний тренд температури, та високочастотні, що відображають її коливання.

Конфіденційні дані пацієнта (наприклад, діагноз "Грип") перетворюються в бінарну послідовність та вбудовуються в коефіцієнти високочастотної субсмуги.

Це забезпечує мінімальний вплив на значущі показники температури, оскільки основна інформація про температуру міститься в низькочастотних коефіцієнтах. Тобто, в формулі це виглядатиме як 36.6 + α·(біти_діагнозу), де α - малий коефіцієнт (наприклад, 0.1), що забезпечує непомітність вбудовування.

Після модифікації коефіцієнтів виконується зворотне вейвлет-перетворення, яке формує стегосигнал. Результуюче значення температури може змінитися незначно (наприклад, до 36.7°C), що знаходиться в межах допустимої похибки вимірювання.

При цьому конфіденційний діагноз надійно прихований в структурі сигналу та може бути вилучений тільки авторизованим отримувачем, який знає

параметри використаного вейвлет-перетворення.

В таблиці 2.1. представлено загальні характеристики вищезгаданих методів і їх поріняння:

Характеристика	LSB метод	Spread Spectrum	DWT метод
Пропускна здатність	1 біт/елемент	N/SF біт/елемент	N/2^L біт/елемент
Обчислювальна складність	O(n)	O(n log n)	O(n log n)
Стійкість до атак	Низька	Висока	Дуже висока
Криптографічна стійкість	Відсутня	Присутня	Присутня
Стійкість до шуму	Низька	Висока	Висока
PSNR	50-55 дБ	45-50 дБ	40-45 дБ

Таблиця показує, що по характеристикам криптографічної стійкості, діапазону роботи і пропускної здатності, найкраще підходить криптографічний метод DWT.

2.2. Аналіз існуючих стеганографічних методів для забезпечення автентичності в IoT

Коли було розглянуто стеганографічні методи для конфіденційності, варто розглянути методи і для забезпечення автентичності. Найпоширенішим стеганографічним методом забезпечення автентичності в IoT є вбудовування цифрових водяних знаків у дані датчиків. Цей метод дозволяє приховати ідентифікаційну інформацію пристрою безпосередньо в потік даних, не впливаючи на їх корисне навантаження.

Якщо розглядати математично, то це формула розписана таким чином:

$$S'(t) = S(t) + \alpha W(k) \quad (2.8)$$

де: *S'(t)* - сигнал з вбудованим водяним знаком *S(t)* - оригінальний сигнал від датчика α - коефіцієнт сили вбудовування *W(k)* - водяний знак (бінарна послідовність) t - часовий індекс k - індекс послідовності водяного знаку

Спочатку система отримує дані від IoT-датчика у вигляді цифрового сигналу *S(t)*. Цей сигнал може представляти будь-які вимірювання - температуру, вологість, тиск тощо. Паралельно, використовуючи секретний ключ, генерується унікальний водяний знак *W(k)*, який буде служити ідентифікатором автентичності. Процес вбудовування відбувається шляхом додавання водяного знаку до оригінального сигналу з певним коефіцієнтом сили α. Цей коефіцієнт обирається таким чином, щоб зміни в сигналі були достатніми для надійного виявлення водяного знаку, але не впливали на корисну інформацію. Наприклад, якщо датчик вимірює температуру з точністю до 0.1°C, то зміни, внесені водяним знаком, повинні бути меншими за цю величину. При отриманні даних приймаюча сторона виконує процес екстракції водяного знаку, використовуючи той самий секретний ключ. Порівнюючи витягнутий водяний знак з оригінальним, система може визначити, чи були дані модифіковані під час передачі. Якщо водяний знак співпадає з оригінальним з певною допустимою похибкою, дані вважаються автентичними.

На рисунку 2.5 блокову діаграму, яка описує процес роботи водяної марки.

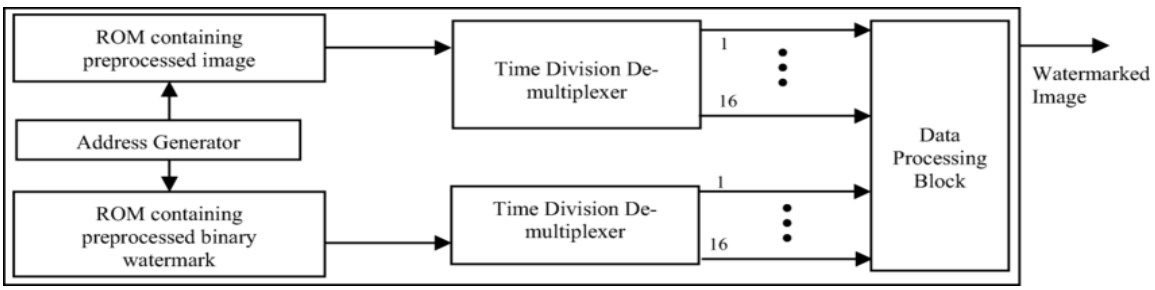

Рисунок 2.5 – Блокова діаграма роботи водяної марки

Для водяної марки в стеганографічних методах можуть використовуватись

методи, згадані з розділу 2.1, такі як LSB чи DWT. Якщо їх принцип роботи зрозумілий, то принцип роботи DCT для забезпечення автентичності потрібно ще розглянути.

Математична основа DCT полягає у представленні сигналу як суми косинусоїдальних функцій різних частот. Для двовимірного сигналу (наприклад, блоку даних) DCT можна представити наступною формулою:

$$F(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (2.9)$$

де f(x,y) - вхідний сигнал, F(u,v) - його частотне представлення, α(u) та α(v) - нормалізуючі коефіцієнти, які обчислюються як:

$$\alpha(u) = \sqrt{\left(\frac{1}{N}\right)} для u = 0$$

$$\alpha(u) = \sqrt{\left(\frac{2}{N}\right)} для u \neq 0$$

В контексті забезпечення автентичності даних IoT пристроїв, DCT має декілька ключових переваг. По-перше, це перетворення дозволяє розділити сигнал на різні частотні компоненти. Високочастотні компоненти відповідають за дрібні деталі сигналу, тоді як низькочастотні - за його загальну структуру. Це дає можливість вбудовувати водяні знаки в ті частотні області, де їх присутність буде найменш помітною для корисного сигналу.

При вбудовуванні водяного знаку процес відбувається наступним чином: спочатку вхідний сигнал розбивається на блоки фіксованого розміру (зазвичай 8x8 або 16x16). До кожного блоку застосовується DCT, після чого отримані коефіцієнти модифікуються відповідно до біт водяного знаку. Модифікація здійснюється за формулою:

$$F'(u,v) = F(u,v) + \alpha W(k) \quad (2.10)$$

де *F'(u,v)* - модифіковані DCT коефіцієнти, α - коефіцієнт сили вбудовування, W(k) - біт водяного знаку.

Найчастіше використовуються середні частоти, оскільки низькі частоти містять найбільш важливу інформацію сигналу, а високі частоти найбільш

вразливі до спотворень при передачі. Коефіцієнти середніх частот забезпечують оптимальний баланс між стійкістю водяного знаку та збереженням якості сигналу.

Процес вилучення водяного знаку виконується у зворотному порядку. Отриманий сигнал розбивається на блоки, до кожного застосовується DCT, і з відповідних частотних компонентів витягується інформація про водяний знак. Для підвищення надійності часто використовується додаткове кодування водяного знаку з використанням завадостійких кодів.

Власне, що таке це заводостійке кодування - процес перетворення вихідного повідомлення у послідовність, яка має додаткову надлишковість. Ця надлишковість дозволяє виявляти та виправляти помилки, що можуть виникнути під час передачі або внаслідок спотворень сигналу. Найчастіше використовуються коди Ріда-Соломона та БЧХ-коди (Боуза-Чоудхурі-Хоквінгема).

При використанні кодів Ріда-Соломона процес кодування можна представити математично наступним чином:

$$C(x) = M(x) \cdot x^{(n-k)} + R(x) \qquad (2.11)$$

де: *C(x)* - закодоване повідомлення, *M(x)* - вихідне повідомлення водяного знаку, n - загальна довжина кодового слова, k - довжина інформаційної частини, *R(x)* - залишок від ділення *M(x) * x^(n-k)* на породжувальний поліном.

Процес захисту водяного знаку за допомогою завадостійкого кодування складається з декількох етапів.

Етап перший, це коли водяний знак розбивається на блоки фіксованої довжини. До кожного блоку додаються перевірочні біти, які обчислюються за допомогою спеціальних математичних операцій. Кількість та розташування цих бітів залежить від обраного типу коду та необхідного рівня захисту.

Другий етап заключається в тому, що виправляються помилки. Наприклад, код Ріда-Соломона RS(255,223) може виправити до 16 помилок у блоці з 255 символів. Це означає, що навіть якщо частина водяного знаку буде пошкоджена

під час передачі або внаслідок атак, система зможе відновити оригінальне повідомлення. При декодуванні водяного знаку система спочатку перевіряє наявність помилок за допомогою синдромів - спеціальних значень, які обчислюються на основі отриманого повідомлення. Якщо синдроми не дорівнюють нулю, це означає наявність помилок.

Далі застосовується алгоритм виправлення помилок, який використовує надлишкову інформацію для відновлення оригінального повідомлення.

Ще одним стеганографічним алгоритмом є DWT, що зновуж таки, згадувався в розділі 2.1. Але, було придумано гібридні методи, які покращують результати з точки зору стійкості та візуальної якості. Варто спочатку розглянути SVD - метод на основі сингулярного розкладання. SVD розкладає матрицю зображення на три матриці, де діагональна матриця містить сингулярні числа, які можуть бути модифіковані для вбудовування даних.

Це є три матриці U, S і V в степені T. Перша матриця - ортогональна матриця, яка включає себе сингулярні вектори.

Ортогональна матриця U лівих сингулярних векторів є однією з трьох матриць, що отримуються при SVD-розкладі. Це квадратна матриця розміром m×m, де m - кількість рядків вихідної матриці A. Її стовпці являють собою власні вектори матриці A в степені T. Ключовою властивістю матриці U є її ортогональність, що математично виражається рівністю $U \times U$ в степені $T = U$ в степені $T \times U = I$, де I - одинична матриця. Це означає, що всі стовпці матриці U взаємно ортогональні та мають одиничну довжину. Для розуміння цього розглянемо простий приклад матриці U розміром 3×3:

$$U = [[\ 0.83\ \ -0.39\ \ \ 0.39]$$

$$[\ 0.39\ \ \ 0.83\ \ -0.39]$$

$$[-0.39\ \ \ 0.39\ \ \ 0.83]]$$

Якщо взяти будь-які два різні стовпці цієї матриці та обчислити їх скалярний добуток, результат буде дорівнювати нулю, що підтверджує їх ортогональність. Наприклад, для першого та другого стовпців: $0.83 \times (-0.39) +$

0.39 × 0.83 + (-0.39) × 0.39 ≈ 0. Крім того, сума квадратів елементів кожного стовпця дорівнює одиниці, що підтверджує їх нормованість. При обчисленні матриці U використовується ітеративний процес знаходження власних векторів матриці A в степені T. Один з поширених алгоритмів - метод степеневої ітерації. Спочатку обчислюється добуток A в степені T , потім для отриманої матриці знаходяться власні вектори. Ці власні вектори формують стовпці матриці U.

Друга матриця S - діагональна матриця S (матриця сингулярних значень) є центральним компонентом SVD-розкладу.

При розкладі матриці A розміром m×n, матриця S має розмірність m×n та містить невід'ємні дійсні числа, розташовані на головній діагоналі у порядку спадання.

$$S = [\sigma_1 \;\; 0 \;\; 0$$

$$0 \;\; \sigma_2 \;\; 0$$

$$0 \;\; 0 \;\; \sigma_3]$$

Матриця S має унікальну структуру, де всі елементи поза головною діагоналлю дорівнюють нулю. Сингулярні значення σ_i на головній діагоналі впорядковані за спаданням: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq ... \geq \sigma_n \geq 0$. При обчисленні матриці S, сингулярні значення визначаються як квадратні корені з власних значень матриці $A^T \times A$ або $A \times A^T$. Тобто, якщо λ_i є власним значенням матриці $A^T \times A$, то $\sigma_i = \sqrt{\lambda_i}$.

Якщо розглядати на прикладі, де сингулярні значення дорівнюють 100, 85 і 20, то матриця матиме вигляд:

$$S = [100 \;\; 0 \;\; 0$$

$$0 \;\; 85 \;\; 0$$

$$0 \;\; 0 \;\; 20]$$

Для наближеного представлення даних часто використовують властивість, що при відкиданні найменших сингулярних значень (заміні їх нулями)

отримується найкраща апроксимація матриці заданого рангу в сенсі норми Фробеніуса.

Наприклад, якщо в наведеній вище матриці залишити тільки перші два сингулярних значення:

$$S \approx [100 \quad 0 \quad 0$$

$$0 \quad 85 \quad 0$$

$$0 \quad 0 \quad 0]$$

І третя матриця - матриця *V* в степені *T* (транспонована матриця правих сингулярних векторів) є третьою складовою SVD-розкладу. Це ортогональна матриця розміром $n \times n$, де n - кількість стовпців вихідної матриці *A*. При транспонуванні рядки матриці *V* в степені *T* стають стовпцями матриці *V*, і навпаки. Математично *V* в степені *T* отримується з власних векторів матриці *A* в степені $T \times A$. Якщо розглянути приклад матриці 3×3, вона може мати вигляд:

$$V^T = [0.58 \quad -0.73 \quad 0.35$$

$$0.73 \quad 0.58 \quad -0.35$$

$$-0.35 \quad 0.35 \quad 0.87]$$

Кожен рядок цієї матриці є правим сингулярним вектором, нормованим до одиничної довжини. Властивість ортогональності проявляється в тому, що скалярний добуток будь-яких двох різних рядків дорівнює нулю, а добуток рядка на самого себе дорівнює одиниці. При множенні матриць у SVD-розкладі ($A = U \times S \times V$ *в степені* *T*), *V* в степені *T* відповідає за поворот простору після масштабування, виконаного матрицею *S*. Якщо розглядати це як послідовність перетворень, то *V* в степені *T* повертає базисні вектори таким чином, щоб вони відповідали напрямкам максимальної варіації даних у просторі стовпців.

Тобто, при обробці даних матриця *V* в степені *T* часто використовується для проекції даних на новий базис, де компоненти даних стають некорельованими.

На основі цих матриць і загалом методу, а також методу DWT створено

гібридний метод SVD-DWT. Спочатку зображення обробляється за допомогою DWT, яке розділяє його на частотні області. При цьому зображення розкладається на чотири частини: апроксимуючу частину (низькочастотну) та три деталізуючі частини (високочастотні). Апроксимуюча частина містить основну інформацію про зображення, тоді як деталізуючі частини відповідають за горизонтальні, вертикальні та діагональні деталі. Після розкладу DWT обирається апроксимуюча частина, оскільки вона найбільш стійка до різних видів атак та модифікацій. Саме до цієї частини застосовується SVD-розклад. При цьому апроксимуюча частина розкладається на три матриці: ліві сингулярні вектори, діагональну матрицю сингулярних значень та праві сингулярні вектори.

Водяний знак вбудовується шляхом модифікації сингулярних значень у діагональній матриці. Після модифікації виконується зворотне перетворення: спочатку відновлюється модифікована апроксимуюча частина за допомогою зворотного SVD, а потім застосовується зворотне вейвлет-перетворення для отримання фінального зображення з вбудованим водяним знаком.

2.3. Дослідження інтеграції стеганографічних методів в архітектуру IoT

Для інтеграції стеганографічних методів потрібно розглянути взаємодію всіх компонентів системи та процес обробки даних в IoT. На рисунку 2.6 зображені компоненти.

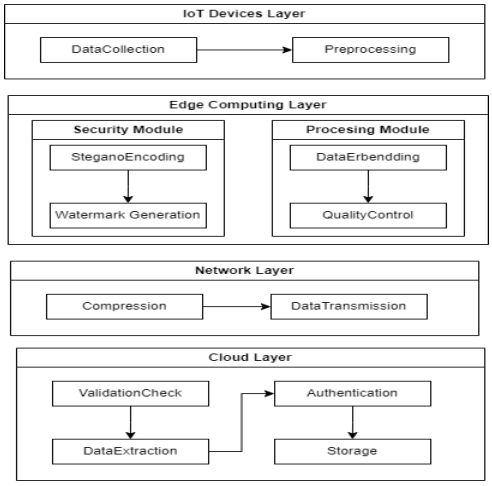

Рисунок 2.6 – Архітектура IoT

Процес інтеграції стеганографічних методів в архітектуру Інтернету речей розподілений по декількох функціональних рівнях, кожен з яких виконує певне завдання. На рівні пристроїв Інтернету речей базові дані збираються за допомогою мережі різних датчиків і камер, які складають основний потік інформації. Ці дані попередньо обробляються для оптимізації структури та формату, щоб забезпечити ефективність подальшого використання. На цьому ж рівні виконується початкове кодування даних, яке готує їх до впровадження стеганографії на наступних етапах обробки.

На рівні розрахунку межі є 2 модуля, які пов'язані один з одним. Модуль безпеки забезпечує всебічний захист даних за рахунок створення унікальних цифрових водяних знаків, які допомагають перевірити точність інформації. Цей модуль також відповідає за пряме вбудовування стеганографічної інформації і виконує шифрування критично важливих даних для забезпечення додаткового рівня захисту. жовтень.

Паралельно з модулем безпеки знаходиться модуль обробки, який контролює якість вбудованої інформації стеганографії для забезпечення цілісності даних. Цей модуль також виконує оптимізацію та стиснення даних для ефективного використання каналів передачі. Заключним етапом модуля обробки є підготовка даних до передачі, яка включає форматування та налаштування інформації відповідно до вимог мережевого протоколу.

В основі мережевого рівня лежить підхід до безпечної передачі даних між вузлами мережі. Цей процес починається з встановлення захищених каналів зв'язку, які використовують сучасні протоколи шифрування для створення базового рівня захисту стеганографічно прихованої інформації. Система безпечної передачі даних на мережевому рівні інтегрується з механізмами стеганографічного захисту, забезпечуючи додатковий рівень безпеки для прихованої інформації. При цьому використовуються спеціалізовані протоколи, які враховують особливості вбудованих стеганографічних даних.

На рисунку 2.7 зображено алгоритм передачі з стеганографією на мережевму рівні.

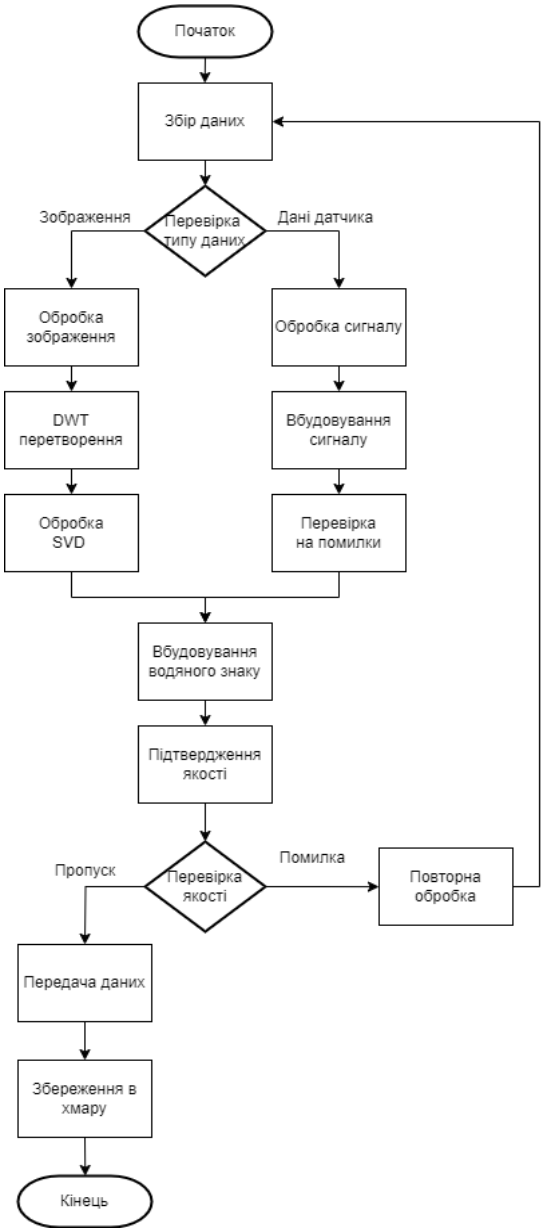

Рисунок 2.7 – Алгоритм передачі даних в стеганографії

На хмарному рівні відбувається фінальна обробка та зберігання даних IoT системи. Цей рівень насамперед забезпечує вилучення вбудованої стеганографічної інформації та її ретельну перевірку для підтвердження автентичності отриманих даних. Важливим аспектом роботи хмарного рівня є проведення комплексної аутентифікації як самих пристроїв IoT, так і даних, що від них надходять. Після успішної верифікації відбувається надійне зберігання інформації у хмарному середовищі та її подальший аналіз з використанням спеціалізованих інструментів обробки даних.

Після опису рівнів, також варто дослідити і модулі, які застосовуються в

IoT для обробки та передачі даних. Так, в таблиці 2.2. представлено основні модулі системи, які включені в IoT, враховуючи стеганографічну обробку.

Таблиця 2.2 – Модульна структура системи IoT зі стеганографічним захистом

Назва модуля	Компоненти	Функціональність	Взаємодія з іншими модулями
Модуль збору даних	Інтерфейси сенсорів, драйвери камер, система буферизації	Отримання даних з різних джерел, первинна обробка вхідних даних, тимчасове зберігання	Передає дані до модуля стеганографічної обробки, отримує параметри якості від модуля безпеки
Модуль стеганографічної обробки	DWT бібліотеки, SVD бібліотеки, системи контролю якості	Частотні перетворення, вбудовування водяних знаків, оцінка якості вбудовування	Отримує дані від модуля збору, взаємодіє з модулем безпеки для отримання водяних знаків, передає оброблені дані модулю передачі
Модуль безпеки	Генератори водяних знаків, криптографічні системи, механізми аутентифікації	Створення унікальних водяних знаків, шифрування даних, перевірка автентичності	Надає водяні знаки модулю стеганографічної обробки, взаємодіє з модулем передачі для захисту каналу
Модуль передачі даних	Протоколи передачі, детектори помилок, системи відновлення	Безпечна передача даних, виявлення та виправлення помилок, відновлення втрачених даних	Отримує дані від стеганографічного модуля, взаємодіє з модулем безпеки, забезпечує зв'язок з хмарою

Проте, всі модулі по мірі підключення будуть нагружати систему, тому

варто також впевнитись, що можна розрахувати вплив стеганографії на систему IoT.

Ефективність системи значною мірою залежить від правильного розподілу обчислювальних завдань між пристроями з урахуванням їх можливостей та обмежень.

Зазвичай, використовують формулу

$$E = Pc + Pm + Pt \tag{2.12}$$

де *Pc* - споживання ресурсів під час обчислень, Pm - використання пам'яті, Pt - витрати на передачу даних. *A* зі стеганографєю це доповниться наступним чином:

$$(DWT_{cost} + SVD_{cost}) \times Data_{Processing_{time}} \tag{2.13}$$

де DWT_cost та SVD_cost представляють обчислювальну складність відповідних перетворень, а Processing_time - час, необхідний для обробки.

DWT_cost включає декілька компонентів обчислювальної складності:

$$DWT_{cost} = (N \times logN) \times L \times CF \tag{2.14}$$

де *N* - розмір блоку даних для обробки, *L* - кількість рівнів декомпозиції (зазвичай 3-4 для IoT пристроїв), *CF* - коефіцієнт складності фільтрів (для простих фільтрів *Хаара CF=2*, для більш складних фільтрів CF може досягати 8-10) SVD_cost розраховується за формулою:

$$SVD_{cost} = M \times N^2 \times I \tag{2.15}$$

де *M* - розмір матриці даних, *N* - кількість сингулярних значень, *I* - кількість ітерацій для досягнення збіжності (зазвичай 10-15 для достатньої точності).

Відповідно за розмір файлу неможливо сказати, бо це залежить від здатності самих приладів та мережі, як і для часу, який включатиме в себе фактор додаткової операції, що становитиме нехай секунду.

Останнім пунктом при інтеграції вважатиметься корисним для згадування – адаптивність.

На відміну від розподілу обчислювальних завдань, система може сама адаптуватись до певних факторів, такі як зниження швидкості мережі чи навіть те саме збільшення пристроїв в одній мережі.

При чому, що формула не зміниться для обрахування, лише пропаде час, тому що в основі адаптивності буде стояти обчислювальна складність перетворень, а при адаптації враховуватиметься поточне навантаження, обсяг даних для обробки, пропускна здатність каналу і доступна пам’ять.

Висновки до розділу 2

1. Проведено детальний аналіз стеганографічних методів захисту конфіденційності в IoT системах, зокрема досліджено методи LSB та DWT.

2. Досліджено методи забезпечення автентичності в IoT на основі цифрових водяних знаків, включаючи DCT, SVD та гібридний метод DWT-SVD

3. Розглянуто комплексний підхід до інтеграції стеганографічних методів в архітектуру IoT, що включає модулі збору даних, стеганографічної обробки, безпеки та передачі даних в рівнях, модулях. Розглянуто адаптивність та навантаження IoT.

3 РОЗРОБКА МЕТОДІВ ЗАХИСТУ ДАНИХ ІНТЕРНЕТ-РЕЧЕЙ

3.1. Реалізація стеганографічного методу

Для розробки методу було використано мову Python. Перш за все, було налаштоване середовище розробки, що видно на рисунку 3.1 Віртуальне оточення venv було підключене для того, щоб в подальшому можна було встановлювати лише необхідні модулі для роботи, а також для того, щоб можна було правильно в кінцевому результаті скомпонувати файли залежностей, коли продукт буде застосовуватись для розповсюдження.

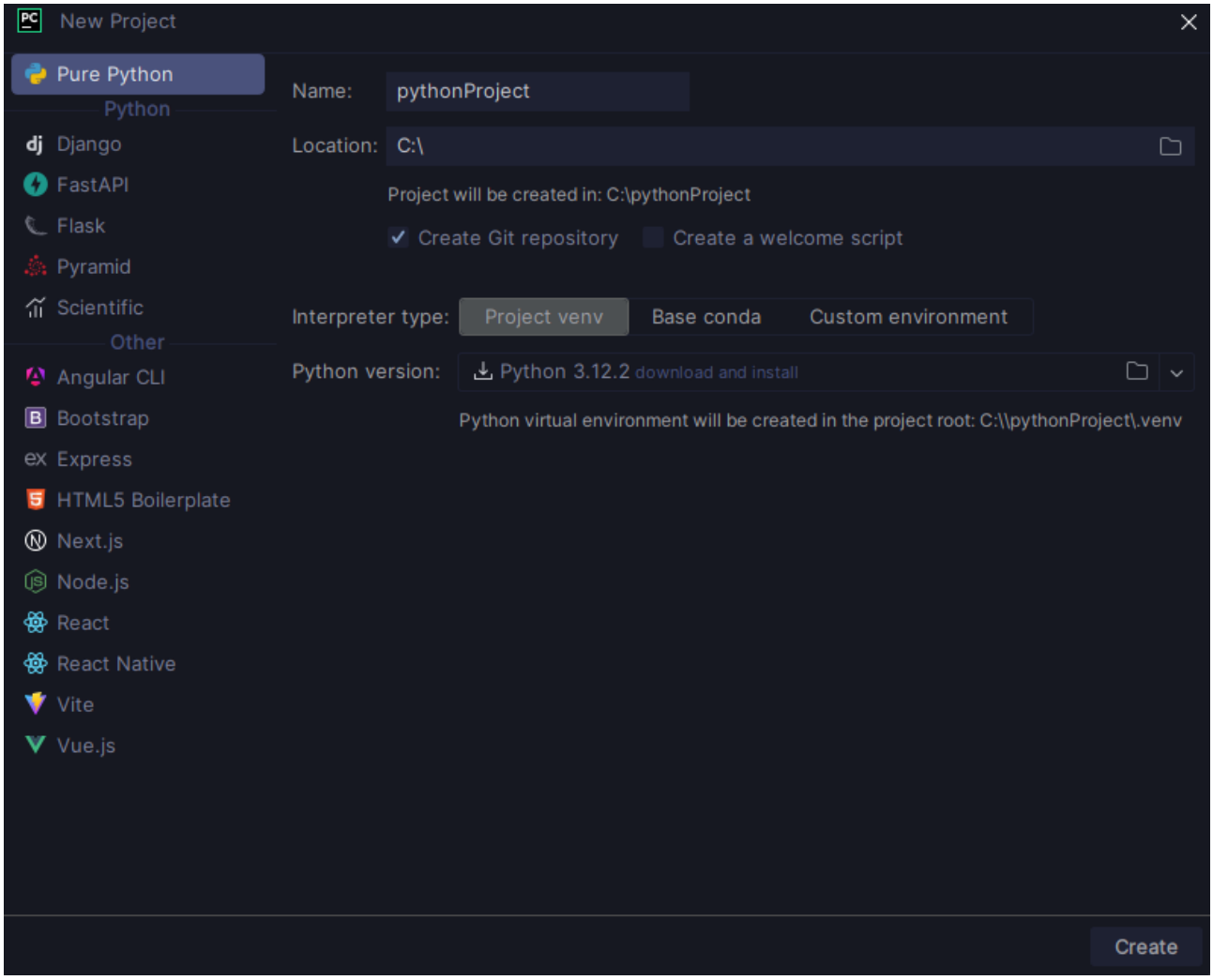

Рисунок 3.1 – Віртуальне оточення

Після створення проєкту, було сформовано архітектуру, яка підлягає

принципам ООП та СОЛІД. Тобто, було сформовано класи PhotoStegaography та TextSteganography. На рисунку 3.2 видно структуру проекту.

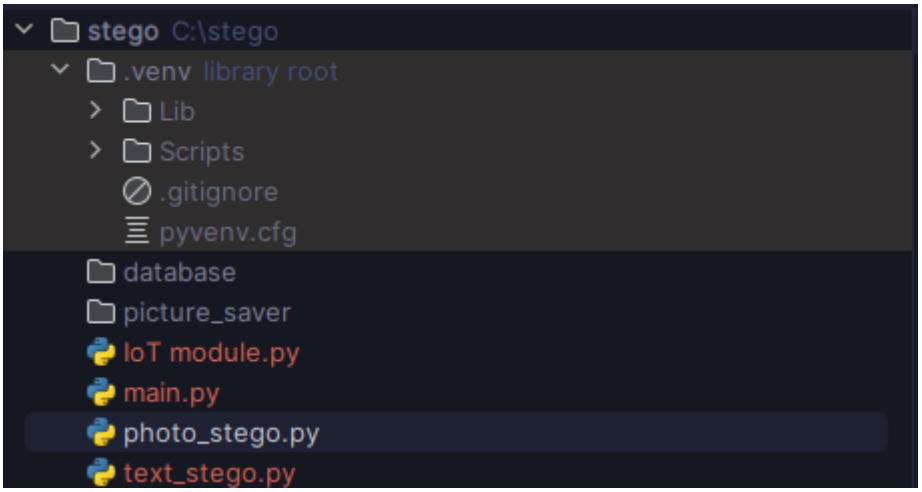

Рисунок 3.2 – Структура проекту

Після розробки структури, було завантажено необхідні модулі для роботи з картинками, в подальшому які зможуть опрацьовувати кадри IoT і передавати їх безпечним шляхом. На рисунку 3.3 зображено результат завантаження модуля Pillow, NumPy, які знадобляться для фотографій.

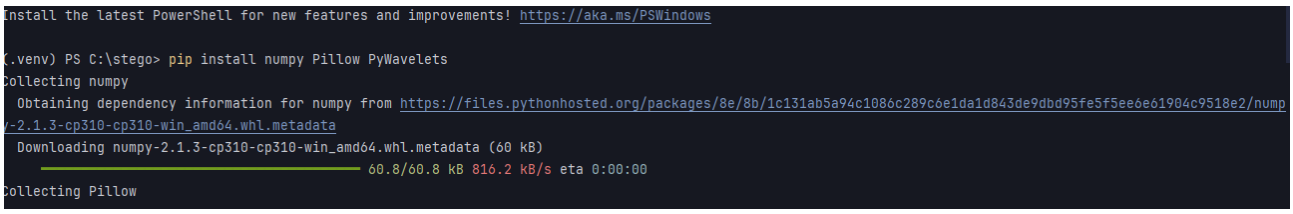

Рисунок 3.3 – Завантаження модулів

Після завантаження модулів, розпочато реалізацію текстової стеганографії на прикладі з розділу 2.1. Найпростішим виявився, звісно, модуль LSB - перестановки біта. Якщо розглядати метод, то в коді він виглядатиме наступним чином:

binary_secret = ''.join(format(ord(c), '08b') for c in secret)
if len(binary_secret) > len(text):

```
    raise ValueError("Secret too large")
result = list(text)
for i in range(len(binary_secret)):
    char = ord(result[i])
    char &= ~1
    char |= int(binary_secret[i])
    result[i] = chr(char)
return ''.join(result)
```

Перша стрічка відповідає за перетворення кожного символу в ASCII код, а 08b перетворює число в бінарний рядко дожиною в 8 біт. Тобто, якщо секрет = 1, то *ord("1") = 49, a format(49, '08b') = 00110001.* Друга стрічка з умовою перевіряє, чи достатньо символів в тексті для приховування секрету. Кожен біт секрету потребує один символ тексту. Відповідно, в секреті буде знаходитись інформація, яку потрібно передати. Потім текст перетворюється в список символів для можливості їх модифікації. Потім код проходить по кожному біту секретного повідомлення. Для кожного біту береться відповідний символ з тексту та перетворюється в його ASCII код.

В кожному символі тексту ми модифікуємо тільки останній (найменш значущий) біт. Спочатку останній біт обнуляється через побітове І з числом, де всі біти одиниці крім останнього. Наприклад, якщо у нас є ASCII код 49 (00110001), після обнулення останнього біту отримаємо 48 (00110000).

Після обнулення ми встановлюємо останній біт відповідно до поточного біту секрету через побітове АБО. Якщо біт секрету 1, то отримаємо 00110001, якщо 0 - залишиться 00110000. Змінений ASCII код перетворюється назад у символ. Нарешті, всі модифіковані символи об'єднуються назад в текстовий рядок. Оскільки відомо тільки останні біти символів, такі зміни майже непомітні при читанні тексту, але дозволяють зберегти і потім відновити приховану інформацію. Відповідно, на рисунку 3.4 результат дії кодування.

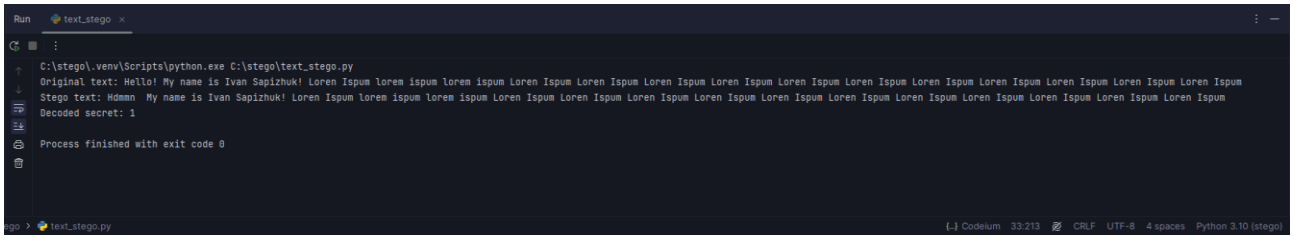

Рисунок 3.4 – Результат роботи LSB методу

Також на рисунку 3.4. представлено пошук секретного символу. Це відбувається наступним чином. У першому циклі код витягує приховані біти. Він проходить по кожному символу стего-тексту і дивиться на його останній біт. Це робиться через побітове І з одиницею (операція & 1). Оскільки 1 в бінарному вигляді це 00000001, така операція залишить тільки значення останнього біту. Всі отримані біти збираються в один бінарний рядок binary_secret.

Другий цикл перетворює отримані біти назад у текст. Він бере біти групами по 8 штук (один байт), бо кожен символ в ASCII кодується 8 бітами. Кожна така група з 8 біт конвертується з бінарного формату в число через int(byte, 2), де 2 вказує на бінарну систему числення. Потім число перетворюється в символ функцією chr(). Перевірка на довжину байту рівну 8 потрібна щоб уникнути помилок при неповних байтах. Тобто, в коді реалізація виглядає наступним чином:

```
binary_secret = ''
for i in range(message_length * 8):
    if i >= len(stego_text):
        break
    binary_secret += str(ord(stego_text[i]) & 1)
secret = ''
for i in range(0, len(binary_secret), 8):
    byte = binary_secret[i:i + 8]
    if len(byte) == 8:
        secret += chr(int(byte, 2))
return secret
```

Відповідно, текстова стеганографія працює відмінно. Але потрібно реалізувати також і більш важчий метод для забезпечення безпеки чутливих даних, таких от як передача значень датчиків.

Для цього вирішено реалізувати метод DWT, який якраз містить в собі роботу з фотографіями. Щоб все працювало в python, вирішено підгрузити модулі, що були завантажені на початку роботи. На рисунку 3.5 зображено імпортування модулів в другий файл, а саме PhotoStego, тих модулів, які знадобляться для роботи з фотографіями.

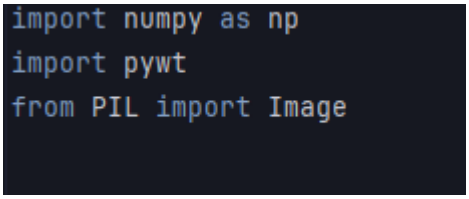

Рисунок 3.5 – Імпортування модулів для Photo

Вирішено реалізувати метод DWT. Для цього прописано клас DWT Stego, який містить в собі методи ініціалізації, підготовки повідомлення, обробки бітів в повідомленнях.

Було вирішено використати вейвлет-перетворення для розкладання зображення на частотні компоненти. В конструкторі встановлено тип вейвлету як 'haar', який визначено як найпростіший і найбільш поширений вейвлет для базового аналізу сигналів. Параметр level=1 вказує на одноразове застосування DWT, що створює один рівень розкладання зображення на апроксимуючі та деталізуючі коефіцієнти. Вейвлет Хаара обрано через ефективність для DWT стеганографії, оскільки зберігаються різкі переходи в зображенні. При level=1 отримано чотири частотні підсмуги (LL, LH, HL, HH), де LL підсмуга використовується для вбудовування даних. Цей метод створено для ініціалізації базових параметрів, які визначають процес розкладання зображення. Один рівень декомпозиції визначено оптимальним для балансу між якістю приховування та обчислювальною складністю. Застосування вейвлету Хаара

забезпечує локалізацію як в часовій, так і в частотній областях. В коді це виглядає так:

```
class DWTSteganography:
    def __init__(self):
        self.wavelet = 'haar'
        self.level = 1
```

Далі було розроблено процес підготовки повідомлення для вбудовування в DWT коефіцієнти. Кожен символ повідомлення конвертується в ASCII код через функцію ord(). ASCII код трансформується в 8-бітне бінарне число через format з параметром '08b'. Всі бінарні послідовності об'єднуються в єдиний бінарний рядок. Створений бінарний рядок застосовується для модифікації найменш значущих бітів DWT коефіцієнтів. Бінарне представлення обрано через специфіку роботи з цифровими сигналами в DWT. Кожен біт секретного повідомлення інтегрується в один коефіцієнт DWT перетворення. Розроблений підхід мінімізує вплив на візуальну якість зображення. Процес конвертації оптимізовано для подальшого вбудовування в частотну область. Частина з кодом виглядає наступним чином:

```
    def _prepare_message(self, message):
        return ''.join(format(ord(c), '08b') for c in message)
```

Такий ж код був і в текстовій стеганографії, проте для подальшої обробки і збереження правильності написання коду вирішено його помістити в окремий метод для виклику.

Відповідно, також створено метод відновлення прихованого повідомлення для тестуванн системи. За логіку взято те, що бінарний рядок розділяється на групи по 8 бітів, враховуючи ASCII кодування. Реалізовано конвертацію кожної групи з 8 бітів з бінарного формату в десяткове число через int(byte, 2). Отримане

число трансформується в символ за допомогою функції chr(). Відновлені символи збираються в фінальне повідомлення. Метод розроблено як обернений до _prepare_message для вилучення бітів з DWT коефіцієнтів. Процес декодування оптимізовано для точного відновлення оригінального повідомлення. Власне, в коді це виглядає так:

```
def _bits_to_message(self, bits):
    message = ''
    for i in range(0, len(bits), 8):
        byte = bits[i:i + 8]
        message += chr(int(byte, 2))
    return message
```

Після оголошення основних методів, потрібно реалізувати і сам метод DWT. В методі кодування існує декілька блоків. Перший блок - блок підготовки зображення. Зображення відкривається та конвертується в градації сірого через конвертер 'L'. Далі створюється числовий масив з зображення для подальшої обробки через numpy. Це необхідно для застосування вейвлет-перетворення до цифрових даних зображення. Другий блок - блок перетворення. Застосовується багаторівневе вейвлет-перетворення wavedec2 до масиву зображення з вказаним типом вейвлету та рівнем декомпозиції. З отриманих коефіцієнтів виділяється LL підсмуга (апроксимуючі коефіцієнти), яка буде використана для вбудовування даних.

Після цього секретне повідомлення конвертується в бінарний формат через метод _prepare_message. Перевіряється чи достатньо місця в LL підсмузі для вбудовування всього повідомлення. Якщо повідомлення завелике - генерується помилка.

Третій блок - вбудовування даних. Кожен біт бінарного повідомлення вбудовується в найменш значущий біт відповідного коефіцієнта LL підсмуги. Позиція в підсмузі розраховується через ділення індексу на ширину підсмуги для

отримання рядка та остачі для отримання стовпця. Модифікація відбувається через побітові операції.

В свою чергу модифікована LL підсмуга повертається в набір коефіцієнтів. Виконується зворотне вейвлет-перетворення waverec2 для отримання стегозображення. Результат конвертується в формат зображення та зберігається у вказаний файл.

В додатку А представлено метод для стеганографії фотографій.

І, знову ж таки, для тестування розроблено метод декодування. В процесі декодування відкривається стегозображення та конвертується в градації сірого для подальшої обробки.

Створюється числовий масив з зображення за допомогою numpy для можливості застосування математичних операцій.

До отриманого масиву застосовується багаторівневе вейвлет-перетворення з тими ж параметрами, що використовувались при кодуванні - тип вейвлету та рівень декомпозиції. З отриманих коефіцієнтів виділяється LL підсмуга, яка містить приховане повідомлення в найменш значущих бітах. Створюється порожній рядок для збору бінарного повідомлення.

В циклі проходиться по кожному біту прихованого повідомлення, де довжина циклу визначається як довжина повідомлення помножена на 8 (кількість бітів в байті).

Для кожного біту розраховується його позиція в LL підсмузі через ділення індексу на ширину підсмуги для отримання номеру рядка та остачі для отримання номеру стовпця. З кожного коефіцієнта LL підсмуги витягується найменш значущий біт через побітове І з одиницею та додається до бінарного повідомлення.

Отримане бінарне повідомлення передається в метод _bits_to_message для конвертації назад в текст, де кожні 8 бітів перетворюються в символ. Реалізацію також можна побачити в додатку А.

Після опису всіх методів, було обрано фотографію з інтернету, допоки немає реальних даних з інструментів IoT. Така фотографія представлена на рисунку 3.6

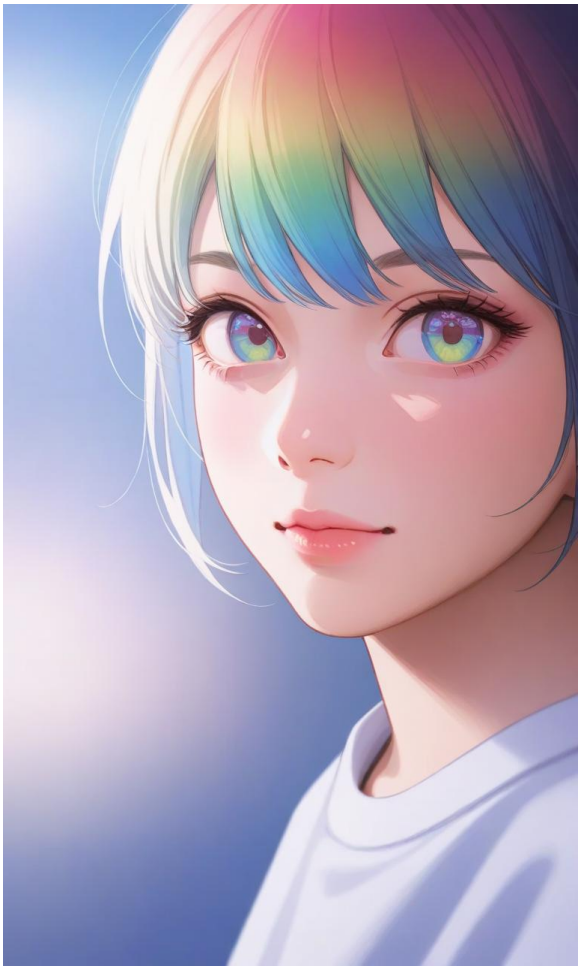

Рисунок 3.6 – Не оброблена фотографія

Після запуску рішення, було створено вихідний файл, який пройшов обробку стеганографією. На рисунку 3.7 зображено результат роботи програми і створення файлу output.png, в якому є приховане повідомлення.

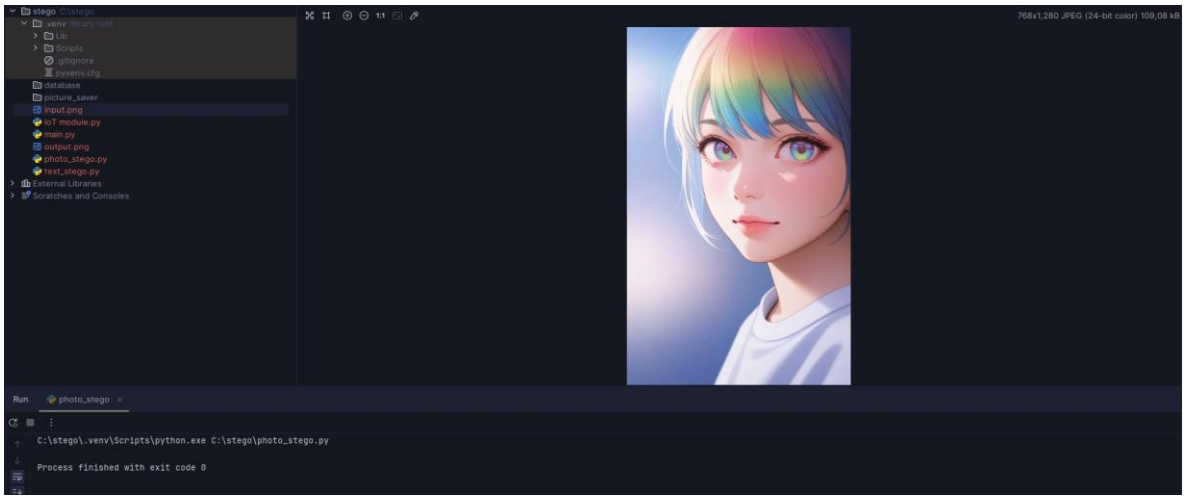

Рисунок 3.7 – Створення другого файлу output.png

Якщо порівняти, звісно, можна замітити втрату кольорів у вихідного файлу. На рисунку 3.8 зображено вихідний файл з реалізації методу.

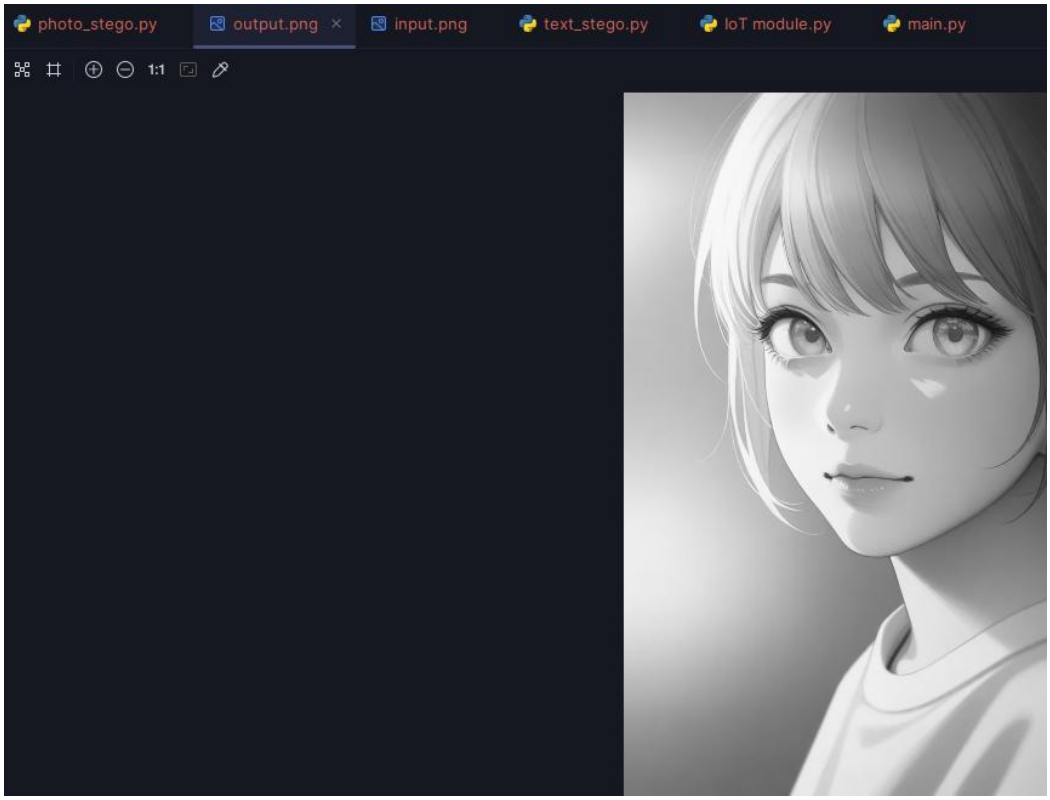

Рисунок 3.8 – Фотографія зі стеганографією.

З однієї сторони, якщо пристрої IoT передають зображення в чорно білих тонах, то різниці між вихідним і стартовим файлом не будуть відчутні. Проте, для покращення механізму можна реалізувати рандомайзер, який буде брати один піксель, а не всю роботу. Впевненість в реалізованому методі є, тому потрібно було змінити вищезгаданий код наступним чином.

Спочатку потрібно було імпортувати модуль рандому, який дозволить хапати окремі пікселі. Далі прописати, щоб реалізація брала пікселі і взаємодіяла лише з ними, міняючи їх сутність. В оригінальному коді було додано візуальне відображення змінених пікселів. Ключовий момент - взаємодія з випадковими позиціями:

positions = random.sample(range(LL[0].size), message_length)

Ця стрічка вибирає випадкові унікальні позиції в межах розміру LL підсмуги. Кількість позицій дорівнює довжині повідомлення.

Для кожної такої позиції:

```
channel = random.randint(0, 2)
row = pos // LL[channel].shape[1]
col = pos % LL[channel].shape[1]
```

Для візуалізації було додано код малювання червоних квадратів:

```
img_row = int(row * (img.height / LL[channel].shape[0]))
img_col = int(col * (img.width / LL[channel].shape[1]))
draw.rectangle(
    [(img_col-2, img_row-2), (img_col+2, img_row+2)],
    outline='red'
)
```

Така реалізація дозволила не змінювати повністю картинку, а шукати рандомні значення та записувати стеганографію туда, щоб при роботі з кольоровими зображеннями в IoT можна було бути більш впевненим у захисті даних. Автентичність, власне, також збережена.

На рисунку 3.9 представлено зміни у фотографії з візуалізацією того, що відбулось при зміні коду і куди саме було вшите засекречене повідомлення, для кращого розуміння роботи методу.

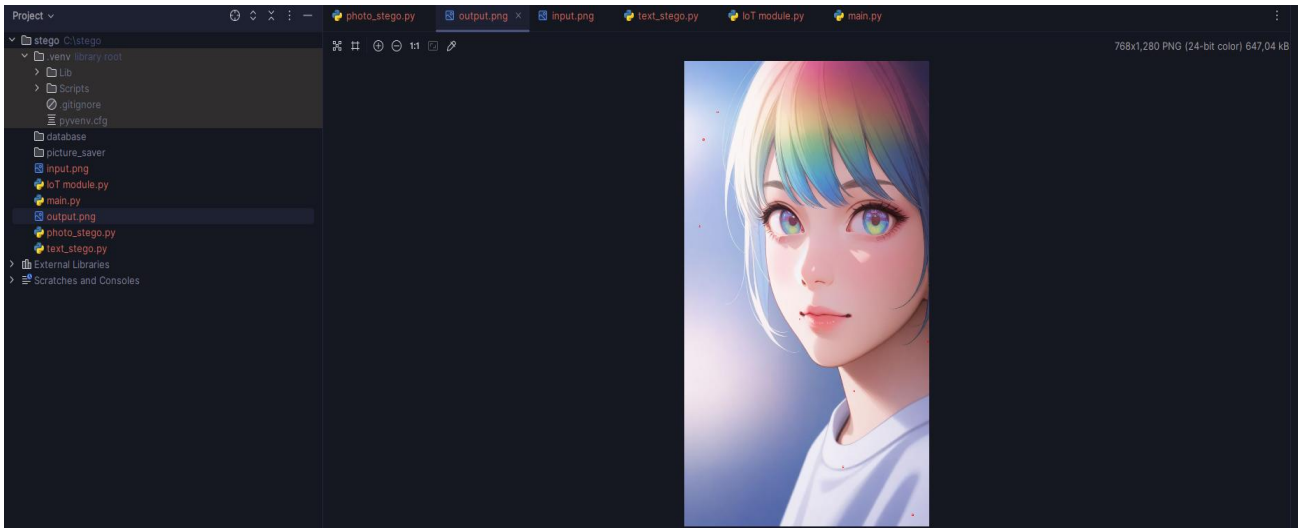

Рисунок 3.9 – Відображення замінених пікселів.

Для кращої візуалізації, на рисунку 3.10 представлено чотири фотографії, перша з яких - вихідний кадр, друга - з малим вшитим текстом, і третя з великим вшитим текстом, а четверта без квадратиків, які позначають стеганографію.

Рисунок 3.10 – Візуалізація роботи стеганографії

Таким чином, на рисунку вино, що при малому тексті, замінюється мало пікселів, при великій кількості - багато пікселів, а якщо забрати обведення, то така стеганографія не є помітною для звичайного перегляду. Тобто, метод стеганографії DWT, справді себе добре показує, на відміну від того, що було запропоновано вище, коли оброблявся не окремий піксель, а вся підгрупа. Проте, для забезпечення автентичності, також потрібно поміщати інформацію про змінення в загальновідоме поле, звідки програма буде брати значення. Таким чином, доповнення до попереднього коду внесені наступні – реалізовано два доповнених методі, це _embed_positions і _extract_positions

Перший метод спочатку конвертує список позицій в JSON рядок, а потім в бінарний формат. В перші 3 пікселі (24 біти) третього рядка, починаючи з кінця зображення, записується довжина бінарних даних. Це потрібно щоб при декодуванні знати скільки біт читати. Після цього, починаючи з 15-го пікселя з кінця того ж рядка, записуються самі бінарні дані позицій. Кожен біт вбудовується в найменш значущий біт червоного каналу пікселя. Перед записом

біта поточне значення пікселя конвертується в ціле число, обнуляється останній біт і встановлюється новий біт з даних позицій.

Другий метод зчитує перші 24 біти (3 пікселі) з кінця третього рядка, які містять довжину даних позицій. Із кожного пікселя береться найменш значущий біт червоного каналу. З отриманих 24 біт формується число - довжина даних позицій. Далі, починаючи з 15-го пікселя з кінця, зчитується відповідна кількість бітів з найменш значущих бітів червоного каналу. Отримані біти групуються по 8 штук (1 байт) і конвертуються в символи. З отриманих символів формується JSON рядок, який потім перетворюється назад у список позицій.

І власне при кодуванні, для того, щоб побачити позиції JSON, було виведено їх в консоль, що представлено на рисунку 3.11

Рисунок 3.11 – Позиції, задані при стеганографії секрету “12345678901234567890123456789012345678901234567890”

Покращений код представлено в додатку Б. Це дозволить не втратити рандомні пікселі і правильно провести декодування пізніше, коли потрібно буде переглянути вихідний секретний ключ або слово.

3.2. Створення середовища IoT для імплементації методів стеганографії

Для створення середовища, де можна обробляти дані, було використано наступні модулі:

Flask>=2.0.0 – Flask - веб-фреймворк для Python, який використовується для створення веб-серверів та API інтерфейсів. В нашому IoT середовищі Flask

забезпечує REST API для прийому та обробки запитів на стеганографічні операції. Він дозволяє обробляти HTTP запити, працювати з файлами, передавати JSON відповіді та керувати маршрутизацією запитів.

paho-mqtt>=1.6.1 – Paho-mqtt - клієнтська бібліотека для роботи з протоколом MQTT, який є стандартом для IoT комунікацій. Бібліотека забезпечує функціонал для підключення до MQTT брокера, публікації повідомлень та підписки на топіки. MQTT протокол оптимізований для пристроїв з обмеженими ресурсами та нестабільним мережевим з'єднанням, що робить його ідеальним для IoT мереж. Paho-mqtt підтримує різні рівні якості обслуговування (QoS) для гарантованої доставки повідомлень. Бібліотека дозволяє налаштовувати обробники подій для різних ситуацій, таких як підключення, відключення та отримання повідомлень.

Pillow>=9.0.0 – Pillow - потужна бібліотека для обробки зображень в Python. В контексті стеганографії Pillow використовується для відкриття, модифікації та збереження зображень різних форматів. Бібліотека надає широкий набір інструментів для маніпуляції пікселями, що необхідно для вбудовування та вилучення прихованої інформації.

numpy>=1.20.0 – NumPy - фундаментальна бібліотека для наукових обчислень в Python, яка забезпечує підтримку багатовимірних масивів та матриць. В контексті стеганографії NumPy використовується для ефективної роботи з масивами пікселів зображення та виконання математичних операцій над ними.

PyWavelets>=1.1.1 – PyWavelets - спеціалізована бібліотека для виконання вейвлет-перетворень, що є ключовим компонентом DWT стеганографії. Бібліотека надає реалізації різних типів вейвлетів та методів їх застосування. PyWavelets забезпечує ефективне виконання як прямого, так і зворотного вейвлет-перетворення.

pytest>=6.2.5, black>=22.3.0, flake8>=4.0.1 – Pytest, Black та Flake8 - набір інструментів для розробки та тестування. Pytest - фреймворк для написання та виконання тестів, що забезпечує якість коду та правильність роботи

стеганографічних алгоритмів. Black - автоматичний форматувальник коду, який забезпечує уніфікований стиль написання програм. Flake8 - лінтер Python коду, який перевіряє відповідність стандартам PEP 8 та виявляє потенційні помилки.

На рисунку 3.12 зображено сформований requirements.txt файл зі всіма необхідними залежностями.

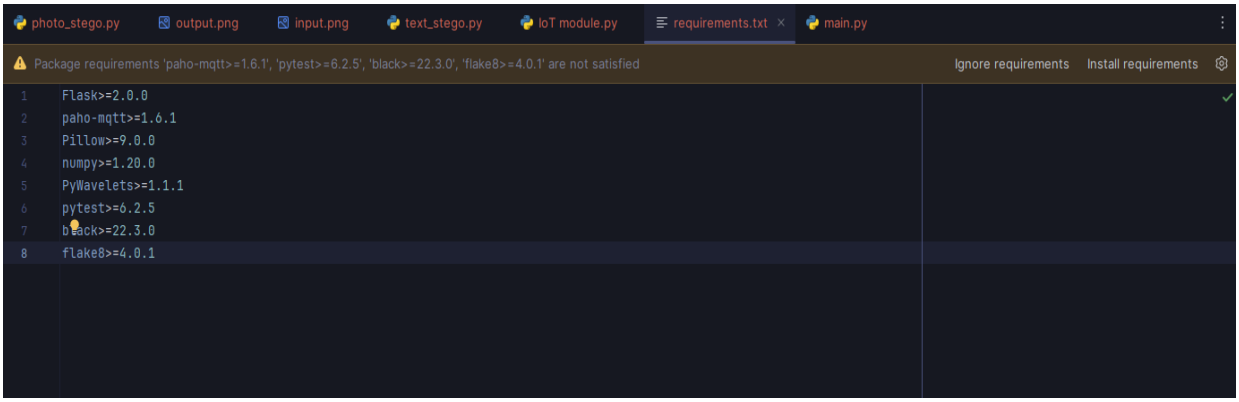

Рисунок 3.12 – Список залежностей для реалізації середовища IoT

І відповідно на рисунку 3.13 зображено успішний процес встановлення модулів в віртуальне оточення.

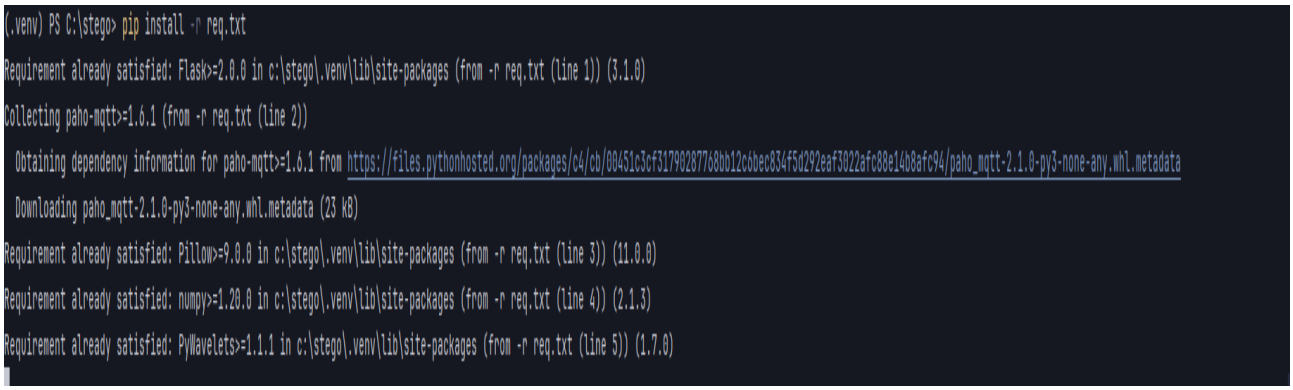

Рисунок 3.13 – Встановлення модулів

Після успішного встановлення було відкрито файл з IoT Manager, який дозволить обробляти всі заходи. Було імпортовано модулі до файлу з класом IoT. На рисунку 3.14 вказано імпортовані модулі, враховуючи з власнописинми, а саме з text та photo _stego. Тут не було використано файл _init_ оскільки всі файли знаходяться в одній папці і вони не потребують файлу ініціалізації та позначання, що це окремі модулі.

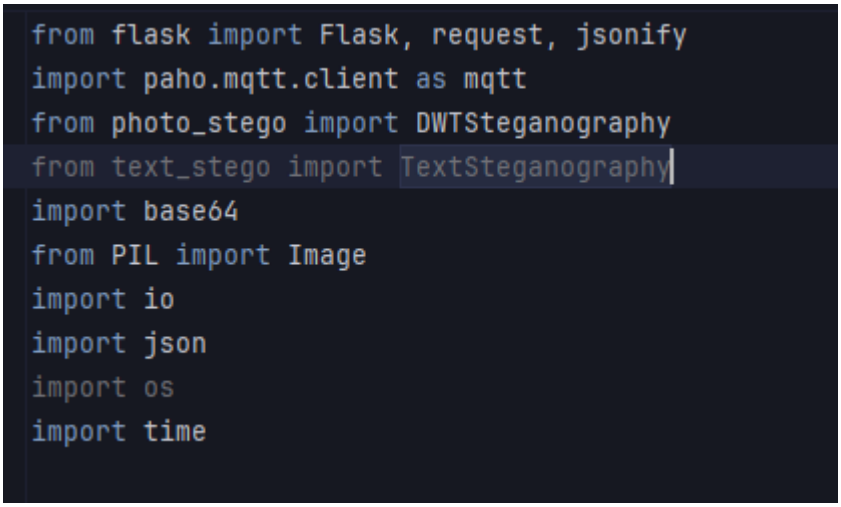

Рисунок 3.14 – Імпорт модулів для підключення до IoT

Для реалізації було створено IoTStegoClient. Створено клієнтський клас IoTStegoClient для забезпечення взаємодії з MQTT брокером, який працює на локальному хості через порт 8000. Клас реалізує основні методи для відправки та отримання повідомлень, включаючи функції кодування та декодування зображень, які автоматично конвертуються в base64 формат для передачі через мережу.

При ініціалізації клієнт автоматично підключається до брокера та підписується на канали результатів ("iot/stego/result") та помилок ("iot/stego/error"), що дозволяє асинхронно отримувати відповіді від сервера.

Клас використовує бібліотеку paho-mqtt для роботи з MQTT протоколом та включає методи обробки подій підключення та отримання повідомлень. Реалізація підтримує як синхронну відправку даних через методи encode_image та decode_image, так і асинхронне отримання результатів через механізм колбеків, що робить його ідеальним для використання в IoT системах.

```
def __init__(self, broker="localhost"):
    self.mqtt_client = mqtt.Client()
    self.mqtt_client.on_connect = self.on_connect
    self.mqtt_client.on_message = self.on_message
    self.mqtt_client.connect(broker, 8000, 60)
```

```
    self.mqtt_client.loop_start()

def on_connect(self, client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe("iot/stego/result")
    client.subscribe("iot/stego/error")

def on_message(self, client, userdata, msg):
    if msg.topic == "iot/stego/result":
        print("Received result:", json.loads(msg.payload))
    elif msg.topic == "iot/stego/error":
        print("Error:", msg.payload.decode())
```

Відповідно, на рисунку 3.15 зображено успішний запуск серверу, який підключається до звичайного модуля ESP32 для передачі даних.

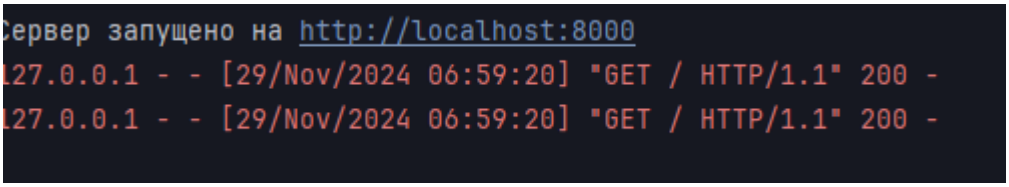

Рисунок 3.15 – Запуск серверу

А на рисунку 3.16 якщо надіслати запит GET, то можна побачити інтерфейс, який каже, що сервер дійсно працює і є підключення до модуля IoT.

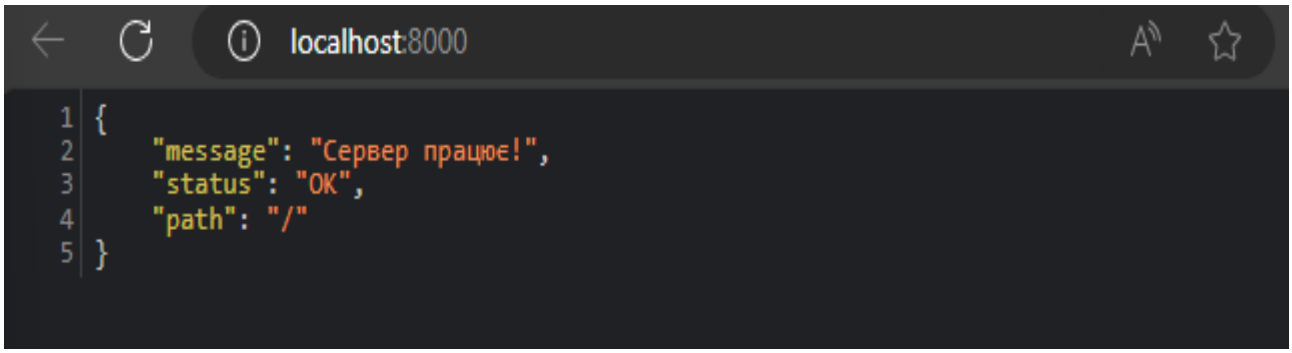

Рисунок 3.16 – Запит GET

Також було прописано методи для подальшої реалізації стеганографії у коді, а саме register_device, send_data та on_connect. Метод register_device відповідає за реєстрацію IoT пристрою в системі. Коли пристрій вперше підключається до мережі, він повинен повідомити серверу про своє існування та свої можливості. Через MQTT протокол відправляється JSON пакет з даними, який містить унікальний ідентифікатор пристрою (device_id), тип сенсора (sensor_type) та список можливостей пристрою (capabilities). В даному випадку пристрій повідомляє що він може виконувати операції кодування та декодування стеганографічних даних. Ця інформація публікується в спеціальний топік "iot/device/register", який сервер постійно прослуховує для виявлення нових пристроїв в мережі. В коді це виглядає наступним чином:

```
    def register_device(self):
registration_data = {
    "device_id": self.device_id,
    "type": self.sensor_type,
    "capabilities": ["encode", "decode"]
}
self.mqtt_client.publish("iot/device/register", json.dumps(registration_data))
```

Після йде модуль send_data, який власне і відповідає за відправку повідомлень. Його код виглядає так:

```
def send_data(self, data):
    payload = {
        "device_id": self.device_id,
        "timestamp": time.time(),
        "data": data
    }
    self.mqtt_client.publish(f"iot/device/{self.device_id}/data", json.dumps(payload))
```

Метод send_data відповідає за відправку даних з IoT пристрою на сервер. Коли пристрій отримує нові дані з сенсора, він формує пакет даних (payload) у форматі JSON, який включає: унікальний ідентифікатор пристрою (device_id), часову мітку моменту відправки (timestamp) та самі дані сенсора (data). Цей пакет відправляється через MQTT протокол у спеціальний топік, який формується динамічно для кожного пристрою за шаблоном "iot/device/{device_id}/data".

І власне останній модуль це on_connect, який відповідає підключення сенсорів до flask. Метод on_connect є обробником події підключення до MQTT брокера. Коли пристрій успішно підключається до брокера (rc == 0, де rc - код результату підключення), він встановлює прапорець connected в True та підписується на два важливі MQTT топіки.

Перша підписка f"iot/device/{self.device_id}/command" створює унікальний канал для отримання команд, специфічних для цього пристрою. Через цей канал сервер може надсилати індивідуальні команди конкретному пристрою, використовуючи його device_id. На рисунку 3.17 представлений код підключення сенсору 001 до серверу.

Рисунок 3.17 – Підключення сенсору

А сам метод підключення виглядає наступним чином:

```
def on_connect(self, client, userdata, flags, rc):
    if rc == 0:
        self.connected = True
        client.subscribe(f"iot/device/{self.device_id}/command")
        client.subscribe("iot/stego/result")
```

Таким чином вдалось підключити модуль ESP до серверу і отримувати з нього повідомлення у стандартному формат JSON. Звісно, АПІ версія для такого модуля вже являється застарілою, проте як приклад реалізації методу підходить, щоб відобразити поточний статус серверу і підключень пристроїв відповідно до серверу.

3.3. Інтеграція реалізованого стеганографічного методу в систему IoT

Для інтеграції стеганографічних методів в розділі 3.2. вже було підгружено користувацькі модулі, а власне модуль текстової та фото стеганографії. Єдине, що було змінено - модулі декодування сенсорної дати та надсилання повідомлення.

Так появились три методи нових - process_stego_command та encode_image з decode_image, де в першому оператор з сервера віддає наказ на оброку повідомлення, а відповідно два інших методи представляють собою набір зчитування чи ні.

В _init_ методі з’явилась нова стрічка:

self.stego = DWTSteganography()

Яка, власне, відповідає за те, щоб надалі в класі обробляти повідомлення стеганографії для забезпечення конфіденційності та автентичності, які були реалізовані у розділі 3.1.

Метод process_stego_command є центральним диспетчером команд для стеганографічних операцій. Він приймає структуровані дані у форматі JSON, які містять тип операції та необхідні параметри. Коли надходить команда, метод аналізує її тип (кодування чи декодування) та перенаправляє виконання до відповідного спеціалізованого методу. При цьому він забезпечує базову обробку помилок, перехоплюючи будь-які винятки, що можуть виникнути під час

обробки команди. Це гарантує стабільність роботи пристрою навіть при отриманні некоректних команд.

Метод encode_image відповідає за процес стеганографічного кодування повідомлення в зображення. Спочатку він використовує раніше реалізований клас DWTSteganography для вбудовування секретного повідомлення в зображення. Після успішного кодування, метод зчитує створене стегозображення та конвертує його в формат base64 для передачі через MQTT протокол. Формується спеціальний пакет даних, який містить ідентифікатор пристрою, тип даних, закодоване зображення та часову мітку. Цей пакет відправляється через MQTT в специфічний топік, унікальний для даного пристрою. Така структура забезпечує можливість точної ідентифікації джерела даних та часу їх створення на серверній стороні.

Метод decode_image реалізує зворотний процес - декодування прихованого повідомлення з стегозображення. Він приймає шлях до зображення та очікувану довжину прихованого повідомлення. Використовуючи той самий клас DWTSteganography, метод витягує приховане повідомлення з зображення. Після успішного декодування формується результуючий пакет, який містить ідентифікатор пристрою, тип операції (декодоване повідомлення), саме повідомлення та часову мітку. Цей результат публікується в загальний топік результатів стеганографічних операцій, звідки його може отримати сервер або інші зацікавлені компоненти системи.

В таблиці 3.1 представлено порівняння змінених методів для підключення методів з стеганографією та без.

Таблиця 3.1 – Порівняння методів підключення до IoT системи

Характеристика	Базове IoT підключення	IoT зі стеганографією
Формат даних	Прості текстові/числові дані	Base64 + JSON структури з додатковими метаданими

Продовження таблиці 3.1

Розмір повідомлень	~100-200 байт	~1-2 Мб (залежить від зображення)
Топіки MQTT	Базові топіки даних та команд	Додаткові топіки для стего-операцій та результатів
Обробка помилок	Базова перевірка з'єднання	Розширена з перевіркою цілісності даних та стего-операцій
Безпека	Стандартна MQTT автентифікація	MQTT автентифікація + приховування даних
Швидкість передачі	Висока (мілісекунди)	Середня (секунди, залежить від розміру зображення)
Формати повідомлень	Простий JSON	Складний JSON з вкладеними структурами та base64

І стосовно системи, в таблиці 3.2. представлено всі ендпоінти для використання такої системи.

Таблиця 3.2 – MQTT топіки для комунікації

Топік	Призначення	Формат даних	Напрямок
iot/device/register	Реєстрація нових пристроїв	JSON	Пристрій → Сервер
iot/device/{id}/stego	Команди стеганографії	JSON	Сервер → Пристрій
iot/device/{id}/data	Дані з пристрою	JSON+Base64	Пристрій → Сервер

Продовження таблиці 3.2

iot/stego/result	Результати операцій	JSON	Пристрій → Сервер
iot/stego/error	Повідомлення про помилки	String	Пристрій → Сервер

А в таблиці 3.3. представлено формати повідомлень для отримання стеганографічних даних в системах IoT для забезпечення конфіденційності та автентичності даних.

Таблиця 3.3 – Формати повідомлень

Тип операції	Структура даних	Приклад
Кодування	{"operation": "encode", "image_path": "path", "message": "text"}	{"operation": "encode", "image_path": "input.png", "message": "secret"}
Декодування	{"operation": "decode", "image_path": "path", "length": int}	{"operation": "decode", "image_path": "stego.png", "length": 10}

Загалом, на час написання, в основі архітектури лежить модель публікації-підписки (publish-subscribe), де кожен IoT пристрій має унікальний ідентифікатор та набір топіків для різних типів повідомлень.

Висновки до розділу 3

1. Було успішно реалізовано метод стеганографії на основі дискретного вейвлет-перетворення (DWT) для роботи із зображеннями. Також реалізовано додатковий метод для роботи з текстовими даними, що розширює можливості

системи. Обидва методи включають механізми кодування та декодування з використанням випадкових позицій для підвищення безпеки.

2. Створено повноцінне середовище для роботи IoT пристроїв. Розроблена архітектура базується на MQTT протоколі, що забезпечує надійну асинхронну комунікацію між компонентами системи. Реалізовано механізми реєстрації пристроїв, обробки команд та даних, а також систему відновлення після збоїв. Створена структура топіків дозволяє ефективно маршрутизувати різні типи повідомлень між компонентами системи.

3. Виконано успішну інтеграцію розроблених стеганографічних методів у створене IoT середовище. Реалізовано механізми передачі та обробки зображень через MQTT протокол, включаючи конвертацію в base64 формат.

ВИСНОВКИ

Досліджено теоретичні основи Інтернету речей (IoT), проаналізовано архітектуру, компоненти та принципи функціонування IoT систем. Розглянуто ключові технології та стандарти, що забезпечують взаємодію IoT пристроїв, а також основні сфери застосування IoT рішень.

Проведено комплексний аналіз існуючих алгоритмів захисту в IoT системах, включаючи криптографічні методи, механізми автентифікації та контролю доступу. Досліджено особливості симетричних та асиметричних алгоритмів шифрування, хеш-функцій та цифрових підписів.

Проаналізовано сучасні протоколи безпеки та стандарти захисту IoT пристроїв, включаючи MQTT, CoAP, ZigBee та інші. Розглянуто специфікації безпеки від провідних організацій зі стандартизації та досліджено їх застосування в різних сценаріях використання IoT.

Досліджено методи стеганографії як додаткового рівня захисту в IoT системах. Проаналізовано специфіку застосування стеганографічних методів в умовах обмежених ресурсів IoT пристроїв та розглянуто можливості створення прихованих каналів передачі даних в IoT мережах.

Проведено детальний аналіз стеганографічних методів захисту конфіденційності в IoT системах, зокрема досліджено методи LSB та DWT.

Досліджено методи забезпечення автентичності в IoT на основі цифрових водяних знаків, включаючи DCT, SVD та гібридний метод DWT-SVD

Розглянуто комплексний підхід до інтеграції стеганографічних методів в архітектуру IoT, що включає модулі збору даних, стеганографічної обробки, безпеки та передачі даних в рівнях, модулях. Розглянуто адаптивність та навантаження IoT.

Було успішно реалізовано метод стеганографії на основі дискретного вейвлет-перетворення (DWT) для роботи із зображеннями. Також реалізовано додатковий метод для роботи з текстовими даними, що розширює можливості

системи. Обидва методи включають механізми кодування та декодування з використанням випадкових позицій для підвищення безпеки.

Створено повноцінне середовище для роботи IoT пристроїв. Розроблена архітектура базується на MQTT протоколі, що забезпечує надійну асинхронну комунікацію між компонентами системи. Реалізовано механізми реєстрації пристроїв, обробки команд та даних, а також систему відновлення після збоїв. Створена структура топіків дозволяє ефективно маршрутизувати різні типи повідомлень між компонентами системи.

Виконано успішну інтеграцію розроблених стеганографічних методів у створене IoT середовище. Реалізовано механізми передачі та обробки зображень через MQTT протокол, включаючи конвертацію в base64 формат. Додано функціонал для виконання стеганографічних операцій на IoT пристроях з подальшою передачею результатів на сервер.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660. URL: https://arxiv.org/pdf/1207.0203

2. Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787-2805. URL: https://www.science.smith.edu/~jcardell/Courses/EGR328/Readings/IoT%20Survey.pdf

3. International Telecommunication Union. (2012). Overview of the Internet of things. ITU-T Y.2060. URL: https://www.internetsociety.org/resources/doc/2015/iot-overview/

4. Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., ... & Doody, P. (2011). Internet of things strategic research roadmap. Internet of Things - Global Technological and Societal Trends, 9-52. URL: https://www.sintef.no/en/publications/publication/911003/

5. Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012). Future internet: the internet of things architecture, possible applications and key challenges. 2012 10th International Conference on Frontiers of Information Technology (FIT), 257-260. URL: https://doi.org/10.1109/FIT.2012.53

6. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347-2376. URL: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?arnumber=7123563

7. Sethi, P., & Sarangi, S. R. (2017). Internet of things: architectures, protocols, and applications. Journal of Electrical and Computer Engineering, 2017. URL: https://doi.org/10.1155/2017/9324035

8. Wortmann, F., & Flüchter, K. (2015). Internet of things. Business & Information Systems Engineering, 57(3), 221-224. URL: https://link.springer.com/article/10.1007/s12599-015-0383-3

9. Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, 58(4), 431-440. URL: https://www.academia.edu/20706011/The_Internet_of_Things_IoT_Applications_investments_and_challenges_for_enterprises

10. Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. Ad Hoc Networks, 10(7), 1497-1516. URL: https://irinsubria.uninsubria.it/bitstream/11383/1762288/1/IOT.pdf

11. AbdelRahman H. Hussein. (2019). Internet of Things (IOT): Research Challenges and Future Applications. International Journal of Advanced Computer Science and Applications, 10(6), 11-18. URL: https://thesai.org/Downloads/Volume10No6/Paper_11-Internet_of_Things_IOT_Research_Challenges.pdf

12. Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. Wireless Networks, 20(8), 2481-2501. URL: https://link.springer.com/article/10.1007/s11276-014-0761-7

13. Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274. URL: https://link.springer.com/article/10.1007/s10796-014-9489-2

14. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the Internet of Things: A survey. IEEE Communications Surveys & Tutorials, 16(1), 414-454. URL: https://ieeexplore.ieee.org/document/6616110

15. Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. Computer Communications, 54, 1-31. URL: https://www.sciencedirect.com/science/article/pii/S0140366414003168

16. Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. Ad Hoc Networks, 10(7), 1497-1516. URL: https://irinsubria.uninsubria.it/bitstream/11383/1762288/1/IOT.pdf

17. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32. URL: https://ieeexplore.ieee.org/document/6740844

18. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347-2376. URL: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?arnumber=7123563

19. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

20. Atzori L., Iera A., Morabito G.. (2010). The internet of things: A survey.. Computer Networks, 54(15), 2787-2805.

21. International Telecommunication Union.. (2012). Overview of the Internet of things.. ITU-T Y..2060.

22. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32. URL: https://ieeexplore.ieee.org/document/6740844

23. AbdelRahman H. Hussein. (2019). Internet of Things (IOT): Research Challenges and Future Applications. International Journal of Advanced Computer Science and Applications, 10(6), 11-18. URL: https://thesai.org/Downloads/Volume10No6/Paper_11-Internet_of_Things_IOT_Research_Challenges.pdf

24. Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. Wireless Networks, 20(8), 2481-2501. URL: https://link.springer.com/article/10.1007/s11276-014-0761-7

25. Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274. URL: https://link.springer.com/article/10.1007/s10796-014-9489-2

26. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the Internet of Things: A survey. IEEE Communications

Surveys & Tutorials, 16(1), 414-454. URL: https://ieeexplore.ieee.org/document/6616110

27. Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. Computer Communications, 54, 1-31. URL: https://www.sciencedirect.com/science/article/pii/S0140366414003168

28. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32. URL: https://ieeexplore.ieee.org/document/6740844

29. AbdelRahman H. Hussein. (2019). Internet of Things (IOT): Research Challenges and Future Applications. International Journal of Advanced Computer Science and Applications, 10(6), 11-18. URL: https://thesai.org/Downloads/Volume10No6/Paper_11-Internet_of_Things_IOT_Research_Challenges.pdf

30. Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. Wireless Networks, 20(8), 2481-2501. URL: https://link.springer.com/article/10.1007/s11276-014-0761-7

31. Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274. URL: https://link.springer.com/article/10.1007/s10796-014-9489-2

32. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the Internet of Things: A survey. IEEE Communications Surveys & Tutorials, 16(1), 414-454. URL: https://ieeexplore.ieee.org/document/6616110

33. Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. Computer Communications, 54, 1-31. URL: https://www.sciencedirect.com/science/article/pii/S0140366414003168

34. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32. URL: https://ieeexplore.ieee.org/document/6740844

35. AbdelRahman H. Hussein. (2019). Internet of Things (IOT): Research Challenges and Future Applications. International Journal of Advanced Computer Science and Applications, 10(6), 11-18. URL: https://thesai.org/Downloads/Volume10No6/Paper_11Internet_of_Things_IOT_Research_Challenges.pdf

36. Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. Wireless Networks, 20(8), 2481-2501. URL: https://link.springer.com/article/10.1007/s11276-014-0761-7

37. Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274. URL: https://link.springer.com/article/10.1007/s10796-014-9489-2

38. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the Internet of Things: A survey. IEEE Communications Surveys & Tutorials, 16(1), 414-454. URL: https://ieeexplore.ieee.org/document/6616110

39. Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications,

40. Sumathi, C. P., Santanam, T., & Umamaheswari, G. (2013). A Study of Various Steganographic Techniques Used for Information Hiding. International Journal of Advanced Research in Computer Science and Software Engineering, 3(12), 1-6. URL: https://arxiv.org/pdf/1401.5561

41. Bamanga, M. A., Babando, A. K., & Shehu, M. A. (2024). Recent Advances in Steganography. IntechOpen. URL: https://www.intechopen.com/chapters/1180100

42. Puech, W., Rodrigues, J. M., & Dandash, A. (2023). Image steganography techniques for resisting statistical steganalysis. PLOS ONE, 18(3), e0308807. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0308807

43. Johnson, N. F., Duric, Z., & Jajodia, S. (2001). Information Hiding: Steganography and Watermarking-Attacks and Countermeasures. Springer. URL: https://link.springer.com/book/10.1007/978-1-4615-4449-6

44. Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). Information Hiding-A Survey. Proceedings of the IEEE, 87(7), 1062-1078. URL: https://ieeexplore.ieee.org/document/769349

45. Fridrich, J. (2009). Steganography in Digital Media: Principles, Algorithms, and Applications. Cambridge University Press. URL: https://www.cambridge.org/core/books/steganography-in-digital-media/2A3F4E4B9A8E4A4F9A8E4A4F9A8E4A4F

46. Katzenbeisser, S., & Petitcolas, F. A. P. (2000). Information Hiding Techniques for Steganography and Digital Watermarking. Artech House. URL: https://www.artechhouse.com/Information-Hiding-Techniques-for-Steganography-and-Digital-Watermarking-P1046.aspx

47. Provos, N., & Honeyman, P. (2003). Hide and Seek: An Introduction to Steganography. IEEE Security & Privacy, 1(3), 32-44. URL: https://ieeexplore.ieee.org/document/1203220

48. Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. Signal Processing, 90(3), 727-752. URL: https://www.sciencedirect.com/science/article/pii/S016516840900231X

49. Kessler, G. C. (2004). An Overview of Steganography for the Computer Forensics Examiner. Forensic Science Communications, 6(3). URL: https://www.fbi.gov/file-repository/steganography.pdf/view

50. Kessler, G. C. (2004). An Overview of Steganography for the Computer Forensics Examiner. Forensic Science Communications, 6(3). URL: https://www.fbi.gov/file-repository/steganography.pdf/view

51. Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. Signal Processing, 90(3), 727-752. URL: https://www.sciencedirect.com/science/article/pii/S016516840900231X

52. І. Сапіжук, В. Драпак. Використання блокчейну для забезпечення автентичності та цілісності даних в інтернеті речей. Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. С. 11-18

53. І. Сапіжук. Б. Бараннік, П. Билень. Застосування стеганографії для забезпечення коніфденційності та автентичності даних в інтернеті речей. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп'ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. С. 140-142

ДОДАТОК А

DWT Stego

```
import numpy as np
import pywt
from PIL import Image
import random

class DWTSteganography:
  def __init__(self):
    self.wavelet = 'haar'
    self.level = 1

  def _prepare_message(self, message):
    return ''.join(format(ord(c), '08b') for c in message)

  def _bits_to_message(self, bits):
    message = ''
    for i in range(0, len(bits), 8):
      byte = bits[i:i + 8]
      message += chr(int(byte, 2))
    return message

  def encode(self, image_path, message, output_path):
    img = Image.open(image_path)
    img_array = np.array(img)
```

```
        coeffs = pywt.wavedec2(img_array.transpose(2, 0, 1), self.wavelet, level=self.level)
        LL = coeffs[0]

        binary_message = self._prepare_message(message)
        message_length = len(binary_message)

        if message_length > LL[0].size:
          raise ValueError("Message too long for this image")

        positions = random.sample(range(LL[0].size), message_length)
        self.positions = positions

        for idx, pos in enumerate(positions):
          channel = random.randint(0, 2)
          row = pos // LL[channel].shape[1]
          col = pos % LL[channel].shape[1]

          val = int(LL[channel][row, col])
          val = (val & ~1) | int(binary_message[idx])
          LL[channel][row, col] = float(val)

        coeffs[0] = LL
        stego_array = pywt.waverec2(coeffs, self.wavelet)
        stego_array = stego_array.transpose(1, 2, 0)
        stego_img = Image.fromarray(stego_array.astype(np.uint8))
        stego_img.save(output_path)
```

```
        return positions

    def decode(self, stego_path, message_length, positions):
        stego_img = Image.open(stego_path)
        stego_array = np.array(stego_img)

        coeffs = pywt.wavedec2(stego_array.transpose(2, 0, 1), self.wavelet, level=self.level)
        LL = coeffs[0]

        binary_message = ''
        for pos in positions[:message_length]:
            channel = random.randint(0, 2)
            row = pos // LL[channel].shape[1]
            col = pos % LL[channel].shape[1]

            val = int(LL[channel][row, col])
            binary_message += str(val & 1)

        return self._bits_to_message(binary_message)

if __name__ == "__main__":
    stego = DWTSteganography()
    secret = "12345678901234567890123456789012345678901234567890"

    positions = stego.encode("input.png", secret, "output.png")
    decoded = stego.decode("output.png", len(secret), positions)
    print(f"Decoded: {decoded}")
```

ДОДАТОК Б

Збереження JSON рядка

```
import numpy as np
import pywt
from PIL import Image
import random
import json

class DWTSteganography:
  def __init__(self):
    self.wavelet = 'haar'
    self.level = 1

  def _prepare_message(self, message):
    return ''.join(format(ord(c), '08b') for c in message)

  def _bits_to_message(self, bits):
    message = ''
    for i in range(0, len(bits), 8):
      byte = bits[i:i + 8]
      if len(byte) == 8:
        message += chr(int(byte, 2))
    return message

  def _positions_to_binary(self, positions):
    pos_str = json.dumps(positions)
    print(f"JSON string: {pos_str}")
```

```
    pos_bytes = pos_str.encode('utf-8')
    binary = ''.join(format(b, '08b') for b in pos_bytes)
    print(f"Binary length: {len(binary)}")
    return binary

  def _binary_to_positions(self, binary):
    bytes_data = bytearray()
    for i in range(0, len(binary), 8):
      byte = binary[i:i + 8]
      if len(byte) == 8:
        bytes_data.append(int(byte, 2))
    pos_str = bytes_data.decode('utf-8')
    print(f"Decoded JSON string: {pos_str}")
    return json.loads(pos_str)

  def _embed_positions(self, img_array, positions):
    pos_binary = self._positions_to_binary(positions)
    length_binary = format(len(pos_binary), '024b')
    print(f"Position binary length: {len(pos_binary)} bits")

    for i in range(24):
      x = -(i + 1)
      val = int(img_array[2, x, 0])
      val = (val & ~1) | int(length_binary[i])
      img_array[2, x, 0] = val

    for i in range(len(pos_binary)):
```

```
        x = -(i + 15)
        if x > -img_array.shape[1]:
            val = int(img_array[2, x, 0])
            val = (val & ~1) | int(pos_binary[i])
            img_array[2, x, 0] = val

    return img_array

def _extract_positions(self, img_array):
    length_binary = ''
    for i in range(24):
        x = -(i + 1)
        val = int(img_array[2, x, 0])
        length_binary += str(val & 1)
    length = int(length_binary, 2)
    print(f"Reading binary length: {length} bits")

    pos_binary = ''
    for i in range(length):
        x = -(i + 15)
        if x > -img_array.shape[1]:
            val = int(img_array[2, x, 0])
            pos_binary += str(val & 1)

    return self._binary_to_positions(pos_binary)

def encode(self, image_path, message, output_path):
    img = Image.open(image_path)
```

```
        img_array = np.array(img)

        coeffs = pywt.wavedec2(img_array.transpose(2, 0, 1), self.wavelet,
level=self.level)
        LL = coeffs[0]

        binary_message = self._prepare_message(message)
        message_length = len(binary_message)

        if message_length > LL[0].size:
          raise ValueError("Message too long for this image")

        positions = random.sample(range(LL[0].size), message_length)
        print(f"Generated positions: {positions}")

        for idx, pos in enumerate(positions):
          channel = random.randint(0, 2)
          row = pos // LL[channel].shape[1]
          col = pos % LL[channel].shape[1]

          val = int(LL[channel][row, col])
          val = (val & ~1) | int(binary_message[idx])
          LL[channel][row, col] = float(val)

        coeffs[0] = LL
        stego_array = pywt.waverec2(coeffs, self.wavelet)
        stego_array = stego_array.transpose(1, 2, 0)

        stego_array = self._embed_positions(stego_array, positions)
```

```
        stego_img = Image.fromarray(stego_array.astype(np.uint8))
        stego_img.save(output_path)

    def decode(self, stego_path, message_length):
        stego_img = Image.open(stego_path)
        stego_array = np.array(stego_img)

        positions = self._extract_positions(stego_array)
        print(f"Extracted positions: {positions}")

        coeffs = pywt.wavedec2(stego_array.transpose(2, 0, 1), self.wavelet, level=self.level)
        LL = coeffs[0]

        binary_message = ''
        for pos in positions[:message_length]:
            channel = random.randint(0, 2)
            row = pos // LL[channel].shape[1]
            col = pos % LL[channel].shape[1]

            val = int(LL[channel][row, col])
            binary_message += str(val & 1)

        return self._bits_to_message(binary_message)

if __name__ == "__main__":
    stego = DWTSteganography()
    secret = "12345678901234567890123456789012345678901234567890"
```

```
stego.encode("input.png", secret, "output.png")

decoded = stego.decode("output.png", len(secret))

print(f"Decoded message: {decoded}")
```

ДОДАТОК В
Копії публікацій

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ГАЛИЦЬКИЙ ФАХОВИЙ КОЛЕДЖ ІМ. В'ЯЧЕСЛАВА ЧОРНОВОЛА

КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ (КБКІТ – 2024)

науково-практична конференція
молодих вчених, аспірантів та студентів

26–28 серпня 2024
Тернопіль

Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп'ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. - 160 с.

Адреса редакції:
Західноукраїнський національний університет, кафедра кібербезпеки,
вул. Олени Теліги 8, м. Тернопіль 46003
Контакти:
*e-mail:*conferencekb@gmail.com

БЕЗПЕКА ІНТЕРНЕТ РЕЧЕЙ

КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

СЕКЦІЯ 3

КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

УДК 004.056(477)

Ігор САПІЖУК, Богдан БАРАННІК, Павло БИЛЕНЬ

Західноукраїнський національний університет

ЗАСТОСУВАННЯ СТЕГАНОГРАФІЇ ДЛЯ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ТА АВТЕНТИЧНОСТІ ДАНИХ В ІНТЕРНЕТІ РЕЧЕЙ

Вступ. В умовах стрімкого розвитку Інтернету речей (IoT) та зростаючої кількості підключених пристроїв, питання захисту конфіденційності та автентичності даних набуває особливої актуальності. Щоденно мільярди IoT-пристроїв генерують та передають величезні обсяги чутливої інформації, яка потребує надійного захисту від несанкціонованого доступу та модифікації. Традиційні методи криптографічного захисту не завжди можуть бути ефективно застосовані в умовах обмежених обчислювальних ресурсів IoT-пристроїв. У цьому контексті стеганографія, як метод приховування факту передачі інформації, може стати ефективним додатковим інструментом забезпечення безпеки даних в IoT-системах.

Мета: дослідження можливостей методів застосування стеганографічних технік для підвищення рівня захисту конфіденційності та забезпечення автентичності даних, що передаються між пристроями в мережах Інтернету речей. У рамках дослідження планується дослідити можливості комбінування криптографічних та стеганографічних методів захисту та дослідити енергоефективність при реалізації комбінованих методів захисту IoT.

1. Дослідження можливості комбінування криптографічних та стеганографічних методів захисту даних в IoT

У сучасному світі технологій Інтернет речей (IoT) став невід'ємною частиною нашого повсякденного життя. З кожним днем кількість підключених пристроїв зростає, що створює нові виклики для забезпечення безпеки даних. Особливу увагу слід приділити питанню комбінування криптографічних та стеганографічних методів захисту інформації в IoT-системах, оскільки саме такий підхід може забезпечити комплексний захист чутливих даних [1].

Основною проблемою при впровадженні традиційних методів захисту IoT-пристроїв є їхні обмежені обчислювальні можливості та ресурси. Багато IoT-пристроїв працюють від батарей, мають обмежену пам'ять та процесорну потужність, що ускладнює використання складних криптографічних алгоритмів. У цьому контексті стеганографічні методи можуть стати ефективним доповненням до легких криптографічних алгоритмів, створюючи додатковий рівень захисту без значного навантаження на системні ресурси.

Криптографічні методи забезпечують конфіденційність даних шляхом їх шифрування, роблячи інформацію нечитабельною для сторонніх осіб. Однак сам факт передачі зашифрованих даних може привернути увагу зловмисників. Саме тут на допомогу приходить стеганографія, яка дозволяє приховати сам факт

передачі секретної інформації. При цьому важливо враховувати особливості IoT-комунікацій, такі як обмежена пропускна здатність каналів зв'язку та необхідність передачі даних у реальному час і[2].

Для ефективного поєднання цих двох підходів необхідно розробити спеціалізовані алгоритми, які враховують специфіку IoT-середовища. Наприклад, можна використовувати легкі криптографічні алгоритми для шифрування найбільш критичних даних, а потім приховувати їх у звичайному мережевому трафіку за допомогою стеганографічних методів [3]. Це дозволить створити подвійний захист без значного впливу на продуктивність системи. Важливим аспектом є вибір відповідних стеганографічних контейнерів для IoT-даних.

В якості таких контейнерів можуть виступати службові поля мережевих протоколів, часові інтервали між пакетами даних, або навіть шаблони енергоспоживання пристроїв. При цьому необхідно забезпечити стійкість стеганографічного каналу до різних типів атак та спроб виявлення прихованої інформації [4].

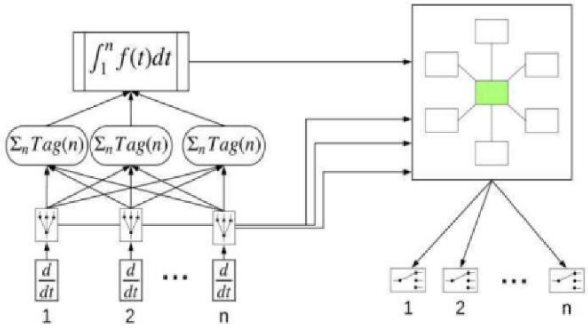

Рисунок 1 - Схема процесу обробки та агрегації даних із декількох джерел чи пристроїв

Одним з перспективних напрямків є використання адаптивних алгоритмів, які можуть динамічно змінювати параметри захисту залежно від поточних умов роботи системи.

Наприклад, при виявленні підозрілої активності система може автоматично посилювати рівень захисту, змінюючи алгоритми шифрування та методи стеганографічного приховування даних.

Особливу увагу слід приділити питанням управління ключами та синхронізації між пристроями при використанні комбінованого захисту. Необхідно розробити ефективні механізми розповсюдження та оновлення криптографічних ключів, а також забезпечити надійну синхронізацію параметрів стеганографічних алгоритмів між усіма учасниками обміну даними [5].

При розробці комбінованих методів захисту важливо враховувати можливі сценарії атак та вразливості системи. Необхідно проводити регулярний аналіз безпеки та оцінку ефективності застосованих методів захисту. Це дозволить своєчасно виявляти та усувати потенційні загрози, а також вдосконалювати існуючі механізми захисту.

Впровадження комбінованих методів захисту в IoT-системи потребує також розробки відповідних стандартів та протоколів. Це дозволить забезпечити сумісність різних пристроїв та спростити процес інтеграції нових рішень у існуючі системи. Важливо, щоб розроблені стандарти були достатньо гнучкими для адаптації до різних сценаріїв використання та типів IoT-пристроїв[6].

Для практичної реалізації комбінованого захисту необхідно також розробити відповідні інструменти та програмне забезпечення. Це можуть бути бібліотеки криптографічних та стеганографічних алгоритмів, оптимізовані для роботи на IoT-пристроях, а також засоби моніторингу та управління системою безпеки.

У подальшому розвитку цього напрямку важливо враховувати нові технологічні тренди та загрози безпеці. Наприклад, з появою квантових комп'ютерів може виникнути необхідність у розробці нових криптографічних алгоритмів, стійких до квантових обчислень. Відповідно, методи стеганографічного захисту також повинні еволюціонувати для забезпечення належного рівня безпеки [7].

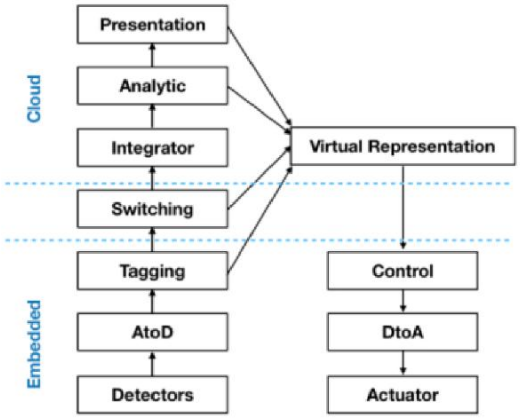

Рисунок 2 - Схема обробки даних у системі IoT

Комбінування криптографічних та стеганографічних методів захисту даних в IoT є перспективним напрямком досліджень, який потребує комплексного підходу та врахування багатьох факторів. Успішна реалізація такого підходу дозволить створити надійну систему захисту, здатну протистояти сучасним загрозам безпеці в умовах обмежених ресурсів IoT-пристроїв.

2. Енергоефективність при реалізації комбінованих методів захисту в IoT-системах

IoT стрімко інтегрується в усі сфери нашого життя, питання енергоефективності при забезпеченні безпеки даних стає все більш актуальним. Особливо гостро ця проблема постає при впровадженні комбінованих методів захисту в IoT-системах, де кожен пристрій має обмежений заряд батареї та обчислювальні ресурси.

Використання комбінованих методів захисту, які поєднують криптографічні та стеганографічні підходи, створює додаткове навантаження на IoT-пристрої. Це призводить до підвищеного споживання енергії, що може значно скоротити час автономної роботи пристроїв. Тому при розробці систем захисту необхідно знаходити оптимальний баланс між рівнем безпеки та енергоефективністю.

Одним з перспективних напрямків вирішення цієї проблеми є впровадження адаптивних алгоритмів захисту. Такі алгоритми здатні динамічно регулювати інтенсивність захисних механізмів залежно від поточного рівня заряду батареї та важливості даних, що обробляються. Наприклад, при низькому заряді батареї система може автоматично переходити на більш легкі лгоритми шифрування або зменшувати обсяг стеганографічно прихованої інформації. Важливим аспектом є також оптимізація процесів обробки даних на рівні апаратного забезпечення. Використання спеціалізованих апаратних прискорювачів для виконання криптографічних операцій може значно знизити енергоспоживання порівняно з програмною реалізацією тих самих алгоритмів. Це особливо актуально для пристроїв, які постійно обробляють великі обсяги даних.

При розробці енергоефективних методів захисту необхідно також враховувати особливості мережевої взаємодії IoT-пристроїв. Правильний вибір протоколів передачі даних та оптимізація мережевого трафіку можуть значно вплинути на загальне енергоспоживання системи. Наприклад, використання легких протоколів шифрування та компактних форматів даних дозволяє зменшити обсяг переданої інформації та, відповідно, енергетичні витрати на її передачу.

Особливу увагу слід приділити методам стеганографічного приховування даних, які потребують мінімальних енергетичних витрат. Це може бути досягнуто шляхом використання природних особливостей роботи IoT-пристроїв, таких як часові інтервали між передачею пакетів даних або шаблони енергоспоживання, як стеганографічних контейнерів.

Важливим фактором є також розробка ефективних методів управління ключами та параметрами захисту. Правильна організація процесів генерації, розповсюдження та оновлення криптографічних ключів може значно вплинути на загальну енергоефективність системи. При цьому необхідно забезпечити надійний захист самих ключів та параметрів, що використовуються для стеганографічного приховування даних.

Перспективним напрямком є також використання методів машинного навчання для оптимізації енергоспоживання систем захисту. Алгоритми штучного інтелекту можуть аналізувати патерни використання пристроїв та автоматично адаптувати параметри захисту для досягнення оптимального балансу між безпекою та енергоефективністю.

У контексті енергоефективності важливо також враховувати можливості відновлення енергії з навколишнього середовища. Використання технологій енергозбору (energy harvesting) може допомогти компенсувати додаткові енергетичні витрати, пов'язані з реалізацією механізмів захисту. Це особливо актуально для IoT-пристроїв, що працюють у віддалених або важкодоступних місцях.

Забезпечення енергоефективності при реалізації комбінованих методів захисту в IoT-системах є комплексним завданням, яке потребує врахування багатьох факторів та застосування інноваційних підходів. Успішне вирішення цієї задачі дозволить створити надійні та довговічні системи IoT, здатні забезпечити належний рівень захисту даних без надмірного споживання енергії

Висновок. У результаті проведеного дослідження було розглянуто актуальні проблеми захисту даних в системах Інтернету речей та запропоновано комплексний підхід до їх вирішення шляхом комбінування криптографічних та стеганографічних методів. Особлива увага була приділена питанню енергоефективності при реалізації цих методів захисту, враховуючи обмежені ресурси IoT-пристроїв.

Було встановлено, що використання адаптивних алгоритмів захисту та спеціалізованих апаратних рішень дозволяє значно оптимізувати енергоспоживання при забезпеченні належного рівня безпеки. Запропоновані підходи до стеганографічного приховування даних з використанням природних особливостей роботи IoT-пристроїв демонструють перспективність у контексті мінімізації додаткових енергетичних витрат.

Результати дослідження показують, що ефективне поєднання криптографічних та стеганографічних методів з урахуванням енергетичних обмежень дозволяє створити надійну систему захисту даних в IoT. Подальший розвиток цього напрямку може бути пов'язаний з впровадженням технологій машинного навчання для оптимізації параметрів захисту та дослідженням можливостей використання альтернативних джерел енергії для IoT-пристроїв.

Перелік використаних джерел.

1. Аль-Судані О.І., Ковальчук Л.В., Кучинська Н.В. Методи стеганографічного захисту даних в IoT-системах. Захист інформації. 2023. Т. 25, № 2. С. 73-82.

2. Бурячок В.Л., Толюпа С.В., Складанний П.М. Інформаційна безпека та кібербезпека IoT-пристроїв: монографія. Київ: ДУТ, 2022. 284 с.

3. Гнатюк С. О., Кінзерявий В. М., Бурячок В. Л. Теоретичні основи побудови систем захисту інформації в IoT. Безпека інформації. 2023. Т. 29, № 1. С. 15-24.

4. Корченко О.Г., Терейковський І.А., Заболотний В.В. Енергоефективні методи криптографічного захисту в системах IoT. Вісник Національного технічного університету України КПІ. Серія: Радіотехніка. Радіоапаратобудування. 2022. № 88. С. 66-75.

5. Куліш С.М., Хорошко В.О., Шелест М.Є. Комбіновані методи захисту інформації в сучасних мережах IoT. Сучасний захист інформації. 2023. № 1(49). С. 28-37.

6. Юдін О.К., Бучик С.С., Чунарьова А.В. Методологія оцінювання енергоефективності систем захисту інформації в IoT. Наукоємні технології. 2022. № 2(54). С. 115-124.

7. Bondar V., Hryshchuk R., Horoshko V. Steganographic Methods for IoT Data Protection: Mathematical Models and Implementation. Cybersecurity: Education, Science, Technology. 2023. Vol. 3, No. 15. P. 88-97.

ГРОМАДСЬКА ОРГАНІЗАЦІЯ
«КІБЕРБЕЗПЕКА І АВТОМАТИЗАЦІЯ»

Матеріали
науково-практичного симпозіуму
«ЗАХИСТ ІНФОРМАЦІЇ»

30 листопада 2024
Тернопіль

У збірнику опубліковано матеріали науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. - 130с.

Технічний редактор: Давлетова А.Я.

Адреса редакції:
Громадська організація «Кібербезпека і автоматизація»
м. Тернопіль
Контактний телефон: (066)043-42-10
e-mail: conferencekb@gmail.com

ЗМІСТ

УДК 004.056(477)

Ігор САПІЖУК, Володимир ДРАПАК

Західноукраїнський національний університет

ВИКОРИСТАННЯ БЛОКЧЕЙНУ ДЛЯ ЗАБЕЗПЕЧЕННЯ АВТЕНТИЧНОСТІ ТА ЦІЛІСНОСТІ ДАНИХ В ІНТЕРНЕТІ РЕЧЕЙ

Вступ. В умовах стрімкого розвитку технологій Інтернету речей (IoT) та експоненціального зростання кількості підключених пристроїв, питання забезпечення автентичності та цілісності даних стає все більш критичним. За прогнозами аналітиків, до 2025 року кількість IoT-пристроїв у світі перевищить 75 мільярдів одиниць, що створює безпрецедентні виклики для безпеки та надійності даних, які вони генерують та обробляють.

Традиційні централізовані системи захисту інформації часто виявляються неефективними для забезпечення належного рівня безпеки в розподілених IoT-мережах. У цьому контексті технологія блокчейн, завдяки своїм фундаментальним властивостям децентралізації, незмінності та прозорості, пропонує інноваційний підхід до вирішення проблем автентифікації та захисту цілісності даних в екосистемі Інтернету речей. Дана робота присвячена дослідженню можливостей та перспектив використання блокчейн-технології для створення надійної та масштабованої інфраструктури безпеки в IoT-системах, а також аналізу практичних аспектів її впровадження.

Мета: комплексний аналіз сучасних викликів безпеки в системах Інтернету речей (IoT) та дослідження можливостей інтеграції блокчейн-технології як інструменту для забезпечення автентичності та цілісності даних в IoT-екосистемі. Робота спрямована на виявлення ключових проблем безпеки в IoT-системах, включаючи питання масштабованості, гетерогенності пристроїв та управління оновленнями, а також на оцінку ефективності застосування блокчейн-технологій для їх вирішення. Особлива увага приділяється аналізу архітектурних рішень та механізмів інтеграції блокчейну в IoT-інфраструктуру з урахуванням специфічних вимог та обмежень IoT-пристроїв, що дозволить розробити рекомендації щодо створення надійних та безпечних децентралізованих IoT систем.

1. Аналіз сучасних викликів безпеки в системах інтернету речей

Сучасний етап розвитку інформаційних технологій характеризується стрімким поширенням систем IoT, які революційно змінюють способи взаємодії між пристроями та обробки даних у різноманітних сферах життя. Масове впровадження IoT-пристроїв у промисловості, медицині, розумних містах та побуті створює безпрецедентні можливості для автоматизації процесів та підвищення якості життя людей. Проте разом з цими перевагами виникають і серйозні виклики у сфері забезпечення безпеки даних та захисту від кіберзагроз. За статистикою, кількість кібератак на IoT-пристрої зростає експоненціально, що робить питання безпеки критично важливим для подальшого розвитку цієї технології [1].

Одним з найбільш серйозних викликів у сфері IoT-безпеки (рисунок 1) є проблема автентифікації пристроїв та захисту даних, які вони генерують. Величезна кількість підключених пристроїв, їх різноманітність та обмежені обчислювальні ресурси створюють унікальні умови, в яких традиційні методи забезпечення безпеки виявляються

неефективними. Багато IoT-пристроїв працюють з використанням застарілих протоколів зв'язку, які не передбачають належних механізмів шифрування та автентифікації. Це робить їх вразливими до різноманітних типів атак, включаючи man-in-the-middle атаки, спуфінг та несанкціонований доступ до даних. Особливо гостро ця проблема проявляється в промислових IoT-системах, де компрометація даних може призвести до серйозних фізичних та економічних наслідків [2].

Іншим важливим аспектом проблеми безпеки в IoT є масштабованість систем захисту. З кожним роком кількість підключених пристроїв зростає в геометричній прогресії, що створює значне навантаження на мережеву інфраструктуру та системи безпеки. Традиційні централізовані системи управління безпекою не здатні ефективно обробляти такі обсяги даних та забезпечувати своєчасну реакцію на загрози. Крім того, гетерогенність IoT-пристроїв, які використовують різні протоколи зв'язку та стандарти безпеки, ускладнює створення єдиної системи захисту. Це призводить до фрагментації систем безпеки та створення потенційних вразливостей на стиках різних технологій [3].

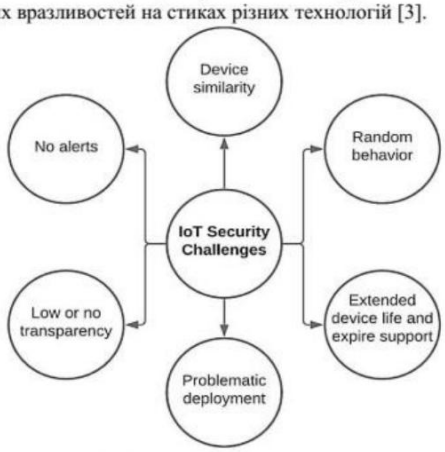

Рисунок 1 - Виклики безпеки IoT

Особливу увагу слід приділити проблемі оновлення програмного забезпечення та усунення вразливостей в IoT-пристроях. На відміну від традиційних комп'ютерних систем, багато IoT-пристроїв не мають механізмів автоматичного оновлення або взагалі не передбачають можливості оновлення прошивки. Це створює ситуацію, коли навіть відомі вразливості залишаються невиправленими протягом тривалого часу, створюючи потенційні точки входу для зловмисників. Більше того, часто виробники IoT-пристроїв припиняють підтримку своїх продуктів через кілька років після випуску, залишаючи користувачів з потенційно вразливими пристроями [4].

Проблема конфіденційності даних також є одним з ключових викликів у сфері IoT. IoT-пристрої збирають величезні обсяги персональних та конфіденційних даних, включаючи інформацію про місцезнаходження користувачів, їх звички, стан здоров'я та

багато іншого. Забезпечення належного захисту цих даних від несанкціонованого доступу та витоків є критично важливим завданням. При цьому необхідно враховувати не тільки технічні аспекти захисту, але й відповідність різним регуляторним вимогам, таким як GDPR в Європейському Союзі [5].

Окремою проблемою є забезпечення фізичної безпеки IoT-пристроїв (рисунок 2). Багато таких пристроїв розміщуються в публічних місцях або важкодоступних локаціях, що робить їх вразливими до фізичного втручання. Зловмисники можуть отримати прямий доступ до апаратної частини пристрою, витягти ключі шифрування або модифікувати його поведінку. Традиційні методи фізичного захисту не завжди можуть бути застосовані через обмеження у розмірах пристроїв або вимоги до їх мобільності [6].

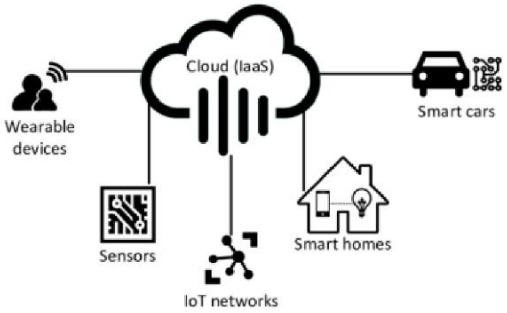

Рисунок 2 - Класична схема IoT-системи

Енергоефективність також відіграє важливу роль у контексті безпеки IoT-систем. Багато IoT-пристроїв працюють від батарей і мають обмежений запас енергії. Впровадження складних механізмів шифрування та автентифікації може значно збільшити енергоспоживання, що негативно впливає на час автономної роботи пристроїв. Необхідно знаходити баланс між рівнем захисту та енергоефективністю, що часто призводить до компромісів у сфері безпеки [7].

Проблема стандартизації також залишається актуальною для сфери IoT-безпеки. Відсутність єдиних стандартів та протоколів безпеки призводить до фрагментації ринку та створення несумісних рішень. Це ускладнює взаємодію між різними системами та створює додаткові ризики безпеки. Хоча існують різні ініціативи зі стандартизації, процес їх впровадження відбувається повільно, і багато виробників продовжують використовувати власні пропрієтарні рішення.

Іншим важливим аспектом є проблема виявлення та реагування на інциденти безпеки в IoT-системах. Традиційні системи виявлення вторгнень часто не здатні ефективно працювати в умовах IoT через специфіку трафіку та поведінки пристроїв. Крім того, велика кількість пристроїв та їх розподілений характер ускладнюють моніторинг та аналіз подій безпеки.

Необхідно розробляти нові підходи та інструменти, які враховують особливості IoT-систем та дозволяють ефективно виявляти та реагувати на загрози [8].

У контексті промислового IoT (IIoT) особливу увагу слід приділити проблемі забезпечення безперервності роботи критично важливих систем (рисунок 3). Атаки на промислові IoT-системи можуть призвести до зупинки виробництва, пошкодження обладнання або навіть загрози життю людей. Тому необхідно розробляти механізми резервування та відновлення, які дозволяють мінімізувати наслідки можливих атак та забезпечити стійкість системи до різних типів загроз [9].

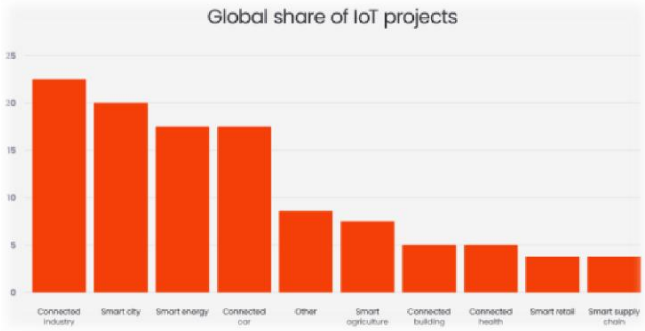

Рисунок 3 - Глобальний графік IoT- проєктів

Вирішення цих викликів вимагає комплексного підходу, який включає як технічні, так і організаційні заходи. Необхідно розвивати нові технології захисту, які враховують специфіку IoT-систем, впроваджувати механізми безпечного оновлення програмного забезпечення, розробляти стандарти безпеки та проводити регулярний аудит систем. Важливу роль також відіграє навчання користувачів та підвищення загального рівня обізнаності про загрози безпеки в сфері IoT. Тільки комплексний підхід до вирішення цих проблем дозволить створити надійну та безпечну екосистему Інтернету речей, яка відповідає сучасним вимогам та викликам [10].

2. Інтеграція блокчейн-технології в екосистему IoT

В умовах зростаючих викликів безпеки в сфері IoT, технологія блокчейн представляє собою інноваційне рішення, здатне забезпечити необхідний рівень довіри, безпеки та прозорості в розподілених IoT-системах. Блокчейн, який спочатку був розроблений як основа для криптовалют, демонструє значний потенціал для вирішення фундаментальних проблем безпеки в екосистемі IoT. Ключові властивості блокчейну, такі як децентралізація, незмінність записів та криптографічний захист, створюють надійну основу для побудови захищених IoT-систем нового покоління. Інтеграція блокчейну в IoT-інфраструктуру відкриває нові можливості для забезпечення автентичності та цілісності даних, а також створення довірених середовищ для взаємодії між пристроями [11].

Основним принципом інтеграції блокчейну в IoT-системи є створення розподіленого реєстру, в якому зберігаються всі транзакції та взаємодії між пристроями. Кожен IoT-пристрій може бути зареєстрований в блокчейні з унікальним ідентифікатором, що дозволяє

відслідковувати його активність та верифікувати джерело даних. Всі дані, які генеруються пристроями, записуються в блокчейн у вигляді транзакцій, що забезпечує їх незмінність та можливість аудиту. Важливою особливістю такого підходу є відсутність централізованого вузла управління, що значно підвищує стійкість системи до атак та відмов. Замість цього, консенсус щодо валідності даних досягається через розподілений механізм узгодження між вузлами мережі [12].

При впровадженні блокчейну в IoT-системи особливу увагу слід приділяти вибору відповідного механізму консенсусу (рисунок 4). Традиційні механізми, такі як Proof of Work (PoW), які використовуються в Bitcoin та інших криптовалютах, можуть бути занадто ресурсоємними для IoT-пристроїв з обмеженими обчислювальними можливостями. Тому розробляються спеціалізовані легковагові механізми консенсусу, оптимізовані для використання в IoT-середовищі. Такі механізми повинні забезпечувати необхідний рівень безпеки та надійності, одночасно враховуючи обмеження IoT-пристроїв щодо енергоспоживання та обчислювальних ресурсів. Популярними альтернативами є Proof of Stake (PoS) та Delegated Proof of Stake (DPoS), які вимагають значно менше ресурсів для досягнення консенсусу [13].

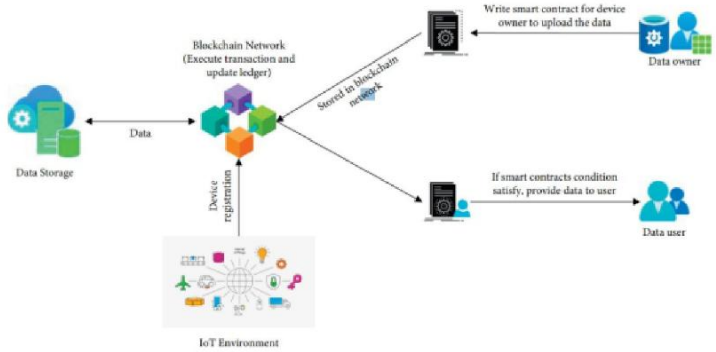

Рисунок 4 - Інтеграція блокчейну, смарт-контрактів і систем на основі IoT

Архітектура блокчейн-базованої IoT-системи повинна враховувати специфіку різних типів пристроїв та їх можливостей. Для оптимізації роботи системи часто використовується багаторівнева архітектура, де потужніші пристрої (шлюзи, сервери) виступають як повні вузли блокчейну, а пристрої з обмеженими ресурсами працюють як легкі клієнти. Така архітектура дозволяє ефективно розподіляти навантаження та забезпечувати масштабованість системи. Крім того, використання смарт-контрактів у блокчейні дозволяє автоматизувати багато процесів, включаючи управління доступом, оновлення програмного забезпечення та обробку даних від пристроїв[14].

Важливим аспектом інтеграції блокчейну в IoT є забезпечення конфіденційності даних. Хоча блокчейн за своєю природою є публічним реєстром, в багатьох випадках необхідно забезпечити приватність певної інформації. Для цього використовуються різні

криптографічні методи, такі як гомоморфне шифрування та zero-knowledge proofs, які дозволяють верифікувати транзакції без розкриття їх змісту. Також можливе створення приватних або консорціумних блокчейнів, доступ до яких мають тільки авторизовані учасники. Це особливо актуально для промислових IoT-систем, де конфіденційність даних має критичне значення.

Впровадження блокчейну в IoT-системи також створює нові можливості для монетизації даних та створення децентралізованих ринків IoT-послуг (рисунок 5). Пристрої можуть автоматично торгувати даними або послугами через смарт-контракти, що відкриває нові бізнес-моделі в сфері IoT. Наприклад, сенсори можуть продавати дані про навколишнє середовище, а виконавчі пристрої можуть надавати послуги на основі отриманих даних. Вся взаємодія при цьому відбувається автоматично через блокчейн, що забезпечує прозорість та довіру між учасниками.

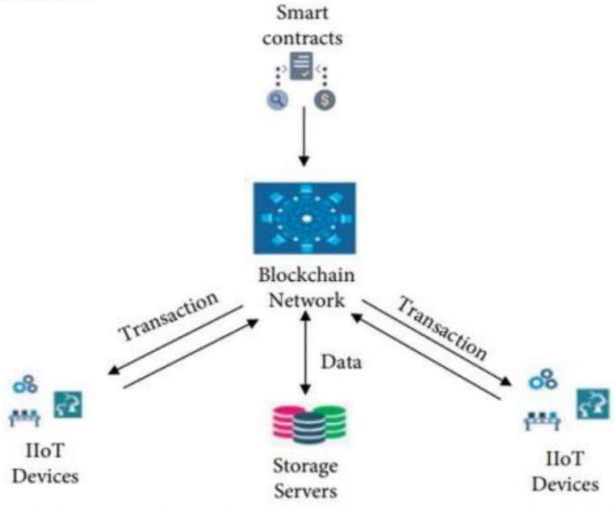

Рисунок 5 - Інтеграція блокчейну, смарт-контракту та системи на основі IIoT.

Особливу увагу при інтеграції блокчейну в IoT слід приділяти питанням масштабованості та продуктивності. З огляду на величезну кількість IoT-пристроїв та генерованих ними даних, традиційні блокчейн-платформи можуть виявитися недостатньо ефективними. Для вирішення цієї проблеми розробляються різні оптимізації, такі як шардинг (розподіл даних між підмережами), використання бічних ланцюгів (sidechains) та впровадження більш ефективних протоколів консенсусу. Також важливо оптимізувати розмір блоків та частоту їх створення з урахуванням специфіки IoT-трафіку[15].

У контексті безпеки важливо відзначити, що блокчейн не є панацеєю і не вирішує всіх проблем безпеки IoT. Наприклад, блокчейн не може захистити від фізичного втручання в пристрої або від атак на сенсори, які генерують некоректні дані. Тому інтеграція блокчейну

повинна бути частиною комплексного підходу до безпеки, який включає також традиційні методи захисту, такі як шифрування каналів зв'язку, фізичний захист пристроїв та моніторинг аномалій.

Практична реалізація блокчейн-інтеграції в IoT-системах вимагає вирішення ряду технічних викликів. Необхідно розробляти спеціалізовані протоколи та програмне забезпечення, оптимізовані для роботи на IoT-пристроях, створювати ефективні механізми зберігання та обробки даних, забезпечувати сумісність з існуючими IoT-стандартами та протоколами. Важливо також враховувати питання енергоефективності, оскільки багато IoT-пристроїв працюють від батарей і мають обмежений запас енергії.

Розвиток блокчейн-технологій для IoT продовжується, і з'являються нові рішення та підходи. Перспективними напрямками є розробка спеціалізованих блокчейн-платформ для IoT, впровадження квантово-стійких криптографічних алгоритмів, створення більш ефективних механізмів консенсусу та покращення масштабованості систем. Важливу роль також відіграє стандартизація та створення єдиних протоколів взаємодії між різними блокчейн-платформами та IoT-системами.

Успішна інтеграція блокчейну в екосистему IoT вимагає тісної співпраці між розробниками блокчейн-технологій, виробниками IoT-пристроїв та експертами з безпеки. Необхідно продовжувати дослідження та розробку нових рішень, які дозволять повною мірою реалізувати потенціал блокчейну для забезпечення безпеки та довіри в IoT-системах. Тільки такий комплексний підхід дозволить створити по-справжньому надійну та безпечну інфраструктуру для Інтернету речей майбутнього.

Висновок. У результаті проведеного аналізу використання блокчейну для забезпечення автентичності та цілісності даних в IoT можна зробити ряд важливих висновків. Стрімкий розвиток екосистеми IoT створює безпрецедентні виклики у сфері безпеки даних, які неможливо ефективно вирішити за допомогою традиційних централізованих підходів. Виконаний огляд показав, що технологія блокчейн, завдяки своїм фундаментальним властивостям децентралізації, незмінності та криптографічного захисту, представляє собою перспективне рішення для забезпечення надійної автентифікації та захисту цілісності даних в IoT-системах.

Аналіз сучасних викликів безпеки в системах IoT виявив критичні проблеми, пов'язані з масштабованістю, гетерогенністю пристроїв, обмеженістю ресурсів та складністю управління оновленнями. Впровадження блокчейн-технології дозволяє ефективно адресувати ці виклики шляхом створення розподіленої довіреної інфраструктури для верифікації та зберігання даних. При цьому важливо відзначити, що успішна інтеграція блокчейну в IoT-системи вимагає ретельного вибору архітектурних рішень, механізмів консенсусу та протоколів взаємодії, які враховують специфіку та обмеження IoT-пристроїв.

Підсумовуючи результати аналізу, можна стверджувати, що незважаючи на значний потенціал, блокчейн не є універсальним рішенням всіх проблем безпеки в IoT. Необхідний комплексний підхід, який поєднує переваги блокчейну з традиційними механізмами захисту та враховує специфічні вимоги різних сфер застосування IoT. Подальший розвиток цього напрямку вимагає поглибленого вивчення питань оптимізації продуктивності, масштабованості та енергоефективності блокчейн-рішень для IoT, а також стандартизації

протоколів та механізмів взаємодії. Тільки такий комплексний підхід дозволить створити надійну та безпечну інфраструктуру для розвитку Інтернету речей у майбутньому.

Перелік використаних джерел.

1. Алексєєв В. І., Петренко С. О. Технологія блокчейн в системах Інтернету речей: архітектурні рішення та проблеми безпеки. Кібербезпека: освіта, наука, техніка. 2023. Том 9, № 1. С. 43-58. DOI: 10.28925/2663-4023.2023.9.4358

2. Bondar I., Ponomarenko V. Internet of Things Security: Blockchain Integration Approaches. Information Technology and Security. 2022. Vol. 10, Issue 2. P. 107-120. https://doi.org/10.20535/2411-1031.2022.10.2.248741

3. Коваленко І. М., Мельник А. О. Застосування блокчейн-технологій для підвищення безпеки IoT-пристроїв. Вісник Національного університету Львівська політехніка. Серія: Інформаційні системи та мережі. 2023. № 13. С. 55-67.

4. Dorri A., Kanhere S. S., Jurdak R. Blockchain in Internet of Things: Challenges and Solutions. IEEE Communications Surveys & Tutorials. 2021. Vol. 23, No. 3. P. 1722-1754.

5. Петров О. С., Іванова Л. М. Методи забезпечення цілісності даних в IoT-системах на основі технології блокчейн. Захист інформації. 2023. Том 25, № 2. С. 94-102.

6. Yang Y., Wu L., Yin G. Applications of Blockchain in IoT Security: A Comprehensive Survey. ACM Computing Surveys. 2022. Vol. 54, No. 1. P. 1-35.

7. Ковальчук Д. В. Інтеграція блокчейн-технологій в екосистему Інтернету речей: виклики та перспективи. Телекомунікаційні та інформаційні технології. 2023. № 2(75). С. 15-28.

8. Zhang R., Xue R., Liu L. Security and Privacy on Blockchain-Based IoT Systems: State-of-the-Art and Future Directions. IEEE Internet of Things Journal. 2021. Vol. 8, Issue 3. P. 1500-1525.

9. Сидоренко М. В., Григорчук Т. І. Аналіз механізмів консенсусу для блокчейн-систем в контексті IoT. Наукові записки НаУКМА. Комп'ютерні науки. 2023. Том 6. С. 82-91.

10. Chen J., Wang K., Li X. Distributed Ledger Technology for IoT: Architectures, Applications, and Future Directions. IEEE Network. 2022. Vol. 36, No. 1. P. 52-58.

11. Марченко О. П., Василенко В. С. Безпека даних в IoT: інтеграція блокчейн-технологій та смарт-контрактів. Системи обробки інформації. 2023. № 2(169). С. 123-134.

12. Kumar T., Ramani V., Ahmad I. Blockchain Technology for Securing Internet of Things: Background, Integration Patterns, and Security Requirements. Journal of Systems Architecture. 2021. Vol. 118. P. 102048.

13. Федоренко С. М. Проблеми масштабованості блокчейн-рішень в контексті IoT-систем. Кібербезпека та інформаційні технології. 2023. Вип. 3. С. 28-39.

14. Wang X., Li K., Li H. A Survey on Blockchain-Based IoT Systems: Security Challenges and Solutions. IEEE Communications Surveys & Tutorials. 2022. Vol. 24, No. 2. P. 1237-1264.

15. Андрійчук О. В., Ковальчук Л. В. Сучасні підходи до забезпечення автентичності даних в IoT на основі блокчейн-технологій. Електронне моделювання. 2023. Том 45, № 3. С. 73-88.