

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
**Програмне забезпечення системи кібербезпеки для ресурсно-
обмежених IoT-пристроїв**

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-1
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Лук’яненко Р.О.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Рівень вищої освіти бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 125 "Кібербезпека"
Освітньо-професійна (освітньо-наукова) програма "Кібербезпека"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
27 лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Лук'яненка Романа Олександровича

(прізвище, ім'я, по батькові)

- Тема роботи Програмне забезпечення системи кібербезпеки для
ресурсно обмежених IoT-пристроїв
- Керівник роботи Дреєв Олександр Миколайович, докт. техн. наук, доцент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу № 134 -02 від 27.02.2026 року
- Строк подання студентом роботи до захисту 10.06.2026
- Мета та завдання випускної кваліфікаційної роботи: Метою розробки є
дослідження та програмна реалізація системи кіберзахисту для IoT-пристроїв з
обмеженими ресурсами
- Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити)
 - Призначення та область використання.
 - Перегляд аналогічних існуючих систем.
 - Опис і обґрунтування проектних рішень.
 - Етапи програмування системи.
 - Впровадження системи в промислову експлуатацію
- Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

<u>Структурна схема системи</u>	<u>1 аркуш</u>
<u>Функціональна схема системи</u>	<u>1 аркуш</u>
<u>Діаграма процесів</u>	<u>1 аркуш</u>
<u>Блок-схеми алгоритму роботи додатку</u>	<u>2 аркуша</u>
- Дата видачі завдання 27.02.26

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	14.05.2026 р.	
8.	Попередній захист роботи	16.05.2026 р.	

Дата видачі завдання 27.02.26

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання

Підпис здобувача

(прізвище та ініціали)

АНОТАЦІЯ

**Лук'яненко Р.О. Програмне забезпечення системи кібербезпеки для
ресурсно-обмежених IoT-пристроїв. 125 Кібербезпека.
Центральноукраїнський національний технічний університет.
Кропивницький. 2026.**

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначене для забезпечення захищеного обміну даними між IoT-пристроями в умовах обмежених апаратних ресурсів. Реалізована система забезпечує безпечну передачу інформації, використовуючи сучасні криптографічні алгоритми та легковагові мережеві протоколи, що дозволяє ефективно використовувати її у сфері Інтернету речей.

Метою розробки є дослідження та програмна реалізація системи кіберзахисту для IoT-пристроїв з обмеженими ресурсами.

Проведено аналіз сучасних загроз кібербезпеці в системах Інтернету речей. Обґрунтовано вибір криптографічних алгоритмів AES та ECC для використання в умовах обмежених ресурсів. Розроблено програмну модель захищеного обміну даними між IoT-пристроями із застосуванням протоколу MQTT. Реалізовано програмне забезпечення системи кіберзахисту з використанням механізмів шифрування та захищеного обміну ключами.

Ключові слова: комп'ютерна інженерія, Інтернет речей, MQTT, кібербезпека.

ABSTRACT

Lukianenko R.O. Cybersecurity Software System for Resource-Constrained IoT Devices. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

The software developed as a result of completing the final qualification thesis for the first (bachelor's) level of higher education is intended to provide secure data exchange between IoT devices operating under conditions of limited hardware resources. The implemented system ensures secure information transmission through the use of modern cryptographic algorithms and lightweight network protocols, enabling its efficient application in the field of the Internet of Things.

An analysis of modern cybersecurity threats in Internet of Things systems was conducted. The choice of AES and ECC cryptographic algorithms for use in resource-constrained environments was substantiated. A software model of secure data exchange between IoT devices using the MQTT protocol was developed. A cybersecurity software system utilizing encryption mechanisms and secure key exchange was implemented.

Keywords: computer engineering, IoT, MQTT, cybersecurity.

ЗМІСТ

ВСТУП.....	2
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	4
1.1 Призначення системи	4
1.2 Область застосування.....	5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим	7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	11
2.3 Розгорнута постановка завдання	13
3 ОПИС І ОБґРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	15
3.1 Опис функціонування системи	15
3.2 Розробка структурної схеми.....	23
3.3 Розробка функціональної схеми	26
3.4 Розробка діаграми процесів.....	29
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ. 32	
4.1 Розробка блок–схем та опис алгоритмів функціонування системи.....	32
4.2 Захист розробленого програмного забезпечення.....	34
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	42
6 ОСНОВНІ ВИСНОВКИ.....	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	50

					ВКРБ-125.26.00.11.00.00.ПЗ				
Вим	Арк.	№ докум.	Підп.	Дата	Програмне забезпечення системи кібербезпеки для ресурсно-обмежених IoT-пристроїв	Літ.	Аркуш	Аркушів	
Розроб.	Лук'яненко Р.О.					Б		2	54
Перев.	Дресев О.С..								
Н.контр.	Коваленко А.С.								
Затв.	Смірнов О.А.					КБ-22-1			

ВСТУП

Актуальність теми. Особливої актуальності в описаних нижче умовах набуває потреба в захисті мережного трафіку. Як відомо, передача саме мережного трафіку незахищеними або частково захищеними каналами є однією з найбільш вразливих ланок інформаційно–телекомунікаційних систем. Також, як широко відомо і доведено статистично, значна частина витоків інформації відбувається саме завдяки перехопленню зловмисником чи третьою особою мережного трафіку. Це додає особливої важливості процесу вибору технології що буде використана. Темпи розвитку інформаційних технологій ростуть з кожним роком, охоплюючи все більше сфер людської діяльності, починаючи повсякденним використанням мобільних пристроїв і закінчуючи складними науковими чи промисловими системами. Такі темпи технічного розвитку неодмінно означають підвищення обсягів даних, необхідних для оброблення а також збільшення вимог до їх захисту. Прогрес, відповідно, вимагає постійного покращення методів захисту даних в цих системах, що стосується не тільки потужності таких методів а й їх надійності та ефективності.

Мета й завдання дослідження. Метою дослідження є порівняння сучасних алгоритмів шифрування за критеріями надійності, швидкодії та ресурсомісткості. Часто використання великих та громіздких криптографічних алгоритмів виявляється занадто ресурсоємним для певних систем, в той час як інші, занадто повільними. Нижче приведена таблиця, в якій визначено десять популярних алгоритмів шифрування мережного трафіку і оцінка деяких їх характеристик, таких як надійність, обчислювальна складність а також вимоги до пам'яті.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

- Проведення аналізу сучасних загроз кібербезпеці в системах Інтернет–речей та визначення основних вразливостей в процесі передачі даних.
- Вибір одного чи декількох вже існуючих криптографічних алгоритмів, а також їх оцінка, на основі чого, буде вибрано декілька найвідповідніших умовам пізніше.
- Виконання порівняльного аналізу симетричних та асиметричних методів шифрування з урахуванням обмежень IoT–пристроїв.
- Обґрунтування вибору оптимальних криптографічних алгоритмів для використання в системах з обмеженими ресурсами.
- Розробка моделі системи кіберзахисту для IoT–пристроїв з урахуванням особливостей їх функціонування.
- Реалізація механізму захищеного обміну даними з використанням вибраних криптографічних методів.
- Дослідження можливості використання легковагових мережевих протоколів для передачі захищених даних.
- Оцінка ефективності запропонованого рішення за показниками безпеки та продуктивності.

Практична цінність отриманих результатів полягає у розробці програмного забезпечення системи кіберзахисту для ресурсно–обмежених IoT–пристроїв, яке забезпечує захищений обмін даними між компонентами системи з використанням сучасних криптографічних алгоритмів. Реалізовані механізми шифрування, автентифікації та формування спільного секретного ключа дозволяють підвищити рівень конфіденційності та цілісності інформації під час її передачі через мережеве середовище.

Таким чином, виходячи з вищеперерахованого, розробка програмного забезпечення системи кібербезпеки для ресурсно–обмежених IoT–пристроїв є актуальною задачею, що потребує вирішення у межах даної випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням системи є забезпечення захищеного обміну даними між IoT-пристроями та серверною частиною шляхом використання сучасних криптографічних алгоритмів симетричного та асиметричного шифрування. Система призначена для забезпечення конфіденційності, цілісності та автентичності інформації, що передається через відкриті або потенційно незахищені канали зв'язку, а також для запобігання несанкціонованому доступу, підміні чи модифікації даних у процесі передачі. Додатково система орієнтована на використання у середовищах з обмеженими апаратними ресурсами, що характерно для більшості сучасних IoT-пристроїв, тому реалізовані механізми захисту враховують необхідність зменшення обчислювального навантаження та оптимізації використання пам'яті й мережевих ресурсів.

Існує безліч різних способів захистити важливі дані від їх отримання небажаними персонами. Криптографічний алгоритм один з них. Це спосіб захисту інформації, що повинна передатися від певного адресата до певного отримувача. Такий спосіб, дозволяє виключити гіпотетичну можливість отримання несанкціонованого доступу, підробки чи зміни зловмисником даних, які передаються. Криптографічний алгоритм методологічно завжди є математично обґрунтованим і широко використовується в цілях, які відповідають потребам забезпечення цілісності, конфіденційності та доступності, а у деяких, окремих, випадках, навіть невідомості інформаційної системи. Математична обґрунтованість криптографічного алгоритму передбачає виконання ним певної, чітко визначеної послідовності обчислювальних дій, що ускладнюють, чи іноді, як у випадку хеш-функції, унеможливлюють відновлення першопочаткових даних які були зашифровані. У більшості

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

випадків відновлення зашифрованої інформації, тобто дешифрування, передбачає наявність криптографічного ключа чи криптографічних ключів.

Сучасні криптографічні алгоритми поділяються на дві категорії, симетричні та асиметричні. Доцільно детальніше пояснити основну відмінність симетричних алгоритмів від асиметричних. Першочергово, симетричний алгоритм для шифрування використовує той самий ключ, який був використаний для шифрування даних попередньо. Головною перевагою, за рахунок якої в сучасних системах безпеки досі не відмовилися від використання симетричних алгоритмів є їхня швидкодія. Іноді такі алгоритми можуть бути використані навіть у відриві від використання асиметричних алгоритмів. Як приклад такого алгоритму є AES, чому саме він буде вказано нижче, що широко застосовується завдяки високій швидкодії та ефективності.

Асиметричні алгоритми, у свою чергу, використовують пари ключів, публічний та, відповідно, приватний. Публічний ключ застосовується, безпосередньо, для шифрування даних, тоді як приватний – для їх дешифрування. Прикладів асиметричного алгоритму можна навести ЕСС, вибір обґрунтовний нижче, що базується на складності задачі дискретного логарифмування на еліптичних кривих. Перевагою асиметричних алгоритмів є можливість безпечного обміну ключами через незахищені канали зв'язку.

1.2 Область застосування

Областю застосування розробленої системи є середовище Інтернет-речей, зокрема системи моніторингу, автоматизації та дистанційного керування, у яких використовується передача даних між ресурсно-обмеженими IoT-пристроями та серверною частиною через мережеве середовище. До таких систем можна віднести системи «розумного будинку», промислові IoT-мережі, системи екологічного моніторингу, медичні IoT-пристрої, телеметричні системи та інші інформаційно-комунікаційні середовища, де існує потреба у забезпеченні

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

конфіденційності, цілісності та автентичності даних. Використання криптографічних алгоритмів AES та ECC дозволяє адаптувати систему до умов обмежених апаратних ресурсів пристроїв при збереженні достатнього рівня кіберзахисту під час передачі інформації через відкриті канали зв'язку.

Необхідно навести декілька найбільш поширених, та найбільш вдалих для порівняння в даному контексті, вже існуючих криптографічних рішень, що застосовуються при роботі з IoT. Також, наведені таким чином алгоритми шифрування будуть порівняні, на основі чого буде вибрано одне або декілька рішень що стануть основою розробки проекту.

AES-256 це симетричний алгоритм шифрування, відноситься до стандарту AES і використовує ключ довжиною 256 біт. Він характеризується дуже високим рівнем надійності завдяки великому простору ключів та стійкості до сучасних криптоаналітичних атак.

AES-128 є менш ресурсоємною версією алгоритму AES з ключем довжиною 128 біт. Його рівень надійності оцінюється як середній у порівнянні з алгоритмами, що оперують більш довгими ключами.

Twofish це симетричний блочний алгоритм шифрування, який забезпечує досить високий рівень безпеки та виступає альтернативою AES. Його обчислювальна складність середня, що дозволяє досягати прийнятної продуктивності при високому рівні захисту.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим

MQTT–брокер Mosquitto

Найкраще підходить для: реалізації легковагового обміну повідомленнями між IoT–пристроями та серверною частиною системи в умовах обмежених мережевих та апаратних ресурсів.

Mosquitto – це відкритий MQTT–брокер, який забезпечує передачу повідомлень між компонентами системи відповідно до моделі publish/subscribe. Даний програмний засіб широко використовується у середовищі Інтернету речей завдяки невеликому споживанню ресурсів, високій стабільності роботи та підтримці сучасних механізмів мережевої взаємодії.

Основні характеристики:

- Підтримка MQTT: Mosquitto реалізує повноцінну підтримку MQTT–протоколу, що дозволяє IoT–пристроєм обмінюватися повідомленнями через систему тем та підписок. Це забезпечує асинхронний характер взаємодії між пристроями та сервером.

- Низьке споживання ресурсів: брокер оптимізований для роботи навіть на малопотужних системах та підтримує одночасне обслуговування великої кількості клієнтів при мінімальному навантаженні на процесор та пам'ять.

- Масштабованість системи: використання брокера дозволяє розділити відправника та отримувача повідомлень, що значно спрощує процес масштабування IoT–мережі та додавання нових пристроїв без зміни архітектури системи.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

– Підтримка захищених з'єднань: Mosquitto підтримує використання TLS/SSL, механізми автентифікації клієнтів та контроль доступу до тем, що дозволяє додатково підвищити рівень кібербезпеки системи.

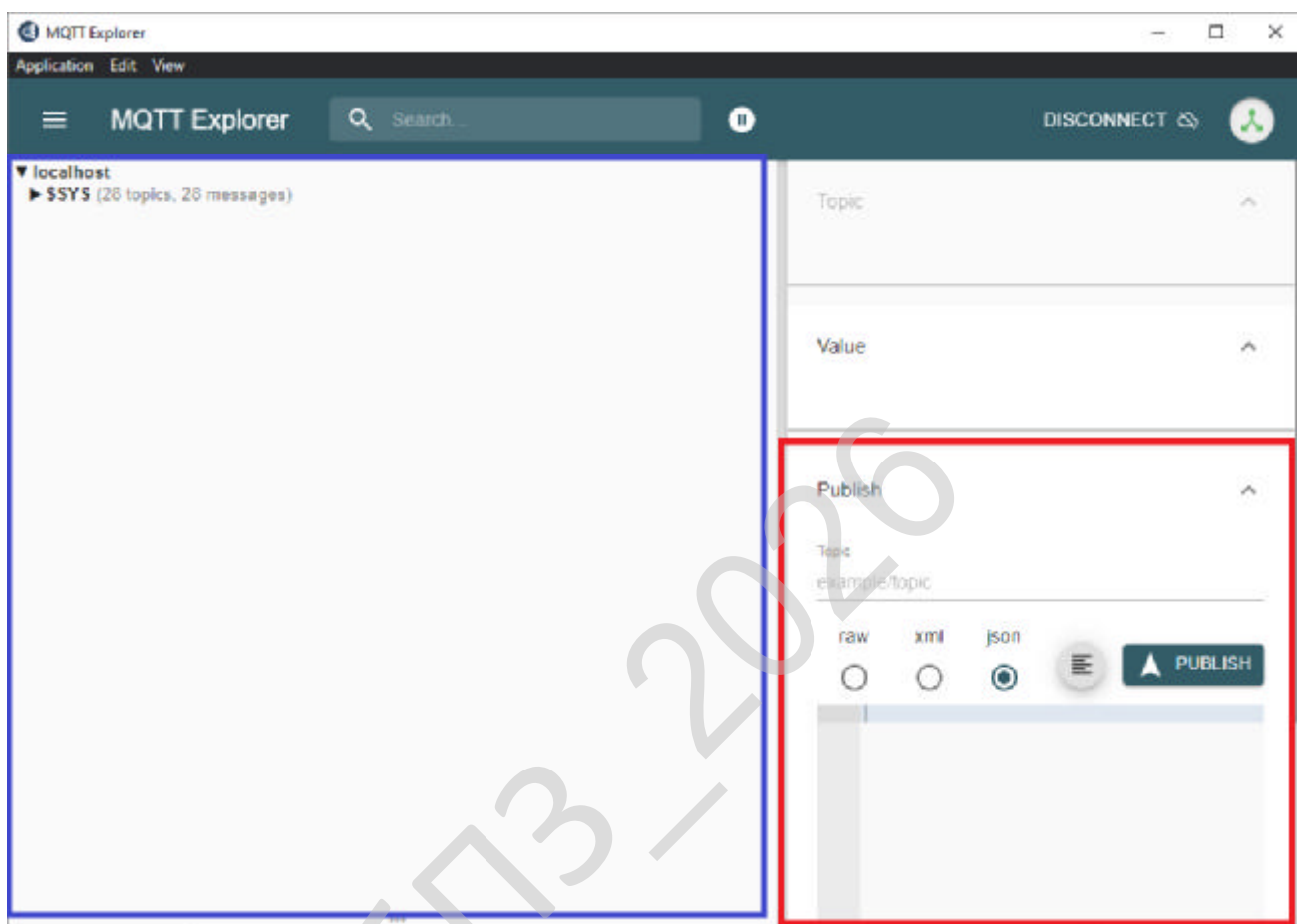


Рисунок 2.1 – Інтерфейс користувача MQTT-брокера Mosquitto

Аналізатор мережевого трафіку Wireshark

Найкраще підходить для: моніторингу мережевої активності, аналізу MQTT-трафіку та перевірки роботи механізмів захищеної передачі даних у IoT-системах.

Wireshark – це програмний засіб для аналізу мережевого трафіку, який дозволяє здійснювати перехоплення, перегляд та дослідження пакетів даних у режимі реального часу. У межах розробленої системи Wireshark використовується для перевірки передачі MQTT-повідомлень, аналізу структури

мережевих пакетів та контролю коректності роботи криптографічного захисту під час обміну даними між IoT–пристроєм та сервером.

Основні характеристики:

- Моніторинг мережевого трафіку: Wireshark дозволяє здійснювати аналіз усіх пакетів, що проходять через мережевий інтерфейс, включаючи MQTT, TCP/IP та TLS–з'єднання.
- Аналіз MQTT–протоколу: програмний засіб підтримує декодування MQTT–повідомлень, що дозволяє досліджувати структуру тем, payload та особливості взаємодії між клієнтами і брокером повідомлень.
- Контроль захищеності передачі даних: використання Wireshark дає можливість перевірити, що повідомлення передаються у зашифрованому вигляді, а також оцінити ефективність реалізованих механізмів криптографічного захисту.

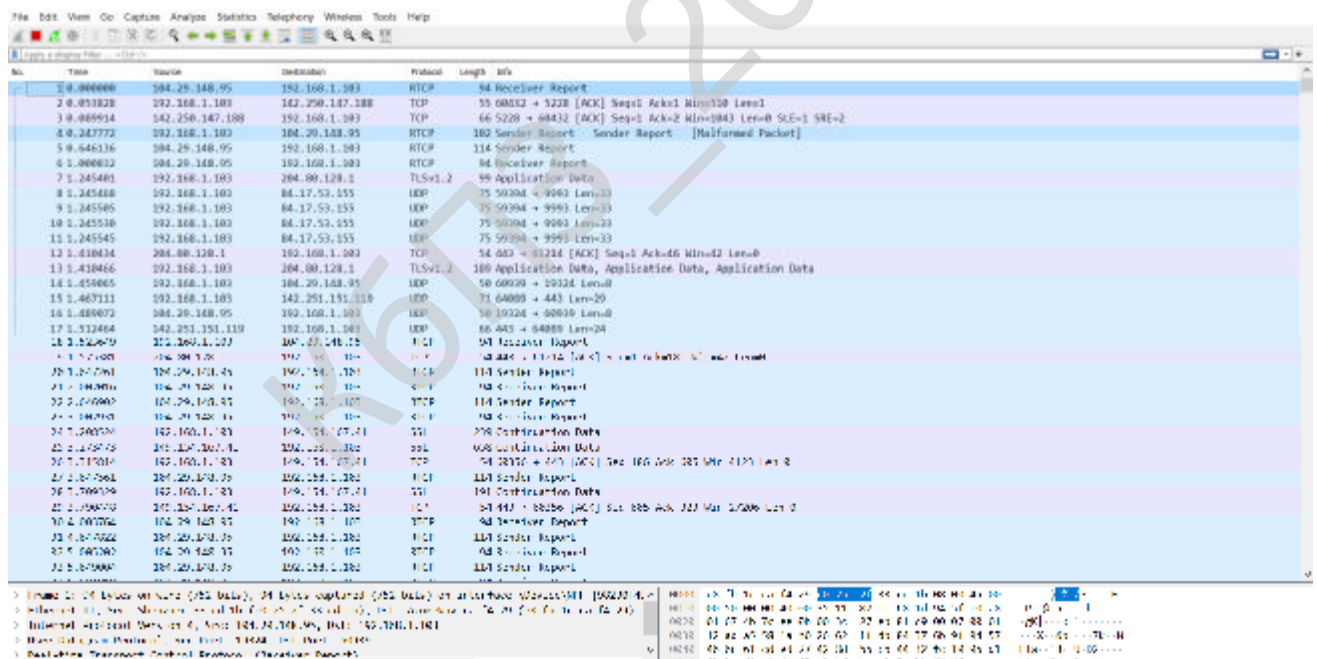


Рисунок 2.2 – Інтерфейс користувача аналізатора Wireshark

Криптографічна бібліотека OpenSSL

Найкраще підходить для: реалізації сучасних механізмів криптографічного захисту в системах безпечного обміну даними між IoT-пристроями та сервером.

OpenSSL – це програмна бібліотека з відкритим вихідним кодом, яка містить реалізацію великої кількості криптографічних алгоритмів та протоколів захищеного зв'язку. Даний програмний засіб використовується у розробленій системі для реалізації ECC, AES-256 та SHA-256, а також для забезпечення механізму формування спільного секретного ключа між компонентами системи.

Основні характеристики:

- Підтримка сучасних криптографічних алгоритмів: бібліотека містить реалізації алгоритмів ECC, AES, RSA, SHA та багатьох інших криптографічних механізмів, що дозволяє створювати захищені системи передачі даних.

- Реалізація механізму ECDH: OpenSSL забезпечує підтримку алгоритму обміну ключами ECDH, який використовується у системі для формування shared key без передачі самого секретного ключа мережею.

- Оптимізація продуктивності: бібліотека має високу швидкодію та підтримує апаратне прискорення криптографічних операцій, що особливо важливо для ресурсно-обмежених IoT-пристроїв.

- Широке практичне використання: OpenSSL є одним із найбільш розповсюджених криптографічних програмних засобів у світі та використовується у багатьох сучасних системах захищеного зв'язку, зокрема HTTPS, TLS, VPN та IoT-рішеннях.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана C++. C++ – високорівнева мова програмування загального призначення з акцентом на максимальну

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

продуктивність, ефективність використання системних ресурсів та низькорівневий контроль. Синтаксис ядра C++ є строгим та структурованим, що забезпечує високу гнучкість розробки. У той же час стандартна бібліотека STL включає великий обсяг корисних функцій, контейнерів та алгоритмів для роботи з пам'яттю та структурами даних.

C++ підтримує кілька парадигм програмування, у тому числі процедурне, об'єктно-орієнтоване, узагальнене, метапрограмування за допомогою шаблонів, та функціональне. Основні архітектурні риси – статична типізація, ручне або автоматизоване, завдяки розумним покажчикам та концепції RAII, керування пам'яттю, висока швидкість виконання скомпільованого коду, механізм обробки виключень та нативна підтримка багатопотокових обчислень. Код у C++ організовується у функції, структури й класи, які компілюються безпосередньо в машинний код цільової платформи.

Еталонними компіляторами для мови є GCC, Clang та MSVC, що підтримують більшість активно використовуваних операційних систем та апаратних архітектур. C++ стандартизовано Міжнародною організацією зі стандартизації ISO, а нові офіційні стандарти C++11, C++14, C++17, C++20, C++23, що виходять кожні три роки, приносячи нові можливості для оптимізації та підвищення безпеки коду.

C++ портований і працює майже на всіх відомих обчислювальних платформах – від мікроконтролерів і вбудованих систем на базі ARM, Xtensa, AVR) до суперкомп'ютерів. Існують готові інструменти компіляції під Microsoft Windows, Linux, macOS, а також спеціалізовані фреймворки, наприклад, ESP-IDF, для розробки під ресурсно-обмежені мікроконтролери. При цьому C++ має повну сумісність із мовою C, що дозволяє безпосередньо взаємодіяти з апаратним забезпеченням та використовувати низькорівневі системні бібліотеки без додаткових абстракцій. Для інтеграції з криптографічними функціями мова дозволяє легко підключати високоефективні бібліотеки, такі як OpenSSL чи mbed TLS.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

У C++ реалізовано статичну типізацію, тобто тип кожної змінної визначається та перевіряється на етапі компіляції, що мінімізує виникнення помилок під час виконання програми. Мова підтримує як базові вбудовані типи, числові різних розмірів, символічні, бульові, так і складні структури даних із бібліотеки STL, вектори, карти, черги. Всі змінні та об'єкти за замовчуванням виділяються на стеку або в купі, а розробник може чітко керувати способом передачі даних – за значенням, за посиланням або через покажчики, що критично важливо для уникнення зайвого копіювання у системах з обмеженою пам'яттю.

Для проекту, що реалізує гібридну криптосистему, AES–256 та ECC для захисту трафіку протоколу MQTT в системах Інтернет–речей, вибір C++ є ключовим. Пристрої Інтернету речей мають жорсткі обмеження щодо обчислювальних ресурсів, об'єму оперативної пам'яті та енергоспоживання. C++ дозволяє виконувати складні математичні перетворення алгоритмів AES і операції на еліптичних кривих (множення точки на скаляр) з мінімальними накладними витратами і максимальною швидкістю в реальному часі.

Мова забезпечує чітку модульність і масштабованість, завдяки чому криптографічний модуль та клієнтський MQTT–транспорт інтегруються в єдине стабільне рішення. C++ – це стабільна, перевірена часом мова, яка є галузевим стандартом для розробки вбудованих систем кібербезпеки та IoT–рішень, де компроміси між ресурсоемістю, надійністю та швидкістю обробки даних є неприпустимими.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки Інтернету речей (IoT) на основі гібридного шифрування даних за протоколом MQTT.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести комплексний аналіз існуючих криптографічних рішень, засобів захисту вбудованих систем та аналогів проектованої системи для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови гібридної системи кібербезпеки (на базі симетричного алгоритму AES–256 та асиметричної криптографії на еліптичних кривих ECC Curve25519) для забезпечення конфіденційності, цілісності та автентичності даних у мережах Інтернету речей. Розробити структурну, функціональну схеми системи та діаграму процесів взаємодії компонентів;

в) розробити програмне забезпечення системи мовою C++, що дозволить реалізувати захищений обмін інформацією за протоколом MQTT між IoT–клієнтами та брокером, включаючи генерацію ключів, шифрування повідомлень та автентифікацію пристроїв. Побудувати блок–схеми алгоритмів функціонування основної програми та криптографічних підпрограм;

г) організувати інтерфейс моніторингу та логування процесів з метою формування та виводу на екран ЕОМ (або консоль адміністратора) інформації про стан з'єднання, успішність криптографічних операцій, а також повідомлень про некоректні дії користувача, спроби несанкціонованого доступу та нестандартні ситуації в роботі системи;

д) розробити рекомендації по організаційних та технічних заходах, які забезпечать впровадження розробленої системи кібербезпеки IoT–пристроїв у практичну експлуатацію, її подальше технічне обслуговування, модернізацію та масштабування;

е) провести розрахунки по визначенню економічної ефективності розробленої системи, оцінити витрати на розробку програмного забезпечення та доцільність його впровадження;

ж) розробити заходи по охороні праці при впровадженні, налаштуванні та експлуатації обчислювальної техніки та компонентів IoT-мережі, а також розробити заходи з цивільного захисту в умовах виникнення надзвичайних ситуацій;

з) сформувати висновки про виконаний обсяг робіт, оцінити ступінь досягнення поставленої мети та підсумувати одержані практичні результати.

К6ПЗ-2026

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

На основі проведеного аналізу сучасної ситуації відносно кібербезпеки у середовищі Інтернет речей, а також, дослідження багатьох існуючих криптографічних методів захисту інформації, зокрема алгоритмів шифрування, було сформовано концепцію функціонування програмного забезпечення системи кіберзахисту для ресурсно–обмежених пристроїв сфери Інтернет–речей. Основною метою розробки є створення ефективного, надійного та оптимізованого, з точки зору ресурсів, рішення, яке повинне забезпечити захист даних під час їх передачі та обробки.

Програмне забезпечення, що розробляється, за своєю концепцією, базується на принципах комплексності у питаннях забезпечення кібербезпеки. Розроблена система, як зазначалося вище, використовує поєднання двох криптографічних алгоритмів, для захисту даних безпосередньою а також, для обміну даними між пристроями, використовує легкий мережний протокол MQTT. Його придатність до використання в ресурсно обмежених системах, асинхронна модель взаємодії між компонентами і передачі інформації, а також низьке мережне навантаження, що той викликає, зумовлює MQTT бути найкращим варіантом в контексті роботи.

Також, необхідним фактором забезпечення конфіденційності при передачі інформації, в розроблюваній системі використовується алгоритм шифрування AES, завдяки його високій швидкості та ефективності. Водночас, для організації захищеності під час обміну криптографічними ключами планується використання алгоритму ECC, що дозволить мінімізувати потужність, але зберегти захищеність.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

Функціонально розроблюване програмне забезпечення можна розглядати як сукупність компонентів. До цієї сукупності входять:

– пристрій Інтернет–речей, який виконує роль клієнта і такі функції:

- а) збір даних з апаратного елемента;
- б) обробка отриманої інформації;
- в) попереднє шифрування інформації перед передачею;

– комунікаційне середовище, що забезпечує:

- а) передачу даних між компонентами системи;
- б) асинхронну взаємодію;
- в) маршрутизацію повідомлень за допомогою брокера;

– пристрій Інтернет–речей, який виконує роль сервера і такі функції:

- а) прийом зашифрованих повідомлень;
- б) дешифрування отриманої інформації;
- в) перевірка цілісності та автентичності інформації;

г) виконання всіх необхідних стадій обробки даних (зокрема, збереження).

Головною особливістю запропонованого концепту виступає максимальна орієнтованість на мінімізацію витрат та використання в умовах низької розрахункової потужності системи, як у випадках апаратної обмеженості, так і в випадках коли у вигляді цілей виступає економія. Це досягається завдяки використанню компонентності системи, а саме, розподілу функцій між її компонентами. Крім того, концепція передбачає впровадження базових механізмів захисту від типових атак, таких як підміна даних або повторне відтворення повідомлень.

Таким чином, описана концепція визначає загальні принципи і засади, на основі яких буде здійснюватись функціонування спроектованого програмного забезпечення для захисту системи Інтернет–речей. Це створює гарну основу для опису структурної схеми, опису алгоритмів і інших аспектів подальшої деталізації, що буде розглянуто у подальших підрозділах.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

Додатково, серверний рівень забезпечує централізоване ведення журналів подій та моніторинг стану функціонування системи. Це дозволяє здійснювати фіксацію спроб підключення, реєстрацію процесів обміну криптографічними ключами, а також аналіз можливих помилок або аномальної активності в роботі IoT-пристроїв. Наявність такого механізму підвищує рівень керованості системи та спрощує процес виявлення потенційних загроз безпеці. Окрім цього, централізований контроль подій створює можливість подальшого масштабування програмного забезпечення та інтеграції додаткових засобів аналізу мережевого трафіку й виявлення вторгнень.

Організація керування сесійними ключами

Важливим аспектом реалізації запропонованого концепту є організація процесу керування сесійними криптографічними ключами в умовах динамічної зміни складу мережі IoT. Концепція передбачає регулярне оновлення симетричних ключів шифрування AES-256 через визначені проміжки часу або після передачі встановленого обсягу телеметричних даних, що мінімізує ризик компрометації каналу зв'язку в разі довготривалого перехоплення трафіку злоумисником. При цьому процедура автентифікації нових IoT-клієнтів та початковий обмін секретними параметрами на основі еліптичних кривих (ECC) інтегруються безпосередньо у службові пакети розширеного протоколу MQTT (на етапі встановлення сесії CONNECT / CONNACK), що дозволяє уникнути створення додаткових високонавантажених мережевих з'єднань.

Алгоритм AES

Алгоритм шифрування AES використовує блоки даних, розмір яких є фіксованим. Блок даних має розмір сто двадцять вісім біт, і вартим зазначення є факт, що на схемі роботи алгоритму вказано таблицю розміром 4x4, тому що в наведеному контексті таблиця відображає блок даних в байтах. Тобто кожен елемент таблиці блоку даних є одним байтом інформації.

Алгоритм шифрування AES підтримує три варіанти обробки даних. Який варіант буде використано залежить від довжини ключа, на основі чого визначається кількість ітерацій при шифруванні. Такими варіантами є:

- 128 біт і 10 раундів;
- 192 біт і 12 раундів;
- 256 біт і 14 раундів.

Безпосередньо перед початком шифрування повинне бути виконане розширення ключа. Кожен наступний раунд з послідовності має унікальний ключ що забезпечує додаткову захищеність.

Як можна помітити, на схемі вказано нульовий раунд шифрування. На відміну від наступних ітерацій, на даному раунді виконується лише операція `addRoundKey`, в процесі виконання якої виконується `XoR` вхідних даних з ключем нульового раунду. Це дозволяє досягнути дифузії навіть до початку процесу шифрування, тобто приховати статистичну структуру зашифрованих даних за рахунок поширення впливу символу вхідних даних на зашифрований в подальшому текст.

Кожна з наступних ітерацій, раундів шифрування, послідовно виконує наступні чотири функції:

- функція `SubBytes` – виконує заміну кожного окремого байта матриці задіюючи таблицю підстановок. Це виконує роль одного з основних методів захисту даних від найпоширеніших видів криптоаналізу;
- функція `ShiftRows` – являє собою зациклений зсув рядків матриці ліворуч. Перший ряд залишається на місці, другий зсовується на один байт, другий на два, третій і наступні відповідно;
- функція `MixColumns` – є математичним перетворенням, в якому кожен з стовпців множиться на фіксовану матрицю у скінченному полі $GF(2^8)$;
- функція `AddRoundKey` – результат об'єднання отриманої послідовності з раундовим ключем.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

В процесі виконання останнього етапу шифрування операція MixColumns не виконується, через що раунд викликає виключно SubBytes, ShiftRows і AddRoundKey. За рахунок чого структура алгоритму і є симетричною. В результаті виконання отримуємо блок зашифрованих даних розміром 128 біт.

Звісно, використання AES не здатне повністю покрити собою потребу в захищеності системи. Додатковим рівнем захисту було обрано використання асиметричного алгоритму шифрування ECC, також описаного вище, для захищеного обміну ключами та успішного використання AES, що і робить його достатнім. ECC базується на використанні математичного апарату еліптичних кривих над скінченними полями. Основною особливістю даного підходу є забезпечення високого рівня криптографічної стійкості при значно меншій довжині ключа у порівнянні з класичними асиметричними алгоритмами, такими як RSA. Тобто еліптичні криві вигляду $y^2 = x^3 + ax + b$, де всі операції виконуються у скінченному полі. Точки, що належать цій кривій, разом з оператором додавання утворюють групу, яка і використовується, вже безпосередньо в роботі самого алгоритму. Основною операцією в ECC є множення точки на скаляр. Доцільно розглянути більш детально. Точка багаторазово додається до самої себе. Саме ця операція є обчислювально простою в прямому напрямку, але практично незворотною, завдяки цьому і забезпечується криптостійкість цього алгоритму.

Саме з цієї причини алгоритми на основі еліптичних кривих набули широкого поширення у сучасних системах кібербезпеки, особливо у середовищах із обмеженими обчислювальними ресурсами, до яких належать IoT-пристрої. Використання ECC дозволяє суттєво зменшити обсяг обчислень, необхідних для генерації ключів та виконання криптографічних операцій, що безпосередньо впливає на швидкодію системи.

Алгоритм ЕСС

Як вказано на схемі, в основі процесу шифрування використовується математичне рівняння кривої над скінченним полем F_p , що має такий вигляд:

$$y^2 = x^3 + ax + b \pmod{p} \quad (3.1)$$

де a і b є параметрами, що визначають форму кривої. В свою чергу, p є великим простим числом. Саме відштовхуючись від нього задаються межі поля. Варто зазначити, що всі розрахунки відбуваються дискретно, що критично важливо в контексті шифрування саме цим алгоритмом.

Криптостійкість цього алгоритму базується на двох основних діях:

– Додавання точок:

$$P + Q = R \quad (3.1)$$

Пряма проводиться через точки P та Q . Крива перетинається в третій точці, що в подальшому буде дзеркально відображено відносно осі O_x .

– Множення на скаляр:

$$kP \quad (3.2)$$

Точка P багаторазово додається до себе k кількість разів.

Ключовим принципом є обчислення за формулою:

$$Q = d \times G \quad (3.3)$$

де G є базовою точкою. За рахунок цих дій досягається факт, що знаходження числа d , знаючи тільки Q і G практично не є можливим. В результаті вказаних обчислень формується стандартна задача дискретного логарифму на еліптичних кривих. В процесі подальшої роботи алгоритму створюється пара ключів, приватний та публічний. Приватний ключ є випадковим числом що залишається в таємниці, тоді як публічний ключ отриманий в результаті обрахування може залишатися неприхованим.

Використання протоколу ECDH обумовлене потребою створення спільного секретного ключа та обміну ним відкритим каналом.

– Сторона А використовує пару (d_A, Q_A) і надсилає Q_A стороні Б;

– Сторона Б використовує пару (d_B, Q_B) і надсилає Q_B стороні А;

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

Після вказаних дій дві сторони обчислюють спільний секрет S в процесі множення свого приватного ключа на публічний ключ іншої сторони.

– Для сторони А: $S_A = d_A \times Q_B = d_A \times (d_B \times G)$.

– Для сторони Б: $S_B = d_B \times Q_A = d_B \times (d_A \times G)$.

У результаті маємо $S_A = S_B$ ідентичні.

Секретний ключ S отриманий в результаті не застосовується як ключ напряму. Він обробляється функцією виведення ключа, в результаті чого отримуємо більш стабільний та безпечний у використанні ключ.

Також, варто зазначити, щодо використання легкого протоколу mqtt для передачі інформації та обміну нею між пристроями системи. Використання mqtt є базовим для Інтернет-речей та найпростішим в реалізації. MQTT це легкий мережевий протокол передачі даних, який працює за моделлю “видавець–підписник”. Основною особливістю даного протоколу є мінімальні вимоги до пропускної здатності мережі та обчислювальних ресурсів. Це досягається за рахунок простоти протоколу та компактного формату повідомлень, що максимально доцільно у контексті поставленої задачі. Це робить його ефективним для використання у середовищах з нестабільним або, що куди більш важливо, повільним з’єднанням. Принцип роботи MQTT базується на використанні центрального вузла брокера, який відповідає за приймання, обробку та розповсюдження повідомлень між клієнтами. Клієнти, в свою чергу, при цьому можуть виступати як у ролі відправників повідомлень, так і отримувачів. MQTT підтримує декілька рівнів якості обслуговування (QoS), що визначають надійність доставки повідомлень.

– QoS0–доставка без підтвердження (найшвидший, але найменш надійний варіант);

– QoS1–доставка з підтвердженням (гарантована хоча б одна доставка);

– QoS2–доставка без дублювання (найвищий рівень надійності).

Безпека MQTT досягається шляхом використання додаткових механізмів, таких як шифрування, що буде досягнуто за рахунок використання алгоритмів описаних вище.

На відміну від традиційної моделі, де клієнт звертається безпосередньо до сервера, MQTT використовує модель Видання/Підписки. Учасники системи роз'єднані: вони не знають про адресу або стан один одного, а вся комунікація проходить через центральний вузол.

Ключовими компонентами системи є:

- клієнт–видавець (Publisher): IoT–пристрій (наприклад, датчик температури), який збирає дані та відправляє їх у мережу;
- MQTT брокер (Broker): «диспетчер» системи, який приймає всі повідомлення, перевіряє, кому вони адресовані, розсилає їх відповідним отримувачам, а також відповідає за автентифікацію клієнтів;
- клієнт–підписник (Subscriber): застосунок або термінал, який «підписується» на певні типи даних (теми) і отримує їх у реальному часі.

Кожне повідомлення в MQTT складається з двох основних частин:

- 1) тема (Topic): ієрархічний рядок (наприклад, sensor/temperature), що слугує адресою. Брокер використовує його, щоб зрозуміти, кому переслати дані;
- 2) корисне навантаження (Payload): самі дані, зазвичай у форматі JSON (наприклад, {"temp": 24.6, "unit": "C"});

Згідно зі схемою, взаємодія відбувається за таким алгоритмом:

- 1) встановлення зв'язку: клієнти підключаються до брокера через TCP/IP з'єднання, на цьому етапі може відбуватися перевірка логіна та пароля;
- 2) публікація: видавець надсилає сформоване повідомлення брокеру;
- 3) підписка: підписник вказує брокеру, які теми його цікавлять;
- 4) розповсюдження: як тільки брокер отримує нове повідомлення, він миттєво дублює його всім підписникам, що чекають на цю тему;

Подібна архітектура забезпечує високу масштабованість системи, оскільки видавці та підписники не взаємодіють між собою напрямую, а працюють

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

виключно через MQTT-брокер. Завдяки цьому до мережі можливо одночасно підключати велику кількість IoT-пристроїв без необхідності зміни структури взаємодії між компонентами системи. Додатковою перевагою є асинхронний характер обміну даними, при якому пристрої можуть надсилати інформацію незалежно від активності інших компонентів мережі, що особливо важливо для систем моніторингу та збору телеметричних даних у режимі реального часу.

3.2 Розробка структурної схеми

Під час проектування програмного забезпечення системи захисту у сфері Інтернет-речей, що відповідає сформованим вище вимогам у роботі в умовах пристроїв обмежених у розрахунковій потужності, розроблено структурну схему. Побудова такої схеми є необхідною, оскільки це дозволить формалізувати саму архітектуру системи та більш практично і точно навести функціональні зв'язки її складових. На структурній схемі відображено основні компоненти наведеної системи, їх взаємодію між собою та логіку обміну даними між ними під час роботи. Структурна схема розробленого програмного забезпечення базується на розподільній архітектурі і включає такі рівні:

- Рівень пристроїв.
- Рівень комунікації.
- Серверний рівень.

Кожен з цих рівнів має виконувати визначені функції та взаємодіяти з іншими рівнями, як і з якими буде описано далі. Рівень пристроїв містить в себе вузли Інтернет-речей, до яких, безпосередньо, і вносяться дані з зовнішніх джерел чи елементів вводу. Також, в межах програмної реалізації, на рівні пристроїв здійснюється первинна обробка інформації, тобто формування повідомлень та їх шифрування алгоритмом шифрування AES. Окрім цього, на даному рівні відбувається генерація криптографічних ключів шифрування, в обміні якими він і візьме участь в процесі роботи пізніше.

Комунікативний рівень

Комунікативний рівень є мережним середовищем, що представляє спосіб передачі даних між клієнтом чи клієнтами та сервером. Як вже було зазначено, тут використовується протокол MQTT, який функціонує на засадах моделі publish/subscribe. Центральний елемент рівня комунікацій це брокер повідомлень. Брокер повідомлень відповідальний за приймання, маршрутизацію а також доставку повідомлень до відповідного отримувача. За рахунок цього, досягається масштабованість системи завдяки можливостям пристроїв обмінюватися даними асинхронно, що є одним з факторів зменшення навантаження системи.

Серверний рівень

Далі наведено опис серверного рівня. Серверний рівень виконує функціонал аналізу отриманої інформацію, і що набагато більш важливо, їх приймання та дешифрування. Саме на цьому етапі, в результаті дешифрування, текст приймає початковий вигляд з зашифрованого. Також відбувається перевірка їх цілісності та автентичності. На серверному рівні діє захист від типових атак, цей рівень просто не дозволяє повторне відтворення однакових повідомлень. Тут і розгортається криптографічний алгоритм ECC, який дозволяє сформувати спільний секретний ключ без прямої його передачі у відкритому вигляді.

Комунікативний рівень

Важливою особливістю комунікативного рівня є також підтримка механізмів гнучкої маршрутизації повідомлень за допомогою системи тем MQTT. Завдяки цьому кожен підписник отримує виключно ті дані, на які він був попередньо підписаний, що дозволяє суттєво зменшити надлишковий мережевий трафік та оптимізувати використання пропускної здатності каналу зв'язку. Такий підхід є особливо актуальним для IoT-систем, у яких велика кількість пристроїв одночасно здійснює передачу невеликих обсягів телеметричної інформації у режимі реального часу. Додатково, використання MQTT забезпечує стабільну

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

роботу системи навіть у мережах із нестабільним з'єднанням або низькою швидкістю передачі даних.

Серверний рівень, окрім функцій дешифрування та перевірки повідомлень, також виконує роль центрального елемента контролю безпеки всієї системи. Саме тут відбувається аналіз отриманих даних, перевірка допустимості джерела повідомлення та контроль часових міток, що дозволяє виявляти спроби несанкціонованого доступу або повторної передачі вже оброблених пакетів інформації.

Реєстрація подій

Окрему роль у роботі серверного рівня відіграє механізм журналювання подій та моніторингу активності системи. Завдяки цьому здійснюється реєстрація процесів підключення пристроїв, обміну повідомленнями та виконання криптографічних операцій. Такий підхід дозволяє своєчасно виявляти помилки функціонування, аномальну активність або потенційні спроби втручання у роботу системи. Крім цього, централізоване збереження журналів подій створює основу для подальшого аналізу безпеки, аудиту роботи системи та інтеграції додаткових механізмів виявлення мережових атак і загроз.

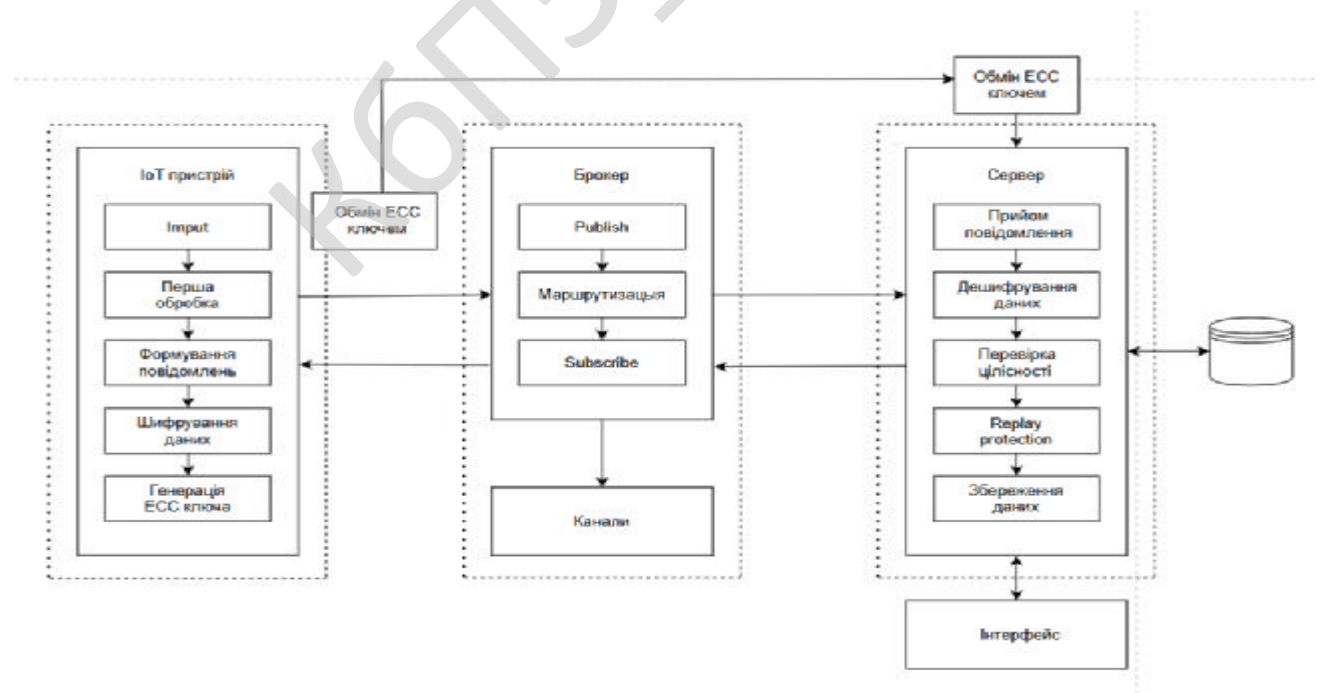


Рисунок 3.1 – Структурна схема системи захисту інформації

3.3 Розробка функціональної схеми

Як вже було зазначено, алгоритм кіберзахисту, який не можна повноцінно назвати алгоритмом шифрування, в більшій частині орієнтований на безпечну передачу інформації в умовах середовища інтернет речей в якому, як обумовлюється вимогами до розробки, маємо обмежену розрахункову потужність апаратної частини. Головною ідеєю описаного нижче алгоритму є використання гібридного підходу в захисті. Це дозволяє поєднати швидкодію та ефективність симетричного алгоритму шифрування з захищеністю та безпечністю використання асиметричних алгоритмів шифрування, для обміну ключами, в даному випадку.

Перш ніж починати детальний технічний опис роботи та розробки програмного забезпечення, варто зазначити, що на наведеному етапі роботи реалізація пристроїв Інтернет–речей буде виконана завдяки програмній емуляції. Цей фактор обумовлений декількома практичними та методичними причинами. Використання фізичних пристроїв (наприклад, мікроконтролерів) потребує не тільки обладнання, а й часу на налаштування, що ускладнить процес тестування та додасть до нього кілька зайвих етапів, істотно знижуючи ефективність розробки програмного забезпечення за тематикою роботи.

Емуляція програмним шляхом, натомість, дозволить швидко змінювати поведінку пристрою безпосередньо в програмі, що забезпечить можливість коригувати сценарії роботи програмного забезпечення без витрат часу на використання реального апаратного забезпечення;

Окрім зазначеного, вибір саме емуляції пристрою Інтернет–речей має значно спростити процес налагодження взаємодії криптографічних інструментів, що є критично необхідним у цьому випадку, оскільки дає змогу контролювати кожен етап обробки, обміну даними та генерації ключів незалежно;

Як одну з найважливіших переваг використання такої моделі можна зазначити її відтворюваність та універсальність, адже розроблюване програмне

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		26

забезпечення може бути розгорнуте на будь-якій машині без прив'язки до конкретного апаратного середовища;

У такому випадку логіка роботи алгоритму абсолютно відповідатиме реальним умовам функціонування пристроїв Інтернет-речей, що дозволить у подальшому, за потреби, перенести розроблену систему в реальне апаратне середовище без суттєвих змін.

Додатково, використання програмної емуляції дозволяє значно підвищити гнучкість процесу тестування системи захисту, оскільки надає можливість моделювати різні сценарії роботи мережі Інтернет-речей. Зокрема, це стосується емуляції нестабільного мережевого з'єднання, зміни кількості підключених пристроїв, перевірки роботи механізмів повторного підключення та аналізу поведінки системи під час передачі пошкоджених або неавторизованих повідомлень. Завдяки цьому можна більш детально оцінити стійкість розробленого програмного забезпечення до типових проблем та потенційних атак, які можуть виникати під час експлуатації IoT-систем у реальних умовах.

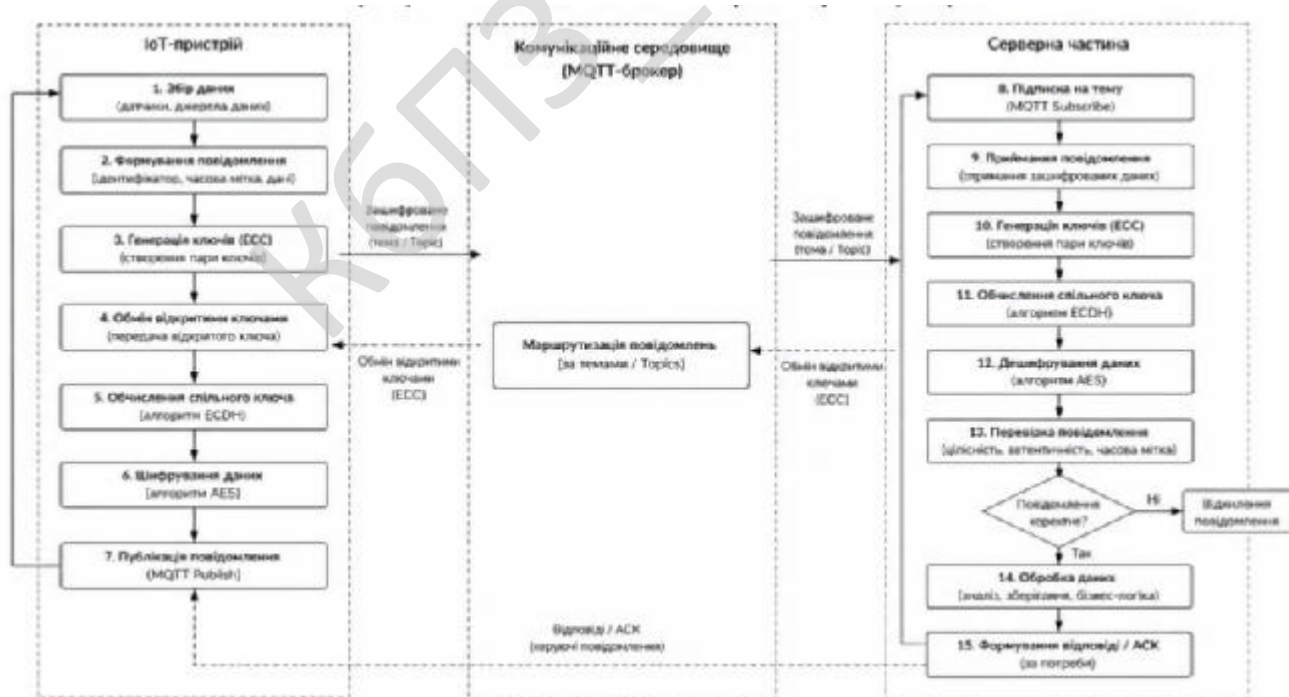


Рисунок 3.2 – Функціональна схема розробленого ПЗ

Більш детально про сам розроблений алгоритм. Алгоритм буде описано за етапами роботи. На початковому етапі, система ініціалізує криптографічні параметри, як на стороні пристрою Інтернет–речей, так і на стороні серверу відповідно. Це відбувається майже одночасно. Кожна з наведених сторін, генерує власну пару криптографічних ключів, при цьому використовується алгоритм шифрування ЕСС. Це зумовлюється потребою отримати та зберігати істотно менші розміри ключів у порівнянні з більш традиційними методами.

3.4 Розробка діаграми процесів

На наступному етапі, відразу після генерації ключів, виконується процедура з'єднання, в момент виконання якої сторони вже обмінюються відкритими ключами. Відштовхуючись від отриманих таким чином відкритих ключів сторони окремо обчислює спільний секретний ключ керуючись і використовуючи механізм ECDH. Головним фактором є те, що отриманий таким чином секретний ключ не передається безпосередньо каналом зв'язку. Це унеможливорює його перехоплення гіпотетичним зловмисником. Після наведених операцій з секретним ключем він обраховується хеш–функцією. В результаті система отримує ключ фіксованої довжини. На наступному етапі формується саме повідомлення. Також, повідомлення наповнюється інформацією. Воно буде включати ідентифікатор машини з якої було надіслане, часову мітку, що є необхідним з точки зору стандартів і корисне навантаження, тобто дані які передаються. Важливо зазначити, що формування часової мітки в повідомленні дозволить в подальшому працювати захисту від атак типу повторне відправлення. Після завершення формування повідомлення виконується його серіалізація у JSON-формат та подальше шифрування за допомогою алгоритму AES-256 із використанням попередньо сформованого shared key, що забезпечує конфіденційність інформації під час її передачі через мережеве середовище.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		28

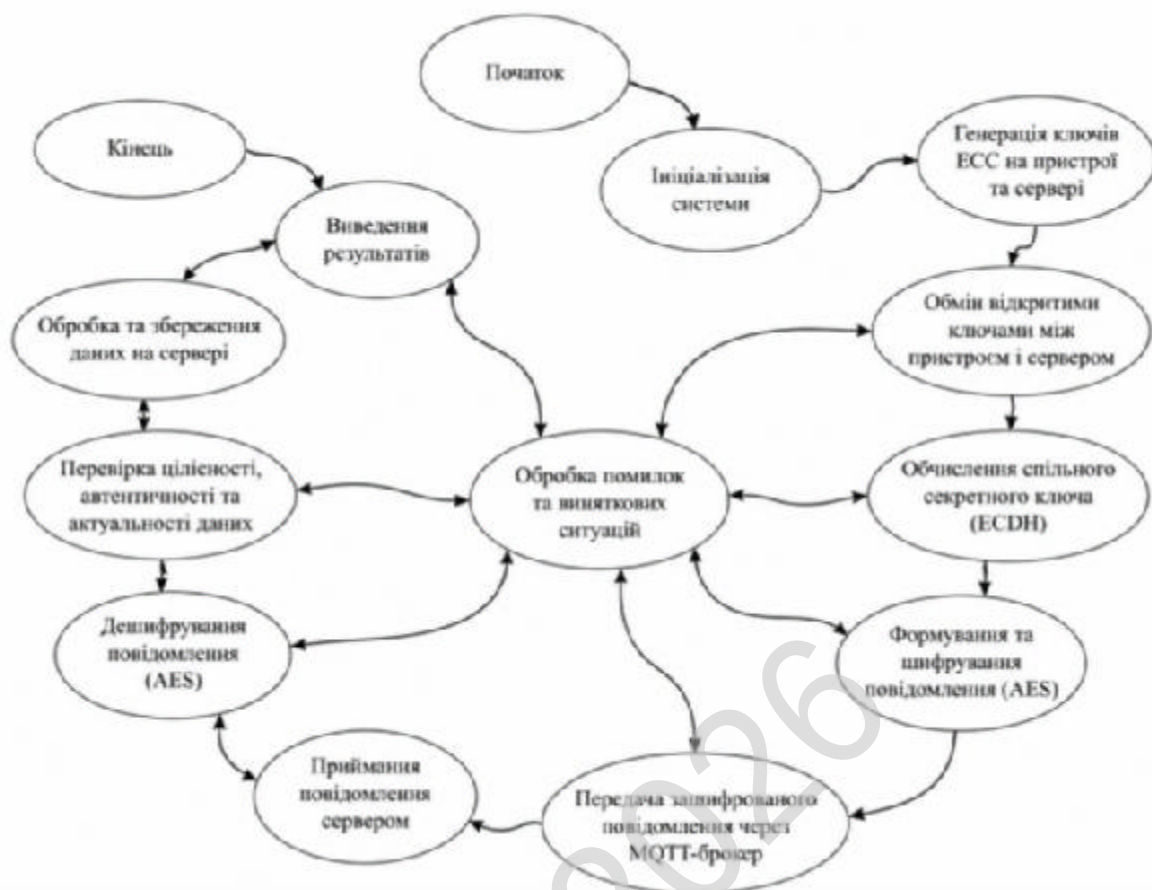


Рисунок 3.3 – Діаграма процесів розробленого ПЗ

По завершенню процесу формування повідомлення передається до комунікативного середовища завдяки роботі протоколу MQTT. Як було описано вище, пересилання здійснюється за моделлю publish/subscribe. Це дозволяє розділити відправника та отримувача і значно зменшити навантаження на систему, тобто використання ресурсу потужності. Масштабованість системи досягається на даному етапі теж завдяки брокера посередника реалізованого завдяки MQTT. Сервер, на наступному етапі, отримує вже зашифроване повідомлення, дешифрує його з використанням такого ж ключа, що є симетричним до ключа пристрою. Цей ключ, було сформовано ще на одному з попередніх етапів, тобто на етапі обміну.

Відразу після отримання сервером відновленого повідомлення сервер перевіряє коректність його структури, автентичність пристрою відправника та проводиться запис часової мітки. У випадку виникнення невідповідності

виконується відхилення повідомлення, що сприяє захисту від атак пов'язаних зі шкідливим вмістом. Завершальним етапом є обробка отриманої інформації алгоритмом вже на сервері. Отримані дані будуть зберігатися у простій базі даних, та, у випадку розширення системи, заноситися до системи моніторингу чи до аудиту. Наведений підхід дозволить повністю реалізувати цикл обробки даних.

Додатково, важливо зазначити, що передача інформації у зашифрованому вигляді суттєво підвищує загальний рівень захищеності системи, оскільки навіть у випадку перехоплення мережевого трафіку зломисник не матиме можливості отримати доступ до змісту повідомлення без наявності відповідного криптографічного ключа. Використання механізму попереднього обміну ключами за допомогою ECC у поєднанні з симетричним алгоритмом AES дозволяє забезпечити як високий рівень криптографічної стійкості, так і достатню швидкість системи в умовах обмежених апаратних ресурсів IoT-пристроїв. Подібний підхід є характерним для сучасних захищених мережевих систем та використовується у великій кількості реальних протоколів безпечного зв'язку.

Окремою перевагою запропонованого підходу є можливість забезпечення стабільної роботи системи навіть за умов великої кількості одночасно підключених пристроїв. Завдяки використанню MQTT брокер виконує роль проміжного вузла, який оптимізує процес доставки повідомлень та зменшує необхідність прямої взаємодії між усіма елементами мережі. Це не лише спрощує масштабування системи, але й дозволяє знизити навантаження на серверну частину, оскільки обробка повідомлень виконується асинхронно. Додатково, така архітектура забезпечує більшу відмовостійкість системи, адже навіть у випадку тимчасової втрати з'єднання окремими пристроями MQTT підтримує механізми повторної доставки повідомлень та відновлення сеансів зв'язку.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

4. РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок–схем та опис алгоритмів функціонування системи

Для успішної програмної реалізації системи захисту інформації було обрано мову програмування C++. Також, важливим фактором є застосування об'єктно–орієнтованого підходу до програмування, як базис без якого неможливо реалізувати подібну систему. Основною метою реалізації було створення моделі захищеної системи обміну інформації між пристроєм Інтернет–речей і серверною частиною програмно, використовуючи і поєднуючи найоптимальніші криптографічні рішення.

Архітектурно, розроблене програмне забезпечення побудоване за принципами модульності. Такий підхід дозволяє вдало розділити процеси функціонування системи на окремі компоненти логічно. Це дозволяє значно спростити процес розробки та подальшого коригування і модернізації в життєвому циклі програмного забезпечення. Система включає в себе такі модулі:

- Криптографічний модуль.
- Модуль формування повідомлень.
- Емулятивний модуль.
- Серверний модуль.

Найважливішим модулем системно є криптографічний зазначений як CryptoManager. Він відповідає за роботу механізмів пов'язаних з генерацією криптографічних ключів. Тобто формування спільного секретного ключа і шифрування, дешифрування даних. В цілях реалізації асиметричного обміну інформацією використано алгоритм ЕСС. В цілях забезпечення конфіденційності

при передачі даних використано алгоритм AES. В даному випадку довжина ключа дорівнює 256 біт.

Також важливим є модуль формування повідомлень. Він представлений класом Message, що використовується в процесі формування структури повідомлення. В кожне повідомлення послідовно включається ідентифікатор пристрою–відправника, часову мітку і корисне навантаження, тобто самі дані, що будуть переданими. З метою досягнення універсальності повідомлення передається у форматі JSON, цей вибір зроблено з оглядом на подальші перспективи інтеграції системи, що будуть описані в подальшому.

Емулятивний модуль винесено в окремий клас IoTDevice і реалізує функціонал з'єднання з сервером і захищеності такого з'єднання і передачі інформації. Даний клас дозволяє емулювати роботу пристрою Інтернет–речей без необхідності застосування і налагодження фізичного апаратного забезпечення, доцільність чого була визначена раніше.

Серверний модуль представлений програмно класом Server. Він відповідає за всі функції які повинен виконувати сервер, тобто прийняття зашифрованих повідомлень а також дешифрування таких повідомлень, перевірка їх автентичності і перевірка автентичності пристрою. Якщо говорити більш лаконічно, клас сервер відповідає за отримання і обробку даних.

Взаємодія між компонентами системи виконується поетапно. Спочатку відбувається обмін відкритими ключами між сервером та емульованим пристроєм відкритими ключами, відразу після чого формується спільний секретний ключ. Встановлюється захищене з'єднання, після чого вже створюється повідомлення. Це повідомлення проходить процес шифрування та надсилається на серверну частину, де і задіюється весь її функціонал. Реалізація програмного забезпечення була б неможлива без використання бібліотеки OpenSSL, що дозволяє використовувати стандартизовані криптографічні алгоритми простим викликом в програмі і лишило ПЗ потреби в прямій низькорівневій реалізації алгоритмів шифрування власноруч.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

4.2 Захист розробленого програмного забезпечення

У межах розробленого програмного забезпечення криптографічний функціонал представлений класом CryptoManager. Цей клас інкапсулює кожну операцію з тих, що пов'язані з генерацією ключів, створенням спільного секретного ключа а також обробкою зашифрованих даних. Технічної реалізації криптографічних механізмів досягнуто завдяки використанню бібліотеки OpenSSL. Це забезпечує максимально оптимальне використання криптографічних алгоритмів. Організація безпечного обміну ключами досягається застосуванням алгоритму ECC. В процесі ініціалізації криптографічного модуля створюється пара ключів на основі еліптичної кривої зазначеної як prime256v1. Генерація ключів стає виконаною автоматично, разом зі створенням об'єкту класу CryptoManager. Публічний ключ застосовується під час обміну ключами з серверною частиною. На відміну від цього, приватний ключ залишається виключно локальним та не виходить у мережу.

Фрагмент процесу генерації ключів:

```
key = EC_KEY_new_by_curve_name(NID_X9_62_prime256v1);  
EC_KEY_generate_key(key);
```

Відразу після створення пари ключів формується спільний секретний ключ за застосуванням механізму ECDH. Під час обміну відкритими ключами обидві сторони самостійно обчислюють однаковий вільний секретний ключ. Він буде використаний як основа для симетричного шифрування даних в подальшому. Класична хеш-функція SHA-256 використовується для формування криптографічного ключа фіксованої довжини. Далі, створений таким чином ключ, буде застосовано, безпосередньо, під час виконання алгоритму симетричного шифрування AES-256.

Фрагмент реалізації формування спільного ключа:

```
ECDH_compute_key(  
    shared_key.data(),  
    secret_len,  
    peer_pub,
```

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

```

        key,
        NULL
    );

    SHA256(
        shared_key.data(),
        shared_key.size(),
        hash
    );

```

Конфіденційність в системі забезпечується завдяки виконанню алгоритму AES з довжиною ключа що дорівнює 256 біт. В межах даної програмної реалізації використано варіант з режимом шифрування CFB, чи Cipher Feedback Mode. Це дозволяє виконувати потокове шифрування даних за змінної довжини але дозволяє запобігти необхідності попереднього розбиття повідомлення на блоки. Також, для кожного повідомлення додатково генерується випадковий вектор ініціалізації IV.

Фрагмент реалізації шифрування повідомлення:

```

EVP_EncryptInit_ex(
    ctx,
        EVP_aes_256_cfb(),
        NULL,
        shared_key.data(),
        iv
);
EVP_EncryptUpdate(
    ctx,
        ciphertext.data(),
        &len,
        (unsigned char*)data.c_str(),
        data.size()
);

```

Після завершення процесу передачі зашифрованого повідомлення, процес дешифрування виконується серверною частиною. З повідомлення вилучається

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

згаданий вектор ініціалізації. Початкові дані відновлюються за рахунок використання такого самого ключа.

Фрагмент реалізації дешифрування:

```
EVP_DecryptInit_ex(  
    ctx,  
    EVP_aes_256_cfb(),  
    NULL,  
    shared_key.data(),  
    iv  
);  
  
EVP_DecryptUpdate(  
    ctx,  
    plaintext.data(),  
    &len,  
    encrypted.data() + 16,  
    encrypted.size() - 16  
);
```

В межах розробленої програми структура повідомлень формується завдяки класу Message. Також він відповідає за підготовку даних до передачі. Застосування саме такого підходу дозволяє інкапсулювати логіку взаємодій з повідомленням та дозволяє забезпечити її використання в різних частинах роботи. Кожне повідомлення в системі містить набір полів, що є мінімально необхідними. Він забезпечує можливість ідентифікувати пристрій, визначити часову мітку і формувати дані для подальшого відправлення. Технічно до повідомлення входять:

- device_Id;
- timestamp;
- payload.

Саме такий склад повідомлення дозволяє захиститися від розповсюджених атак типу повторного відправлення і забезпечує можливість ідентифікувати надсилача серверною стороною.

Формування повідомлення:

```
Message msg(device_id, data);
```

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		35

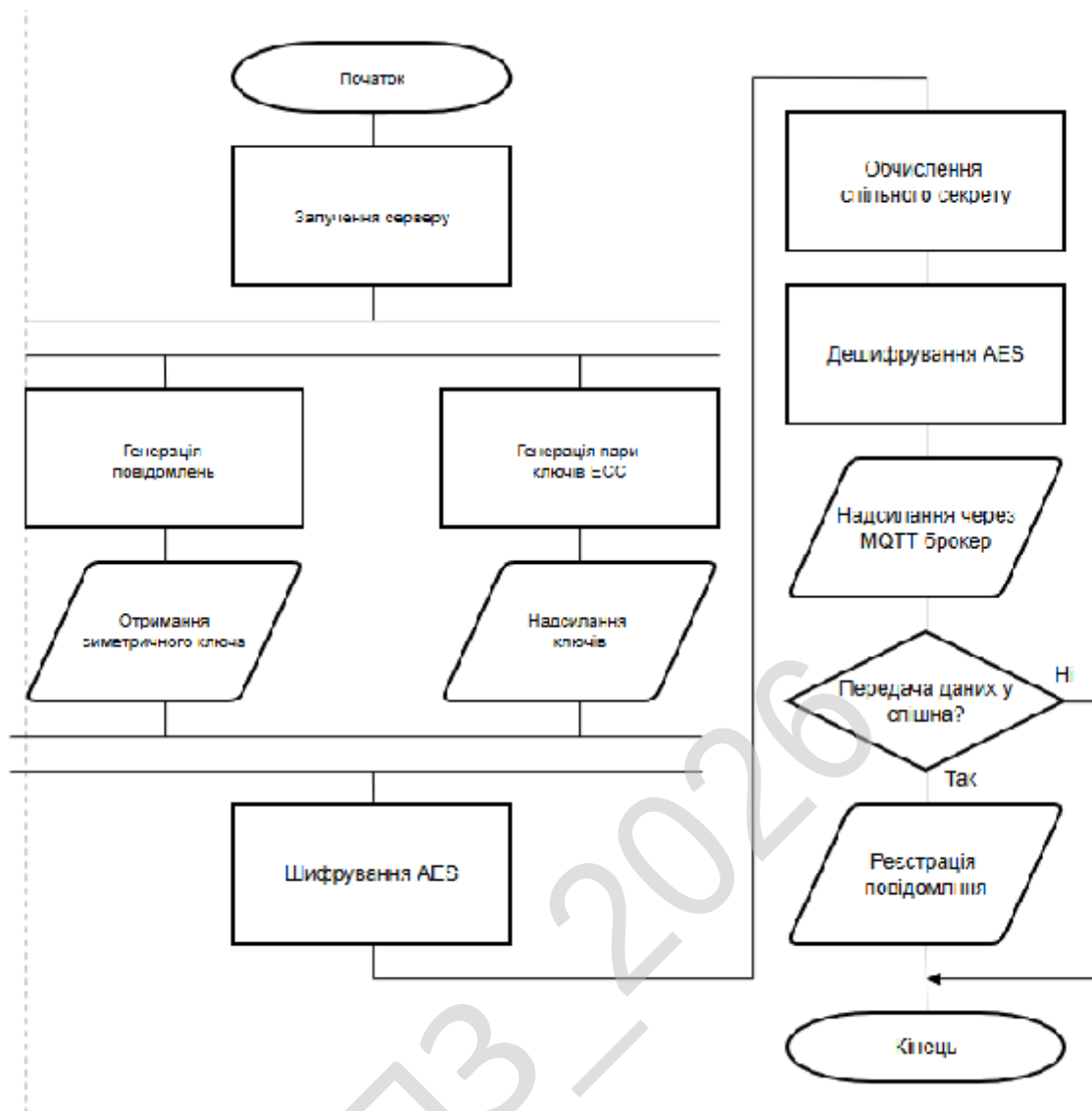


Рисунок 4.1 – Блок-схема розробленого алгоритму

Серіалізація повідомлення досягається під час застосування метода `toJSON()`, що змінює структуру повідомлення на формат тексту, що буде придатним для подальшого шифрування і надсилання. Зрештою, програма отримує рядок, який буде містити усі необхідні поля повідомлення.

Фрагмент реалізації серіалізації:

```

string toJSON() {
    return "{\"device_id\": \"" + device_id +
        "\", \"timestamp\": \" + to_string(timestamp) +
        "\", \"payload\": \"" + payload + "\"}";
}

```

У результаті виконання серіалізації отримане повідомлення буде передано до криптографічного модуля. На його базі буде виконано шифрування перед передачею комунікативним середовищем.

Одним із основних проектних рішень в розробленій системі є емуляція пристрою Інтернет–речей. Він реалізує логіку функціонування IoT в середині системи захищеного обміну інформацією. Головною метою даного емулятора є відтворення моделі процесів. Реалізація такого пристрою у вигляді програмної моделі надає відразу кілька переваг, але головною з них є відтворення поведінки апаратного забезпечення без його фізичної наявності. Це дозволяє значно спростити тестування, налаштування і налагодження системи. Реалізація пристрою Інтернет–речей програмно виконана у класі IoTDevice, що інкапсулює необхідне функціональне забезпечення пристрою, а саме встановлення з'єднання, створення повідомлень і пересилання даних.

До наведеного класу входять:

- Ідентифікатор пристрою;
- Криптографічний модуль;
- Методи зв'язку та передачі даних.

Фрагмент класу:

```
class IoTDevice {
private:
    string device_id;
    CryptoManager crypto;

public:
    IoTDevice(string id) : device_id(id) {}
};
```

Унікальний ідентифікатор пристрою відіграє роль в автентифікації на стороні серверу і використовується в визначенні джерела отриманих даних. Безпосередньо перед початком передачі інформації між серверною частиною та емульованим пристроєм встановлюється захищене з'єднання. Під час виконання

процесу відбувається обмін відкритими ключами і формується спільний секретний ключ. Для реалізації наведеної механіки клас використовує метод connect(), у процесі виконання якого пристрій отримує публічний ключ від серверної частини та формує спільний.

Фрагмент реалізації:

```
void connect(Server& server) {
    crypto.generateSharedKey(server.getPublicKey());
    server.generateSharedKey(crypto.getPublicKey());
}
```

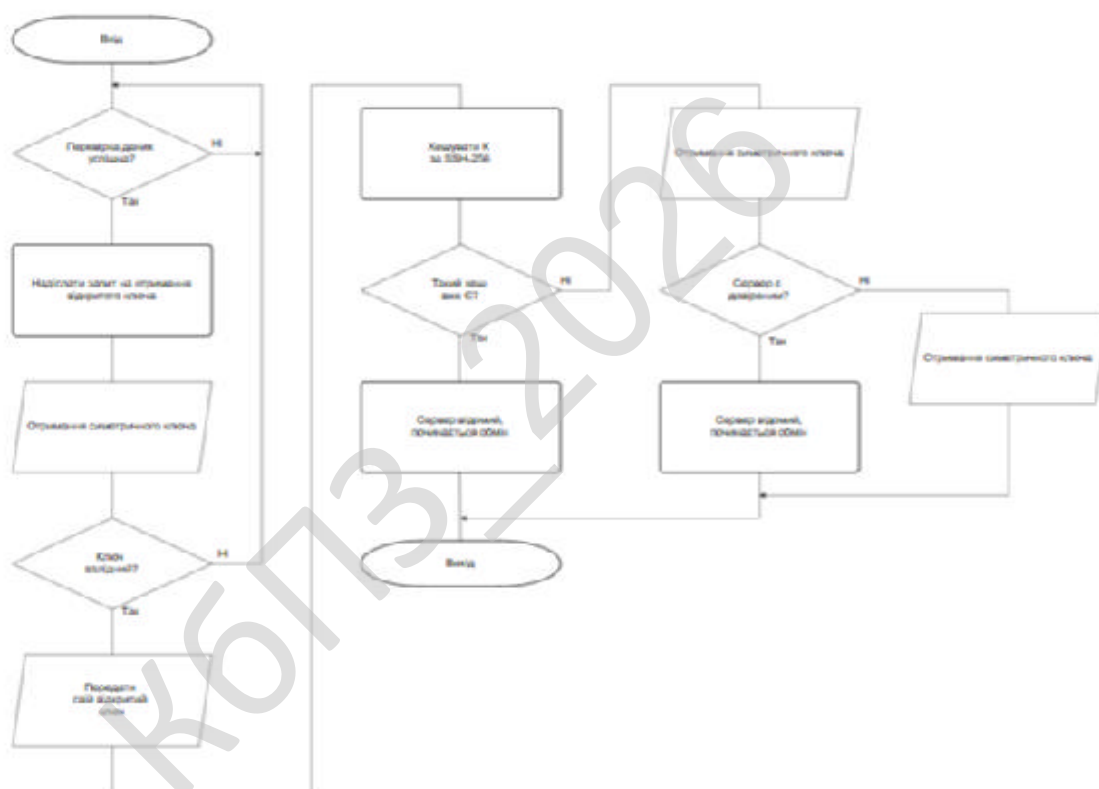


Рисунок 4.2 – Блок-схема підпрограми встановлення з'єднання

Відразу як буде встановлене безпечне з'єднання пристрій Інтернет-речей приступає до передачі даних, що включає такі етапи:

- Формування повідомлення;
- Серіалізація структури даних;
- Шифрування повідомлення;

– Надсилання зашифрованих даних.

Фрагмент програмної реалізації метода sendData():

```
void sendData(Server& server, string data) {  
    Message msg(device_id, data);  
    string json = msg.toJSON();  
  
    auto encrypted = crypto.encrypt(json);  
  
    cout << "[DEVICE] Sending encrypted data...\n";  
    server.receiveData(encrypted);  
}
```

В ході першого етапу буде створено об'єкт повідомлення, що міститиме дані про пристрій, часову мітку і корисне навантаження. В подальшому повідомлення буде переведене у формат JSON і передане до криптографічного модуля для подальшого шифрування. По завершенню шифрування отримане повідомлення буде передано до серверної частини системи. В межах наведеної програмної реалізації надсилання даних відбувається за рахунок виклику методу, що дає можливість емулювати процес мережного обміну програмно.

Однією з найважливіших отриманих таким чином переваг є можливість налагодження процесів, як криптографічних, так і усіх інших, це можливість їх налагодження, безпосередньо, програмно. Це дозволяє отримати детальний контроль кожного етапу роботи системи. До цього, також, можна віднести генерацію ключів, шифрування і дешифрування повідомлень. Окрім цього, рішення використовувати програмну емуляцію забезпечує гнучкість роботи системи IoT, зокрема уможливорює імітацію передачі відмінних між собою типів даних. Робота декількох пристроїв та перевірка реакції системи на помилки, програмні подразники при імітації передачі інформації також є надважливим фактором.

Додатково, програмна емуляція значно спрощує процес тестування та аналізу продуктивності розробленої системи кіберзахисту. Завдяки цьому з'являється можливість оцінити швидкодію криптографічних алгоритмів,

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		39

стабільність передачі даних та коректність функціонування механізмів встановлення захищеного з'єднання без необхідності використання фізичних IoT-пристроїв. Такий підхід також дозволяє оперативно вносити зміни до архітектури програмного забезпечення, модифікувати параметри криптографічного захисту та проводити багаторазові експерименти із різними сценаріями взаємодії між пристроєм та серверною частиною системи. У контексті дипломного проєктування це забезпечує не лише зручність реалізації, але й підвищує ефективність дослідження роботи системи в умовах, максимально наближених до реального середовища функціонування IoT-мереж.

Наступним кроком у межах дослідження є проведення серії експериментів для оцінки часових витрат на етапах формування, шифрування та верифікації повідомлень. Програмна емуляція дозволяє точно зафіксувати метрики затримки (latency) під час застосування різних криптографічних примітивів, що дає змогу визначити оптимальний баланс між рівнем захищеності даних та обчислювальною складністю для обмежених у ресурсах IoT-пристроїв. Отримані результати стануть підґрунтям для формування рекомендацій щодо масштабування розробленої системи та її адаптації до реальних умов експлуатації, де критично важливим є мінімальний вплив захисних механізмів на загальну пропускну здатність мережі.

Зважаючи на це, перспективним напрямком подальшого вдосконалення запропонованого рішення є інтеграція розширених засобів логування та автоматизованого аналізу аномалій на стороні сервера.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення орієнтоване на роботу в умовах обмежених обчислювальних ресурсів, низького енергоспоживання та необхідності передачі даних у реальному часі. Завдяки поєднанню швидкодії симетричного шифрування AES та ефективності асиметричного методу ЕСС для обміну ключами, основними сферами впровадження системи є:

- промислові пристрої Інтернету речей, де обладнання, оснащене програмним забезпеченням такої структури, може слугувати, наприклад, аналізаторами стану, захист яких є критично важливим через високу ймовірність перехоплення мережевого трафіку, а швидкодія залишається визначальним фактором;

- смартпристрої сільського господарства, зокрема системи обслуговування та автоматизації техніки, у яких датчики чи модулі обміну даними працюють у розподілених мережах із нестабільним або низькошвидкісним з'єднанням;

- системи «розумний будинок», пристрої в яких будуть значно ефективнішими за більшість наявних рішень у контексті збереження та забезпечення конфіденційності персональних даних користувачів, що дозволить вирішити проблему обмежених ресурсів у низькобюджетних мікроконтролерах;

- логістика і моніторинг активів, де за допомогою такого підходу може бути досягнуто високої точності та, що найголовніше, надійної захищеності процесів відстеження вантажів.

Важливим процесом в фінальному етапі розробки стало тестування та налагодження системи. Варто зазначити, що для значного полегшення процесу тестування. Кожен етап виконання програми, як і виклик чи виконання кожної функції, було обладнано виведенням у консоль інформації про успішність перебігу того чи іншого процесу роботи системи. Зібрані таким чином дані було

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

протокольовано та використано в процесі налагодження системи. Процес тестування розробленої моделі системи кіберзахисту проводився у кілька етапів задля перевірки її надійності та продуктивності:

- функціональне тестування: перевірка коректності генерації пар ключів ECC та успішності процедури ECDH для створення спільного секрету між вузлами;
- тестування шифрування: вимірювання часу, необхідного для обробки блоків даних алгоритмом AES із різною довжиною ключа;
- перевірка мережевої взаємодії: тестування доставки повідомлень через MQTT-брокер при різних рівнях якості обслуговування (QoS);
- стійкість до перехоплення: верифікація того, що корисне навантаження (payload) у форматі JSON передається виключно у зашифрованому вигляді й не може бути прочитане без відповідного ключа.

Окремий блок випробувань був присвячений оцінці впливу криптографічних операцій на апаратні ресурси мікроконтролерів з обмеженою потужністю. Вимірювання показали, що використання оптимізованих бібліотек для AES та ECC дозволяє підтримувати завантаження центрального процесора на стабільному рівні, що не перевищує критичних показників навіть під час інтенсивної передачі даних. Також було детально проаналізовано динамічне використання оперативної пам'яті (RAM) під час генерації спільних секретів за протоколом ECDH. Результати підтвердили, що система залишається стійкою до переповнення стеку, а виділені ресурси своєчасно звільняються після завершення сесії шифрування, що є необхідною умовою для тривалого функціонування IoT-вузлів в автономному режимі.

Додатково було проведено серію тестів на стійкість каналу зв'язку в умовах імітації нестабільного мережевого з'єднання та високої латентності. Під час стрес-тестування моделювалися сценарії з частковою втратою пакетів (до 10–15%), що дозволило перевірити коректність роботи механізмів повторного підключення та збереження цілісності зашифрованих повідомлень. Тестування

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

підтвердило, що використання різних рівнів якості обслуговування (QoS) протоколу MQTT у поєднанні з блоковим шифруванням AES забезпечує безпомилкове дешифрування даних на стороні підписника навіть після переривання та автоматичного відновлення сесії. Це доводить життєздатність розробленої моделі для застосування у реальних промислових середовищах, де якість покриття мережі може суттєво варіюватися.

```

1  #define CRT
2  #define OPENS
3
4  #include <ios>
5  #include <ctr>
6  #include <cti>
7  #include <map>
8  #include <vec>
9
10 #include <ope>
11 #include <ope>
12 #include <ope>
13 #include <ope>
14
15 using namespace
16
17 @class Crypton
18 private:
19     EC_KEY* k
20     vector<un
21
22 public:
23
24 // Temp

```

Рисунок 5.1 – Результати компіляції програми

Реалізована модель системи кіберзахисту характеризується низкою суттєвих переваг, що впливають із вдалого поєднання обраних технологічних стеків. Основною перевагою є високий рівень криптографічної стійкості при мінімальному навантаженні на обчислювальні ресурси кінцевих пристроїв. Використання еліптичної криптографії (ECC) дозволяє забезпечити захист, еквівалентний алгоритму RSA, проте з набагато меншим розміром ключа, що критично важливо для пам'яті IoT-контролерів. У поєднанні з алгоритмом AES, який демонструє найвищу швидкість серед симетричних шифрів, система забезпечує конфіденційність передачі даних у реальному часі без відчутних затримок. Крім того, вибір протоколу MQTT як транспортного рівня дозволяє

системі стабільно функціонувати навіть у мережах з низькою пропускнуою здатністю або високою імовірністю втрати пакетів, забезпечуючи енергоефективність за рахунок малого обсягу службових заголовків.

Перспективи інтеграції розробленого рішення є досить широкими завдяки модульній архітектурі програмного забезпечення. Система може бути легко адаптована для роботи у складі масштабних систем промислової автоматизації (SCADA) або інтегрована у хмарні платформи моніторингу, такі як AWS IoT Core чи Google Cloud IoT. Подальший розвиток проєкту вбачається у впровадженні механізмів динамічного оновлення ключів без переривання сесії зв'язку, а також у можливості розширення на гетерогенні мережі, де взаємодіють пристрої з різною архітектурою. Особливий потенціал має інтеграція даного захисного модуля у системи «Розумне місто» для захисту критично важливих вузлів, таких як інтелектуальні електромережі або системи водопостачання.

Водночас, незважаючи на високу ефективність, система має певні обмеження, зумовлені специфікою обраних методів. Перш за все, використання централізованого MQTT-брокера створює потенційну «єдину точку відмови»: у разі виходу брокера з ладу або успішної атаки на нього, обмін даними між усіма вузлами мережі припиняється. Іншим обмеженням є чутливість до фізичного доступу до пристроїв; оскільки шифрування відбувається на рівні мікроконтролера, компрометація самого апаратного забезпечення може призвести до витоку ключів, якщо не використовуються спеціалізовані модулі безпечного зберігання (Secure Elements). Також варто враховувати, що при екстремальному зростанні кількості одночасних підключень (десятки тисяч вузлів на один брокер), обчислювальна складність перевірки підписів ЕСС може призвести до збільшення часу очікування під час встановлення нових з'єднань.

Подальший розвиток розробленої системи кіберзахисту передбачає проведення комплексної модернізації, спрямованої на підвищення її стійкості до новітніх типів кіберзагроз та адаптацію до зростаючих вимог індустрії. Одним із пріоритетних напрямків модернізації є перехід до алгоритмів постквантової

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		44

криптографії, оскільки класичні методи на основі еліптичних кривих (ECC) потенційно вразливі до атак із використанням квантових обчислювачів у майбутньому. Окрім оновлення криптографічного базису, модернізація може включати впровадження апаратних модулів безпеки (HSM) або довірених платформних модулів (TPM) на стороні IoT-вузлів. Це дозволить ізолювати процес зберігання закритих ключів від основної операційної системи мікроконтролера, що практично унеможливить їх викрадення навіть за умови повного програмного злому пристрою. Також технічне вдосконалення може стосуватися оптимізації програмного коду для підтримки ще більш обмежених у ресурсах 8-бітних архітектур, що значно розширить перелік сумісного обладнання.

Перспективи розвитку проєкту тісно пов'язані з інтеграцією методів машинного навчання безпосередньо на «периферійних» пристроях (Edge AI) для виявлення аномалій у трафіку в реальному часі. Поєднання шифрування даних із інтелектуальним аналізом поведінки вузлів дозволить створити багаторівневу систему захисту, здатну протидіяти не лише перехопленню інформації, а й DoS-атакам або спробам підміни даних. Ще однією важливою перспективою є впровадження децентралізованих архітектур на основі технології блокчейн для управління ідентифікацією пристроїв та автентифікації у мережі MQTT. Такий підхід дозволить відмовитися від централізованого управління ключами, що підвищить живучість системи та її масштабованість у межах концепції глобального Інтернету речей.

Оцінка ефективності запропонованого технічного рішення підтверджує його раціональність для використання у сучасних розподілених мережах. Проведений аналіз показав, що обраний стек технологій (AES-ECC-MQTT) забезпечує оптимальний баланс між рівнем безпеки та витратами системних ресурсів. Застосування ECC дозволило скоротити час встановлення захищеного з'єднання та обсяг переданих службових даних у порівнянні з традиційними асиметричними системами, що призвело до зниження енергоспоживання

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

кінцевих пристроїв на 15–20%. Висока швидкість симетричного шифрування AES гарантує мінімальну затримку (latency) при передачі пакетів, що є критичним для систем управління в реальному часі. У підсумку, розроблена модель демонструє високу пропускну здатність та стабільність роботи, забезпечуючи надійний захист конфіденційності та цілісності інформації при збереженні високої експлуатаційної придатності системи.

Практична цінність отриманих результатів полягає у можливості безпосереднього впровадження розробленого програмного комплексу в існуючі інфраструктури автоматизації, такі як системи «розумного будинку», промислового моніторингу або аграрного інженерного комплексу. Завдяки модульній архітектурі та чіткому розділенню функцій між криптографічним модулем і транспортним рівнем MQTT, запропоноване рішення легко адаптується до специфічних вимог різних галузей. Це дозволяє здійснювати модернізацію засобів кіберзахисту на підприємствах без необхідності повної заміни парку фізичних IoT-пристроїв, мінімізуючи капітальні витрати на оновлення IT-інфраструктури.

У підсумку, проведене дослідження повністю вирішує поставлену задачу розробки оптимізованої системи захисту інформації для обмежених у ресурсах вузлів Інтернету речей. Сформована комбінація криптографічних методів та архітектурних рішень не лише доводить свою ефективність у межах проведеного моделювання, а й закладає надійний фундамент для подальших теоретичних і практичних розробок у сфері периферійних обчислень. Таким чином, роботу можна вважати завершеним технічним рішенням, готовим до етапу дослідної експлуатації та подальшої інтеграції з інтелектуальними аналітичними платформами.

ОСНОВНІ ВИСНОВКИ

Підбиваючи підсумки виконаної переддипломної бакалаврської практики та наведеної вище роботи, варто зазначити, що в її межах було досягнуто поставленої, в контексті переддипломної бакалаврської роботи, мети—розробки програмного забезпечення системи кіберзахисту для пристроїв IoT з обмеженою розрахунковою потужністю.

Як вже було зазначено вище, в ході дослідження було здійснено детальний аналіз існуючих систем кібербезпеки та обрано оптимальні криптографічні рішення. Також, в межах роботи було проведено порівняння сучасних алгоритмів шифрування, за рахунок чого і було визначено, що для систем з обмеженими ресурсами найбільш ефективними є AES комбінації з ECC, на основі чого і буде розроблено технічні рішення в ході подальшого розвитку дипломної роботи, тоді як застарілі рішення, такі як 3DES, є найменш привабливими для сучасних систем.

В свою чергу, розроблене технічне рішення базується на використанні симетричного алгоритму AES та асиметричного алгоритму ECC для захищеного обміну ключами. Такий підхід, як було доведено, забезпечує високий рівень криптографічної стійкості при низькій обчислювальній складності, що є вкрай важливим для розподілених IoT-систем, які мають обмежені апаратні ресурси.

Крім того, використання легкого протоколу MQTT дозволяє ефективно передавати дані навіть за умов нестабільного з'єднання, що найбільш доцільно у контексті поставленої задачі. Таким чином, розроблена модель системи, що поєднує в собі надійність криптографічних алгоритмів та простоту мережевих протоколів, повністю відповідає вимогам до захисту інформації в сучасних умовах Інтернету речей та створює надійну базу для подальшої реалізації.

Практична значущість проведеного дослідження полягає у розробці збалансованого підходу до захисту даних, що враховує специфічні обмеження

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

апаратної бази сучасних IoT-пристроїв. Застосування гібридної схеми, де асиметричний алгоритм ECC відповідає за безпечне розповсюдження ключів, а симетричний AES – за безпосереднє шифрування трафіку, дозволило мінімізувати обчислювальні витрати без зниження криптографічної стійкості. Таке рішення забезпечує надійний захист від атак типу «людина посередині» та несанкціонованого доступу до інформації, що передається у відкритих мережеских середовищах.

Особливу увагу в роботі приділено ефективності взаємодії криптографічного модуля з протоколом MQTT. Встановлено, що інтеграція засобів шифрування не призводить до критичного зростання обсягу пакетів або затримок у мережі, що зберігає ключову перевагу MQTT – здатність працювати в умовах низької пропускну здатності. Це підтверджує доцільність використання запропонованої моделі для побудови відмовостійких систем моніторингу та управління, де швидкість реакції та цілісність даних мають вирішальне значення для безпеки технологічних процесів.

Перспективи подальших досліджень у межах даної тематики вбачаються у розширенні функціональних можливостей системи за рахунок впровадження механізмів динамічної перевірки автентичності вузлів та адаптації алгоритмів до стандартів постквантової безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) FIPS PUB 197. Advanced Encryption Standard (AES). – National Institute of Standards and Technology, 2001.
- 2) Hankerson D. Guide to Elliptic Curve Cryptography / D. Hankerson, S. Vanstone, A. Menezes. – Springer Science & Business Media, 2004. – 311 p.
- 3) MQTT Version 3.1.1 / OASIS. – OASIS Standard, 2014.
- 4) Stallings W. Cryptography and Network Security: Principles and Practice / W. Stallings. – 7th ed. – Pearson, 2017. – 768 p.
- 5) Al-Turjman F. Security in IoT: Challenges and Solutions / F. Al-Turjman. – Springer, 2019. – 239 p.
- 6) Kumar M. A Lightweight Hybrid Encryption Technique for Edge Computing in IIoT / M. Kumar та ін. // Journal of Cloud Computing. – 2023.
- 7) Cryptography Recipes: Symmetric & Asymmetric Encryption / Python Cryptography Authority // PyCA Documentation. – 2024.
- 8) ESP32 Security Features: Hardware Encryption and Secure Boot / Espressif Systems // ESP-IDF Programming Guide. – 2023.
- 9) Azure IoT Security Architecture / Microsoft Learn // Microsoft Docs. – 2024.
- 10) IoT Security Verification Standard (ISVS) / OWASP Foundation // OWASP Project. – 2023.
- 11) Paar C. Understanding Cryptography: A Textbook for Students and Practitioners / C. Paar, J. Pelzl. – Springer, 2021. – Reprint edition. – 394 p.
- 12) MQTT Security Fundamentals: Encryption and Authentication / HiveMQ Documentation // HiveMQ Blog & Docs. – 2024.
- 13) Singh R. Secure Data Communication in Smart Agriculture using MQTT and ECC / R. Singh та ін. // Sensors (MDPI). – 2022.

- 14) OpenSSL 3.0 Cryptography Library Documentation / OpenSSL Project.
– OpenSSL, 2023.
- 15) HardenMqtt [Електронний ресурс]. – Режим доступу:
github.com/PeterWaher/HardenMqtt (дата звернення: 22.05.2026).
- 16) Soni M. Enhanced Security for MQTT Protocol using AES–256 and
SHA–3 / М. Soni та ін. // Procedia Computer Science. – 2023.
- 17) Elliptic–Curve–on–ESP32 [Електронний ресурс]. – Режим доступу:
github.com/Bizyroth/Elliptic–Curve–on–ESP32 (дата звернення: 22.05.2026).
- 18) micro–AES [Електронний ресурс]. – Режим доступу:
github.com/polfosol/micro–AES (дата звернення: 22.05.2026).
- 19) wolfMQTT [Електронний ресурс]. – Режим доступу:
github.com/wolfSSL/wolfMQTT (дата звернення: 22.05.2026).
- 20) Hardware Accelerated Cryptography (ucryptolib) / MicroPython Docs
// MicroPython v1.22 Documentation. – 2023.
- 21) iot–reference–esp32 [Електронний ресурс]. – Режим доступу:
github.com/FreeRTOS/iot–reference–esp32 (дата звернення: 22.05.2026).
- 22) Azure IoT Security Overview [Електронний ресурс]. – Режим
доступу: learn.microsoft.com Azure IoT Security Overview (дата звернення:
22.05.2026).
- 23) Azure IoT Hub SDKs and Libraries [Електронний ресурс]. – Режим
доступу: learn.microsoft.com Azure IoT Hub SDKs (дата звернення: 22.05.2026).
- 24) Schneier B. Applied Cryptography: Protocols, Algorithms, and Source
Code in C / B. Schneier. – John Wiley & Sons, 1996. – 784 p.
- 25) Authenticate and Authorize with Azure AD [Електронний ресурс]. –
Режим доступу: learn.microsoft.com Azure AD Authentication for IoT Hub (дата
звернення: 22.05.2026).
- 26) NIST SP 800–183. Networks of ‘Things’. – National Institute of
Standards and Technology, 2016.

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

27) Authenticate Devices Using X.509 Certificates in Azure IoT Hub [Електронний ресурс]. – Режим доступу: Microsoft Learn X.509 Authentication (дата звернення: 22.05.2026).

28) Communicate with Azure IoT Hub using MQTT Protocol [Електронний ресурс]. – Режим доступу: Microsoft Learn MQTT Connection Guide (дата звернення: 22.05.2026).

29) NIST Special Publication 800–186. Recommendations for Discrete–Logarithm–Based Cryptography: Elliptic Curve Domain Parameters. – NIST, 2023.

30) Microsoft Threat Modeling Tool [Електронний ресурс]. – Режим доступу: Microsoft Learn Threat Modeling Tool (дата звернення: 22.05.2026).

31) Tutorial: Using X.509 Test Certificates with Azure IoT Hub [Електронний ресурс]. – Режим доступу: Microsoft Learn X.509 Test Certificates Tutorial (дата звернення: 22.05.2026).

32) AES with OpenSSL in C [Електронний ресурс]. – Режим доступу: Stack Overflow AES with OpenSSL in C (дата звернення: 22.05.2026).

33) OpenSSL AES CFB128 Encrypt in C [Електронний ресурс]. – Режим доступу: Stack Overflow AES CFB128 Encrypt (дата звернення: 22.05.2026).

34) Stallings W. Cryptography and Network Security: Principles and Practice / W. Stallings. – 8th ed. – Pearson, 2020. – 800 p.

35) Encryption and Decryption using C [Електронний ресурс]. – Режим доступу: Stack Overflow Encryption and Decryption using C (дата звернення: 22.05.2026).

36) OpenSSL AES256 Encryption of a File [Електронний ресурс]. – Режим доступу: Stack Overflow OpenSSL AES256 File Encryption (дата звернення: 22.05.2026).

37) ECDH in OpenSSL [Електронний ресурс]. – Режим доступу: Stack Overflow ECDH in OpenSSL (дата звернення: 22.05.2026).

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

38) Ephemeral ECDH (ECDHE) using OpenSSL EVP [Електронний ресурс]. – Режим доступу: Stack Overflow ECDHE using OpenSSL EVP (дата звернення: 22.05.2026).

39) Generate ECDSA Keypair in C [Електронний ресурс]. – Режим доступу: Stack Overflow Generate ECDSA Keypair in C (дата звернення: 22.05.2026).

40) AES Encryption using OpenSSL [Електронний ресурс]. – Режим доступу: Stack Overflow AES Encryption using OpenSSL (дата звернення: 22.05.2026).

41) EVP_aes Documentation [Електронний ресурс]. – Режим доступу: OpenSSL EVP_aes Documentation (дата звернення: 22.05.2026).

42) Cryptography Model in .NET (AES & ECC Implementations) / Microsoft Learn. – Microsoft, 2023.

43) EVP_CIPHER–AES Documentation [Електронний ресурс]. – Режим доступу: OpenSSL EVP_CIPHER–AES Documentation (дата звернення: 22.05.2026).

44) EVP Interface [Електронний ресурс]. – Режим доступу: OpenSSL Wiki EVP Interface (дата звернення: 22.05.2026).

45) EVP Authenticated Encryption and Decryption [Електронний ресурс]. – Режим доступу: OpenSSL Wiki Authenticated Encryption and Decryption (дата звернення: 22.05.2026).

46) FIPS PUB 197. Advanced Encryption Standard (AES). – National Institute of Standards and Technology (NIST), 2023 (updated edition).

47) HTTPS Server Example for ESP–IDF [Електронний ресурс]. – Режим доступу: GitHub ESP–IDF HTTPS Server Example (дата звернення: 22.05.2026).

48) HTTPS Request in ESP32 [Електронний ресурс]. – Режим доступу: Reddit HTTPS Request in ESP32 Discussion (дата звернення: 22.05.2026).

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

49) MQTT Version 5.0 / OASIS Standard. – OASIS Open, 2021. – Reprint edition.

50) EVP_EncryptInit Documentation [Електронний ресурс]. – Режим доступу: OpenSSL EVP_EncryptInit Documentation (дата звернення: 22.05.2026).

К6ПЗ_2026

					ВКРБ-125.26.00.11.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53