БАКАЛАВРСЬКА РОБОТА

БР.КІ – 00.00.00.000 ПЗ

Група КІ-22-1

Фридрик Марія

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра комп'ютерних систем і мереж

Фридрик Марія Ігорівна

УДК 004.4

БАКАЛАВРСЬКА РОБОТА

Розробка алгоритмічно-програмних рішень шифрування даних для пристроїв IoT з обмеженими обчислювальними ресурсами

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня *Фридрик М.І.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник *доц. Топалов А.М*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,проф. _____ *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Інститут *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КСМ.
(С.І. Мельничук)
«_25_» _04_ 2026 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Фридрик Марія Ігорівна

(прізвище, ім'я, по батькові)

1. **Тема проекту (роботи)** Розробка алгоритмічно-програмних рішень шифрування даних для пристроїв IoT з обмеженими обчислювальними ресурсами

керівник проекту (роботи) доц. Топалов А.М

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від *від 25.05.2025 №273/7*

2. Строк подання студентом роботи *11 червня 2026 р.*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області та вимог до системи*

2. Моделювання та вибір методів обробки даних

3. Проектування та реалізація системи

4. Тестування, оцінка ефективності та висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

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

7. Дата видачі завдання *20 лютого 2026 р* .

КАЛЕНДАРНИЙ ПЛАН

№ з/п	**Назва етапів дипломного проекту (роботи)**	**Строк виконання етапів проекту (роботи)**	**Примітка**
1	*Постановка задачі та вибір технології для розробки.*	*Лютий, 2026*	
2	*Огляд наявних аналогів. Збір інформації про інформаційне та програмне забезпечення.*	*Березень, 2026*	
3	*Розробка програмного забезпечення*	*Квітень, 2026*	
4	*Тестування програмного забезпечення*	*Травень, 2026*	
5	*Оформлення пояснювальної записки*	*Червень, 2026*	

Студент _____ *Фридрик М.І.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Топалов А.М.*
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

У сучасному світі стрімкий розвиток інформаційних технологій та концепції Інтернету речей (IoT) призводить до масового впровадження розумних пристроїв у різних сферах життєдіяльності людини — від побутових систем автоматизації до промислових і критично важливих інфраструктур. IoT-пристрої активно використовуються для збору, обробки та передачі даних у реальному часі, що значно підвищує ефективність управління процесами. Водночас зростання кількості підключених пристроїв створює нові виклики у сфері кібербезпеки, зокрема щодо забезпечення конфіденційності, цілісності та автентичності переданих даних.

Актуальність теми зумовлена необхідністю розробки ефективних алгоритмічно-програмних рішень шифрування для IoT-пристроїв, які мають обмежені обчислювальні ресурси, енергоспоживання та пам’ять. Традиційні криптографічні алгоритми, що широко застосовуються у потужних обчислювальних системах, часто є надто ресурсоємними для використання у вбудованих пристроях. Це створює потребу у створенні оптимізованих, легковагових (lightweight) криптографічних методів, які забезпечують належний рівень безпеки без значного навантаження на апаратні ресурси.

Існуючі рішення у сфері безпеки IoT, як правило, або не враховують обмеження пристроїв, або не забезпечують достатнього рівня захисту даних. Це підкреслює необхідність комплексного підходу до розробки алгоритмів шифрування, що поєднують ефективність, швидкодію та низьке енергоспоживання. Особливої уваги потребує адаптація криптографічних алгоритмів до умов обмежених ресурсів, що є ключовим фактором при проєктуванні сучасних IoT-систем.

Метою даної роботи є розробка алгоритмічно-програмних рішень шифрування даних для IoT-пристроїв з обмеженими обчислювальними ресурсами, які забезпечують високий рівень безпеки при мінімальних витратах ресурсів.

Для досягнення поставленої мети необхідно вирішити такі завдання: провести аналіз предметної області та існуючих підходів до забезпечення безпеки в IoT-системах; дослідити сучасні криптографічні алгоритми та оцінити їх придатність для використання в умовах обмежених ресурсів; визначити критерії ефективності алгоритмів шифрування; розробити та оптимізувати алгоритмічні рішення для IoT-пристроїв; реалізувати програмний модуль шифрування; провести експериментальні дослідження та оцінити ефективність запропонованих рішень.

Об’єктом дослідження є процеси забезпечення інформаційної безпеки в IoT-пристроях з обмеженими обчислювальними ресурсами.

Предметом дослідження є алгоритмічні та програмні методи шифрування даних, адаптовані до умов обмежених ресурсів IoT-пристроїв.

Методи дослідження включають аналіз наукових джерел і сучасних підходів до криптографії, порівняльний аналіз алгоритмів шифрування, моделювання та оптимізацію алгоритмів, експериментальне дослідження продуктивності та енергоспоживання, а також оцінку ефективності запропонованих рішень.

У результаті виконання роботи буде розроблено програмний модуль шифрування, орієнтований на використання в IoT-пристроях, який забезпечує баланс між рівнем безпеки та ефективністю використання ресурсів. Особлива увага приділятиметься зниженню обчислювальної складності алгоритмів, мінімізації енергоспоживання та забезпеченню надійного захисту даних.

Бакалаврська робота містить 67 сторінок, 26 рисунків, 6 таблиць, 3 розділи, висновки та список використаних джерел із 40 найменувань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО СИСТЕМИ БЕЗПЕКИ IoT-ПРИСТРОЇВ

1.1 Основи інформаційної безпеки в IoT-системах

З розвитком технологій Інтернету речей (IoT) дедалі більше вбудованих пристроїв, датчиків та інших фізичних об'єктів підключаються до Інтернету, роблячи доступною інформацію, яку вони збирають, передають і зберігають, для підписників послуг IoT, аналітичних компаній або муніципалітетів. Вбудовані системи та системи IoT стають повсюдними і розгортаються з використанням різних технологій зв'язку, таких як Zigbee, LoRaWAN, Sigfox, WiFi, 3G/LTE, у різних галузях, зокрема системах домашньої автоматизації, охороні здоров'я, автомобільній галузі, промислових установках, муніципальних послугах, критичній інфраструктурі, приватному та громадському просторі [1].

Зі зростанням популярності IoT також виникають проблеми для провайдерів та розгортальників послуг IoT, пов'язані з сумісністю з різними IoT-платформами та інтеграцією пристроїв із різними технічними характеристиками й від різних виробників в єдиний додаток або систему. Ця проблема стає очевидною, щойно замовник починає приділяти увагу питанням безпеки. Використання системи IoT має характеризуватися не лише ефективністю передачі даних, характеристиками пристроїв і протоколом зв'язку, а й захистом даних, оскільки в більшості випадків витік персональних даних може скомпрометувати всю систему IoT.

У цьому контексті забезпечення конфіденційності, цілісності та доступності даних, що передаються в IoT, залишається постійним викликом, оскільки фундаментальні принципи інформаційної безпеки завжди були ключовим питанням під час розгортання реальних застосунків. Хоча багато досліджень зосереджено на покращенні безпеки систем IoT, постійне зростання кількості атак свідчить про те, що ще потрібно виконати значну роботу. Це зумовлено поєднанням факторів: починаючи від ризиків, пов'язаних з IoT, які

часто є невідомими або важко ідентифікованими, і закінчуючи ефективністю заходів безпеки, яку зазвичай складно оцінити.

Крім цих притаманних викликів, значна кількість пристроїв IoT продовжує розгортатися без достатнього рівня безпеки, що піддає їх ризикам [2]. Найзручнішим способом забезпечення конфіденційності та цілісності є використання шифрування даних, які передаються між пристроями IoT.

Нещодавнє опитування показало, що більшість компаній стурбовані безпекою IoT, проте 24 % компаній ще більше стурбовані функціональністю продукту [3]. Понад 50 % компаній активно впроваджують стратегію безпеки IoT на своїх пристроях і процесах, а 85 % компаній використовують шифрування для передачі даних від кінцевого пристрою до шлюзу.

Вибір алгоритму шифрування має здійснюватися обережно, оскільки використання алгоритмів із довгими ключами, хоча й підвищує рівень безпеки, водночас є обчислювально складним і зменшує термін служби пристрою. Традиційні алгоритми шифрування вимагають значних ресурсів, тоді як вбудовані пристрої обмежені своєю обчислювальною потужністю, обсягом пам’яті та ємністю акумулятора. Тому розробники пристроїв IoT стикаються з проблемою балансування між рівнем безпеки та енергетичними можливостями [4].

Через обмежену енергію та обчислювальну потужність пристроїв IoT не всі алгоритми шифрування, що використовуються в комп’ютерних мережах, підходять для IoT. Для вирішення цієї проблеми були спеціально розроблені легковагові блочні шифри (lightweight block ciphers), призначені для роботи на пристроях з обмеженими ресурсами. Такі алгоритми зазвичай характеризуються меншим розміром блоку, меншим розміром ключа, нижчим споживанням пам’яті (завдяки мінімальним накладним витратам на шифрування/дешифрування) та меншою кількістю раундів виконання.

Обмежені енергетичні можливості низькоресурсних пристроїв є найкритичнішим викликом [5]. Проблеми з енергоспоживанням суттєво впливають на термін служби екосистеми IoT, що, у свою чергу, впливає на

термін служби елементів і процесів IoT, а також на оптимізацію політики управління в компанії.

Оцінка терміну служби пристрою може допомогти передбачити середній час між відмовами (Mean Time Between Failures — MTBF) та живучість ширшої екосистеми IoT, а також визначити терміни технічного обслуговування.

У цьому дослідженні ми пропонуємо модель для оцінки терміну служби пристрою з урахуванням захищеної передачі даних та аналізуємо десять легковагових алгоритмів шифрування, які рекомендовано використовувати в IoT. Ми покажемо, як тип шифрування даних і довжина даних впливають на термін служби та енергоспоживання пристрою.

1.2 Теоретичні засади криптографії для ресурсно-обмежених пристроїв

Значний успіх та широке розгортання технологій Інтернету речей (IoT) останніми роками створюють дедалі більші виклики в сфері безпеки. Пристрої IoT використовуються в багатьох сферах послуг і галузях промисловості, як у персональних, так і в комерційних застосунках. При цьому характер загроз відрізняється залежно від домену та сценарію впровадження. Типовий рівень безпеки технологій IoT залишається досить низьким, попри їхню повсюдність і популярність.

Безпека IoT. Ще у 2015 році компанія Symantec провела дослідження безпеки 50 пристроїв IoT і дійшла висновку, що 19 % перевірених пристроїв здійснюють обмін даними без шифрування (наприклад, без використання Transport Layer Security — TLS), а жоден із пристроїв не забезпечував взаємну автентифікацію [6]. Інше дослідження проаналізувало порушення безпеки 21 розумного пристрою і виявило, що лише 9 із них мали будь-які механізми захисту, тоді як решта легко піддавалися зламу через слабкі механізми безпеки, зокрема шифрування.

Аналогічно протестували рівень безпеки 28 пристроїв і встановили, що

39 % із них не використовували TLS для зв’язку.

Більшість компаній забезпечують захист IoT не як частину загальної бізнес-стратегії, а лише на рівні окремих бізнес-підрозділів. Слабка реалізація принципу «security by design» та обмежений контроль над технологіями в підключених пристроях є прямим наслідком такої стратегії і призводять до зростання кількості кібератак на Інтернет речей. За даними, у період з 2015 по 2018 рік 20 % організацій зазнали атак на свої системи IoT [7].

Згідно з дослідженням 2020 року, лише 60 % порушень безпеки виявляються та блокуються засобами кіберзахисту, тоді як інші 40 % інцидентів залишаються «прихованими» і походять саме з екосистеми IoT [8].

Тому забезпечення безпеки є важливим завданням на всіх рівнях системи. При правильній реалізації заходів кіберзахисту ймовірність успішного зламу залишається мінімальною, особливо з урахуванням того, що мільярди користувачів щодня користуються системами IoT [9].

Моделі зв’язку IoT та безпека. Шифрування залишається найефективнішим способом захисту даних під час передачі. Воно може застосовуватися на будь-якій ділянці шляху передачі даних: device-to-device, device-to-gateway, device-to-cloud або gateway-to-cloud. Вибір алгоритму шифрування залежить від архітектури та моделі зв’язку системи IoT. За визначенням, в середовищі IoT усі пристрої повинні бути підключені до Інтернету, для чого необхідно забезпечити інтерфейси для підключення «речей» до зовнішнього світу.

Залежно від рівня та можливостей зв’язку не всі «речі» є рівноцінними. Сфера пристроїв IoT є надзвичайно гетерогенною через різноманіття апаратного забезпечення, операційних систем і протоколів зв’язку між кінцевими пристроями та сервісами.

Більшість циклів життя пристроїв IoT свідчать про те, що пристрої виробляються та розгортаються в різних локаціях по всьому світу. Деякі «речі» мають прямий доступ до Інтернету та можуть нативно реалізовувати криптографічні механізми, тоді як для інших, що обмежені обчислювальними

ресурсами та енергоспоживанням, впровадження шифрування може бути складним завданням.

Типи пристроїв представлено на рис. 1.1. Вони поділяються на потужні (resourceful), обмежені в ресурсах (resource-constrained) та безресурсні (resourceless) пристрої.

Рисунок 1.1 Класифікація кінцевих пристроїв IoT.

Потужні кінцеві пристрої мають необмежене джерело живлення, вони автономні та не потребують додаткових елементів інфраструктури для роботи. Такі пристрої можуть одночасно виконувати роль датчиків, шлюзу, HTTP-сервера, сховища даних, здійснювати fog-обчислення та працювати як платформа Web of Things з усіма наданими нею сервісами.

На відміну від потужних пристроїв, обмежені в ресурсах «речі» мають обмежені можливості живлення та обчислювальної потужності, але можуть працювати як датчики, виконувати базові обчислення та обмінюватися даними з іншими сутностями (шлюзом, хмарою чи веб-серверами).

Безресурсні кінцеві пристрої — це пасивні об’єкти, які можуть бути виявлені за допомогою унікальних ідентифікаторів, таких як RFID-мітки та QR-коди. Вони не можуть виконувати обчислення, не мають сховища чи пам’яті, а лише відображають ідентифікатор або набір символів, для роботи з якими потрібен додатковий пристрій і програмне забезпечення. У деяких випадках

їхні можливості можуть бути розширені за рахунок програмного забезпечення, розгорнутого в хмарі або на локальному шлюзі.

Ця робота зосереджується саме на обмежених у ресурсах об’єктах, оскільки ця категорія становить найбільший ризик — пристрої, які мають достатній рівень обчислювальної потужності, щоб становити інтерес для зловмисника, але не мають достатніх ресурсів для впровадження повного набору захисних механізмів.

Ми дослідимо, які алгоритми шифрування можна застосовувати до таких пристроїв і як вони впливають на термін служби пристрою.

Для оцінки ефективності традиційних алгоритмів шифрування, що використовуються в комп’ютерних мережах, порівняно з легковаговими алгоритмами шифрування для IoT, необхідно розглянути та зрозуміти модель зв’язку типової системи IoT.

Модель зв’язку відображає основи IoT, що стосуються протоколів обміну інформацією, мережевих протоколів і програмного забезпечення. Вона є корисною на етапі проєктування архітектури IoT для розуміння інтероперабельності елементів і необхідного програмного забезпечення.

Парадигма Інтернету речей базується на двох моделях взаємодії/зв’язку: прямій (direct) та транзитній (transit).

При прямому з’єднанні пристрій IoT передає інформацію безпосередньо іншому кінцевому пристрою IoT (наприклад, датчик — актуатору) або хмарному сервісу, який обробляє дані та генерує відповідну дію.

У транзитних взаємодіях спеціальний пристрій або шлюз виконує роль посередника: отримує інформацію від інших пристроїв IoT і передає зібрані дані постачальнику послуг застосунку для обробки або, у випадку fog-обчислень, взаємодіє зі спеціальним пристроєм, який обробляє локальні запити.

Як зазначалося вище, моделі прямої взаємодії включають обмін даними device-to-device та device-to-cloud [10].

На рис. 1.2 представлено високорівневу класифікацію моделей зв’язку для IoT.

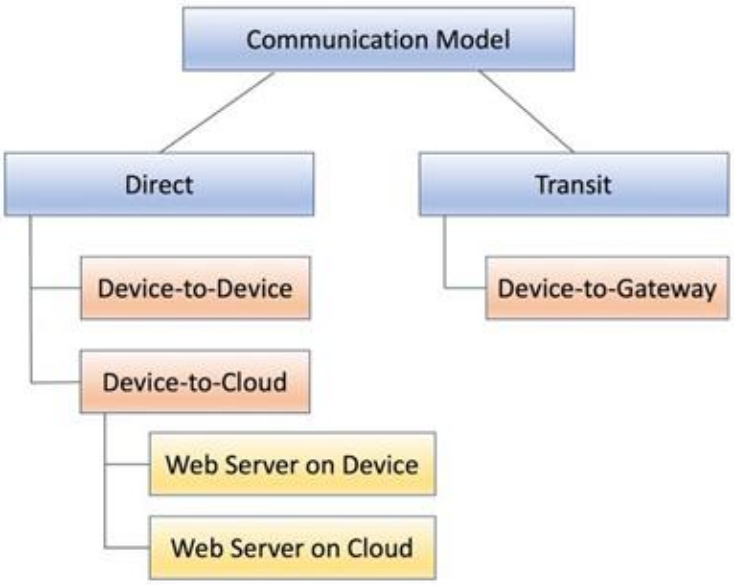

Рисунок 1.2 Класифікація моделей зв’язку в IoT.

Device-to-Device. У цій моделі два або більше пристроїв безпосередньо з’єднані між собою та обмінюються даними без проміжного пристрою (рис. 1.3). Модель device-to-device особливо поширена в системах домашньої автоматизації, HVAC-системах та системах моніторингу персонального здоров’я. Вона характеризується низькою швидкістю передачі, невеликим розміром пакетів, коли дані не обов’язково мають поширюватися серед багатьох користувачів.

Об’єкти в моделі device-to-device належать до другого типу пристроїв IoT — обмежених у ресурсах — і включають портативні та носимі пристрої, лампочки, вимикачі освітлення, термостати, дверні замки, монітори серцевого ритму, підключені до смарт-годинника тощо.

У більшості випадків підключення забезпечується протоколами короткого радіусу дії з низьким енергоспоживанням, такими як Bluetooth (включаючи енергоефективні варіанти — Bluetooth Smart або Bluetooth версії 4.0+), Z-Wave чи ZigBee, залежно від можливостей пристрою [11]. Низькоенергетичні протоколи зв’язку дозволяють пристроям працювати місяці або навіть роки від одного елемента живлення. Їхня нижча складність також дає

змогу зменшити розмір і вартість пристрою.

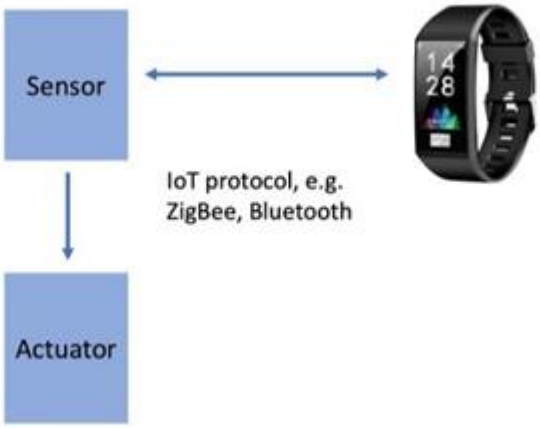

Рисунок 1.3 Зв’язок типу device-to-device.

У моделі device-to-device безпека спрощується завдяки використанню радіотехнологій короткого радіусу дії та близькості комунікаційних пристроїв. Однак у разі цілеспрямованих атак можливе дистанційне перехоплення трафіку за допомогою електромагнітних хвиль — цей вектор атак добре досліджений [12].Відсутність або слабке шифрування в комунікаціях device-to-device призводить до несанкціонованого перехоплення, модифікації або відстеження даних, що може негативно вплинути на безпеку всіх систем IoT, мережі та операційного рівня.

Device-to-Cloud. У цій моделі пристрої підтримують протоколи HTTP та TCP/IP, надають REST-інтерфейс і можуть безпосередньо підключатися до Інтернету через веб-API для обміну даними та керувальними повідомленнями. Для підключення використовуються, наприклад, Wi-Fi, стільниковий зв’язок або Ethernet, як показано на рис. 1.4. Ця модель зв’язку підходить для відносно потужних пристроїв, здатних запускати легковаговий веб-сервер. Зазвичай вбудовані веб-сервери мають значно обмеженіші ресурси порівняно з клієнтами, які до них звертаються (наприклад, браузерами або мобільними телефонами). Завдяки ефективній оптимізації крос-рівневого стеку HTTP і TCP та перенесенню обчислювально-інтенсивних завдань на сторону сервера такі веб-

сервери з розширеними функціями можуть мати обсяг пам’яті всього від 8 КБ.

Існує багато прикладів таких вбудованих веб-серверів, зокрема GoAhead, Barracuda Embedded Web Server, Lighttpd або Slinger від Neutrino [13].

Рисунок 1.4 Пряма модель зв’язку (Device-to-Cloud).

Об’єкти типу device-to-cloud належать переважно до потужних (resourceful) пристроїв, іноді — до обмежених у ресурсах, і включають пристрої на базі Raspberry Pi та Photon, розумні лампочки або камери спостереження, підключені через Wi-Fi, вбудовані споживчі IoT-пристрої, такі як Samsung Smart TV і Nest Learning Thermostat, розумні трекери або дверні дзвінки, які використовують механізм publish/subscribe для зв’язку зі смартфоном, що прослуховує події.

Підключення до хмари дозволяє користувачеві дистанційно взаємодіяти зі своїм пристроєм IoT через веб-застосунок: від перегляду домашньої камери спостереження чи віддаленого оновлення програмного забезпечення до підключення додаткових сервісів, таких як голосові асистенти або аналіз поведінки. У цих випадках модель device-to-cloud розширює можливості пристрою IoT, додаючи зручність кінцевому користувачеві.

З точки зору безпеки ця модель створює більше викликів, ніж альтернатива device-to-device, оскільки пристрої в домашній мережі повинні справлятися з тунелюванням, NAT або проходженням TCP-трафіку через

фаєрвол. Вона також безпосередньо піддає пристрій загрозам з боку Інтернету. Крім того, ця модель вимагає двох типів облікових даних: один — для рівня доступу до пристрою (наприклад, SIM-карта мобільного пристрою), а інший — для доступу до хмари.

Безпека в цьому типі зв'язку базується на шифруванні транспортного рівня, яке забезпечують протоколи на кшталт DTLS, а також на базовій технології (наприклад, Wi-Fi або 3G/LTE) [14]. Зазвичай пристрій IoT і хмарний сервіс належать одному постачальнику. Будь-яка спроба інтеграції пристроїв різних виробників в єдиний хмарний сервіс також підвищує проблеми безпеки.

Однією з модифікацій моделі device-to-cloud є випадок, коли пристрій IoT не може запустити HTTP-сервер і потребує потужної масштабованої хмарної платформи, яка виконує роль шлюзу. Така модель є «хмаро-орієнтованою» (cloud-powered): REST API надається хмарним сервером, а пристрій використовує інший протокол, наприклад CoAP або MQTT, для зв'язку з сервером.

Ця модифікована модель підходить для IoT-застосунків, розгорнутих на великій географічній території з великою кількістю пристроїв, які потребують централізованого керування (наприклад, датчики забруднення повітря). Обидві моделі представлено на рис. 1.5.

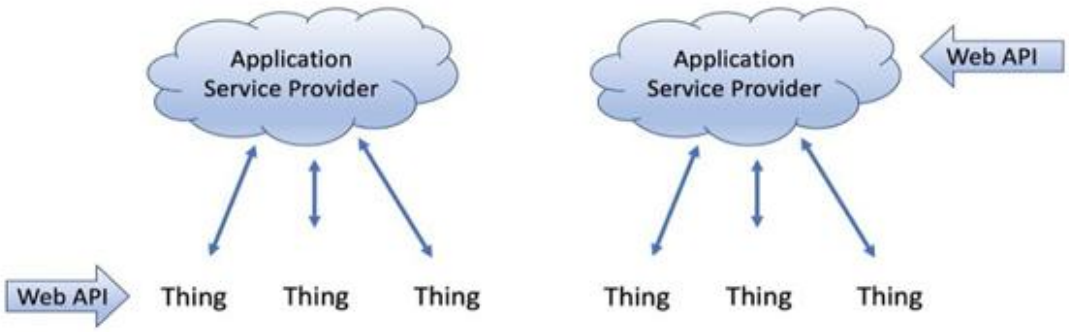

Рисунок 1.5 Приклади розгортання веб-сервера в моделі device-to-cloud.

Device-to-Gateway. Ця модель базується на проксі-зв'язку, коли пристрій

IoT використовує проміжний пристрій для доступу до постачальника послуг застосунку, як показано на рис. 1.6. У цій моделі пристрій IoT не здатен безпосередньо підтримувати HTTP-запити і потребує потужнішого пристрою, наприклад смартфона, який підключає носимі пристрої для моніторингу здоров’я або трекери активності [15].

Для вирішення цього завдання шлюз застосунку (application gateway) діє як проксі для пристрою, надаючи REST API, через який зовнішній застосунок може спілкуватися з проксі за допомогою простого HTTP-клієнта. Шлюз можна використовувати для підключення до мережі різноманітних існуючих пристроїв IoT.

Роль локального шлюзу зазвичай виконує смартфон, на якому запущено застосунок для зв’язку з пристроєм і передачі даних у хмару. Ця модель широко застосовується в популярних споживчих продуктах, таких як персональні фітнес-трекери з батарейним живленням, які не можуть використовувати Wi-Fi або Ethernet через високе енергоспоживання, але можуть застосовувати низькоенергетичні протоколи, такі як ZigBee або Bluetooth.

Рисунок 1.6 Транзитна модель зв’язку (Device-to-Gateway).

У цьому контексті «річ» доступна через протокол, що не базується на HTTP. Об’єкти типу device-to-gateway належать до другого типу пристроїв IoT — обмежених у ресурсах — і включають датчики дверей з батарейним

живленням, датчики та актуатори в системах домашньої автоматизації, які підключаються до шлюзу через трансивери Z-Wave, ZigBee або інші технології передачі даних, а згодом — до хмарного сервісу (наприклад, екосистема SmartThings від Samsung).

У моделі device-to-gateway локальний шлюз забезпечує сервіси безпеки, такі як захист передачі та доставки даних. Шлюз може додавати рівень захисту або автентифікації, тимчасово збирати та зберігати дані, надавати семантичні описи для «речей» тощо.

Таким чином, обмежені в ресурсах пристрої, які є основним фокусом цього дослідження, розгортаються в моделях device-to-device, device-to-gateway та хмаро-орієнтованій device-to-cloud.

Для всіх цих сценаріїв проблема впровадження безпеки становить значний інтерес, оскільки обмежені в ресурсах пристрої, як уже зазначалося, не мають обчислювальних або енергетичних ресурсів, необхідних для традиційних методів захисту [16].

Цей виклик спричинив низку досліджень у сфері легковагової криптографії (light cryptography), яка пропонує компромісне рішення безпеки за допомогою низьковартісних, низьколатентних реалізацій, що споживають менше пам’яті [17].

Для оцінки ефективності легковагових алгоритмів (Light Weighted Algorithms — LWA) або шифрів (Lightweight Ciphers — LWC) використовуються різні параметри, які дозволяють їх порівнювати: рівень безпеки, споживання потужності та енергії, час виконання, площа чіпа, показник ефективності (Figure of Merit — FOM) або затримка.

Хоча всі ці параметри важливі для наукової спільноти, більш надійним і всеохопним показником можливостей пристрою є його термін служби. Тому метою нашої роботи є дослідження того, як вибір алгоритму захисту впливає на термін служби пристрою та системи.

Наступний розділ містить огляд існуючих підходів до оцінки ефективності криптографії в IoT та оцінки терміну служби пристроїв IoT [18].

1.3 Аналіз існуючих рішень та пов’язаних досліджень у сфері IoT-безпеки

Зважаючи на попередні дослідження, що стосуються терміну служби пристроїв IoT, слід зазначити, що жодне з них не розглядало одночасно енергоспоживання на передачу даних та енергоспоживання на забезпечення захисту даних.

Багато робіт пропонують окремі оцінки часу виконання шифрування/розшифрування для різних шифрів, енергоспоживання та потужності під час шифрування/розшифрування, або енергоспоживання під час передачі даних за допомогою різних технологій зв’язку. Хоча ці параметри дійсно є важливими, фактичний термін служби батареї/пристрою є значно більш інформативним показником, оскільки він чітко вказує, як довго система потенційно зможе працювати автономно без заміни елементів живлення.

Усі дослідження можна розділити на дві основні групи: (a) дослідження терміну служби пристрою за різних мережевих топологій, фізичних умов та/або вимог до обробки даних; (b) дослідження, що вивчають безпеку передачі даних.

Типовим прикладом є де порівнювали різні легковагові алгоритми (Twofish, Blowfish, DES, 3DES, AES, RC2, RC4 та ChaCha20) за розміром блоку, розміром ключа, часом виконання, споживанням процесора та пам’яті. Ці шифри тестувалися на пристроях IoT на файлах розміром від 1 МБ до 128 МБ. Результати показали, що алгоритм Twofish мав найвищу швидкість виконання серед усіх блочних шифрів, а потоковий шифр ChaCha20 продемонстрував найкращі показники ефективності за всіма параметрами як серед блочних, так і серед потокових шифрів [19].

Проведено оцінку різних блочних шифрів та ранжування їх продуктивності на основі використання пам’яті, завантаження процесора та обчислювальної вартості. Алгоритми: XXTEA, RC5, AES, CGEA. Автори дійшли висновку, що RC5 є найбільш ефективним за використанням пам’яті,

але рекомендували застосовувати AES-256 або CGEA через більший розмір ключа.

Аналізували різні існуючі легковагові рішення для IoT. Дослідження містить аналіз алгоритмів за розміром ключа, розміром блоку, кількістю раундів та можливими атаками на основі огляду літератури. Автори розглянули шифри PRESENT, HIGHT, RSA, ECC, AES і дійшли висновку, що для сценаріїв IoT більш придатними є симетричні алгоритми завдяки меншим ключам і зниженій складності, що призводить до швидшого шифрування та меншого навантаження на процесор. Однак усі вони мають однакове суттєве обмеження — безпечний обмін секретним ключем, тому покладаються на більш захищений метод розподілу секретного ключа з подальшим використанням симетричної криптографії для забезпечення конфіденційності та цілісності [20].

У роботі представлено огляд сучасної легковагової криптографії. Автори провели структурний аналіз кількох криптографічних алгоритмів, зокрема PRINT, SIMON, KATAN, HISEC, OLBCA, PRINCE, PRESENT, KLEIN, TWINE. Деякі з них використовують мережу Фейстеля, інші — SPN, але кожен має власні властивості. Було виконано порівняльний аналіз за архітектурою, розміром ключа, розміром блоку та кількістю раундів. Аналіз показав, що якщо алгоритм має достатню кількість S-блоків і добре розвинені лінійні операції, то він забезпечує високий рівень безпеки, а вартість залежить від дизайну.

Оцінювали продуктивність різних шифрів з точки зору енергоспоживання. Для моделювання вони обрали алгоритми HIGHT і KATAN через наявність глибокого аналізу варіантів дизайну та результатів у літературі. Результати показали, що оптимальне енергоспоживання досягається при розмірі блоку від 48 до 96 біт та кількості раундів 16 або менше.

Хоча багато оглядів включали різні алгоритми шифрування для IoT, більшість з них фокусувалися на аналізі одного алгоритму або порівнянні лише кількох [21]. Деякі роботи надають оцінку продуктивності для більшого списку легковагової криптографії (LWC), проте глибокий аналіз не проводився. Найсвіжіше дослідження, яке забезпечує комплексне перехресне порівняння

шифрів LWC, включаючи 54 реалізації [22].

Друга група досліджень зосереджувалася на оцінці терміну служби батареї/пристрою та вивченні шляхів його підвищення. Робіта аналізує термін служби лише під час передачі даних залежно від використаних бездротових технологій. Інші дослідження розглядають, як структурні характеристики системи IoT (гетерогенність пристроїв, топологія мережі тощо) впливають на термін служби. Робота досліджує вплив характеристик самого пристрою, таких як тривалість циклу сенсингу, збирання даних, режими активності, на термін служби пристрою.

Запропонували розумне перепланування робочих циклів (duty-cycles) для збільшення терміну служби пристрою та оцінили своє рішення за допомогою інструменту симуляції. Під час симуляції не враховувалися алгоритм безпеки та технологія передачі даних.

Споживання енергії та термін служби пристрою також спеціально для різних датчиків з батарейним живленням у системах домашньої автоматизації. Характеристики вимірювалися протягом тривалого часу для різних режимів роботи датчиків (сплячий, активний) та з різних місць у будинку (туалет, спальня, кухня) з використанням технології 6LoWPAN [23]. Мета дослідження — надати базові дані для подальшого прогнозування. Автори не згадували, чи була передача даних зашифрованою.

Розглядали питання терміну служби пристрою лише з позиції технології бездротової передачі. Вони дослідили енергоспоживання та термін служби пристрою при різній тривалості циклу та розмірі пакета даних для технологій 6LoWPAN, 802.15.4, 802.11ah, Bluetooth Low Energy, LoRa та SIGFOX [24]. Результати показали, що для малих обсягів даних найбільший термін служби забезпечує BLE, хоча 802.15.4 ненамного відстає; для ультранизької інтенсивності трафіку найкращі показники мали LoRa та SIGFOX, тоді як при високій інтенсивності передачі даних кращим було використовувати 802.11. Автори не враховували накладні витрати на безпеку для кожної технології, за винятком SIGFOX, для якого вони додали 2 байти до розміру пакета для

врахування використання HMAC для автентифікації повідомлень.

У роботі склали перелік параметрів, що впливають на енергоспоживання та термін служби пристроїв IoT. Серед них: робочий цикл радіомодуля на рівні MAC, розмір заголовка, протокол прикладного рівня, протокол зв'язку. У ході експерименту визначено мінімальні та максимальні значення терміну служби (у годинах) для кожного параметра [25]. Автори не згадували жодних механізмів безпеки у зв'язку з енергоспоживанням.

Вирішували задачу оптимізації топології для подовження терміну служби мережі IoT. Їх математичний підхід використовує заздалегідь задані значення енергії кожного вузла та балансує їх. Конкретні параметри мережі чи пристрою (наприклад, використання CPU та пам'яті, розмір і частота передачі даних, можливість застосування шифрування чи інших методів захисту) до дослідження не включалися.

Досліджували споживання енергії та термін служби пристрою при різній тривалості циклу сенсингу. Запропонований метод розраховує оптимальний період сенсингу на основі даних навчання, які включають залишковий час, залишкову потужність та обсяг спожитої потужності кожного вузла [26].

На основі аналізу існуючих рішень для оцінки терміну служби пристроїв IoT можна виділити три основні групи факторів, що на нього впливають:

• Технічні характеристики: площа кристалу, критична дальність передачі, рівень сигналу, бездротовий адаптер, виробник апаратного забезпечення, використання процесора та пам'яті.

• Функціональні характеристики: протоколи бездротового зв'язку, протоколи передачі даних, характер і частота передаваних даних, можливість використання шифрування та інших методів захисту.

• Структурні характеристики: зокрема топологія мережі та гетерогенність пристроїв IoT.

Цей короткий огляд демонструє, що в умовах обмежених ресурсів пристроїв IoT використання традиційних технологій зв'язку як для передачі даних, так і для їх захисту є вкрай складним завданням. Саме тому цей етап

впровадження систем IoT характеризується великою різноманітністю рішень, що також підтверджує важливість і актуальність досліджень терміну служби пристроїв IoT.

Автори повинні обговорити отримані результати та те, як їх можна інтерпретувати з точки зору попередніх досліджень і робочих гіпотез. Результати та їх наслідки слід розглядати в максимально широкому контексті. Також можна виділити напрямки майбутніх досліджень.

РОЗДІЛ 2 МЕТОДИ, МОДЕЛІ ТА АЛГОРИТМИ ШИФРУВАННЯ

ДАНИХ ДЛЯ IoT-ПРИСТРОЇВ

2.1 Огляд сучасних криптографічних алгоритмів для IoT

Більшість алгоритмів, які наразі використовуються в IoT, належать до групи легковагової криптографії (LWC) — симетричних блочних шифрів. Такі шифри базуються на двох типах структури: мережі заміщення-перестановки (Substitution-Permutation Network — SPN) та структурі Фейстеля [27].

Структура Фейстеля розділяє блок даних на дві рівні частини і використовує раунди для шифрування, причому кожен раунд має два окремі процеси: один для шифрування відкритого тексту та один для техніки заміщення. Процес розшифрування подібний до шифрування, відмінність полягає лише в тому, що ключі застосовуються у зворотному порядку. Як легко зрозуміти, рівень безпеки прямо пропорційний кількості раундів, але це збільшення відбувається за рахунок вищої затримки (latency), пов'язаної з реалізацією. Повільне шифрування та розшифрування є недоліком цієї структури, що робить її непридатною для мереж з низькою затримкою.

Головною перевагою структури Фейстеля є низьке споживання пам'яті завдяки використанню одного й того самого програмного коду для процесів шифрування та розшифрування. Це можна ефективно реалізувати в IoT-пристроях з батарейним живленням та низьким середнім енергоспоживанням.

Існує кілька шифрів на основі Фейстеля, зокрема: DES, TEA, Camellia, SEA, CLEFIA, TWINE, LBlock, Piccolo, Blowfish, HIGHT [28].

Структура SPN використовує одну раундову функцію, яка застосовується до всього блоку даних. Вона базується на принципі Шеннона щодо конфузії (confusion), що реалізується через заміщення, та принципі дифузії (diffusion) за допомогою лінійної трансформації. Безпека залежить від складності лінійної функції. Процес розшифрування виконується простим оберненням операцій шифрування і забезпечує швидшу обчислювальну продуктивність порівняно з шифрами Фейстеля.

Головною перевагою SPN є можливість реалізації з низькими ресурсними витратами, оскільки SPN потребує менше енергії, ніж інші структури, для забезпечення того самого рівня безпеки, завдяки меншій кількості раундів виконання. Недоліком є високий рівень вразливості SPN-алгоритмів до диференціального та лінійного криптоаналізу через відсутність розкладу ключів (key schedule).

До SPN-шифрів належать: AES, PRESENT, KLEIN, Serpent.

Асиметричні шифри менш використовуються в IoT, але є дуже популярними для пристроїв з достатніми ресурсами, оскільки вони значно більш обчислювально затратні, ніж симетричні алгоритми. Такі шифри також можуть застосовуватися для обміну ключами та автентифікації IoT-пристрою перед шифруванням і передачею даних. Типовими прикладами є RSA та ECC.

За конструкцією ECC потребує меншого розміру ключа, ніж RSA, для забезпечення еквівалентного рівня безпеки. Спостерігається тенденція переходу від RSA до ECC — популярного вибору розробників систем безпеки IoT, який рекомендований компанією Symantec, оскільки вимоги до безпеки змушують збільшувати розмір ключів. Крім того, вони включені до більшості систем керування криптографічними ключами, що використовуються малим і середнім бізнесом, що робить їх привабливими для компаній [29].

Недоліком симетричних шифрів є великий розмір ключа, вище споживання пам’яті та низька швидкість виконання. Це є викликом для розробників і дослідників щодо вдосконалення ECC шляхом зменшення вимог до пам’яті та обчислювальної складності.

Для оцінки терміну служби IoT-пристрою з застосованим криптографічним механізмом ми, після глибокого аналізу сучасних досліджень безпеки IoT, обрали кілька симетричних алгоритмів та алгоритм ECC: AES, TEA, XTEA, HIGHT, KLEIN, Blowfish, Twofish, Serpent та ECC.

Tiny Encryption Algorithm (TEA), є легковаговим шифром з архітектурою Фейстеля, реалізованим у дуже короткому програмному коді з простими операціями XOR, додавання та зсуву бітів. Алгоритм працює з 64-бітним

блоком даних і 128-бітним ключем. Він вважається безпечним, але через низьку складність вразливий до атак, пов’язаних з ключем. Для підвищення безпеки TEA було модифіковано в Extended TEA (XTEA) та Block TEA (XXTEA) [30].

XTEA використовує 64-бітний блок, має 32 повних цикли, у кожному з яких два раунди структури Фейстеля. XXTEA має 64-бітний блок і 128-бітний ключ із змінною кількістю раундів мережі Фейстеля (від 6 до 32 повних циклів). Криптоаналіз XXTEA показав, що він стійкий до атаки на основі відкритого тексту з використанням диференціального криптоаналізу при 2^{59} запитах [31].

Advanced Encryption Standard (AES) — це блочний шифр на основі SPN, стандартизований NIST (також відомий як Rijndael). AES має фіксований розмір блоку 128 біт і працює з матрицею 4×4 байти. Алгоритм використовує чотири трансформації для перетворення відкритого тексту в шифротекст: SubBytes, ShiftRows, MixColumns та AddRoundKey (XOR з раундовим ключем). Доступні розміри ключів — 128, 192 та 256 біт, що відповідає 10, 12 і 14 раундам. Чим більший розмір ключа, тим сильніше шифрування, проте AES залишається вразливим до атак типу «людина посередині» (man-in-the-middle) [31]. Зазвичай сенсори реалізують AES-128, але для сильно обмежених ресурсів AES може бути надто енергоємним.

PRESENT — ультралегковаговий блочний шифр на основі SPN, стандартизований ISO/IEC. Має розмір блоку 64 біти, розмір ключа 80 або 128 біт і 31 раунд. Розмір коду дуже малий (близько 1000 байтів), використовується лише один S-бокс замість восьми. Низьке споживання пам’яті та низька складність дозволяють реалізувати цей шифр у RFID-мітках, що неможливо зі стандартним AES. Алгоритм орієнтований на апаратну реалізацію і став основою для багатьох інших ультралегковагових шифрів. Незважаючи на компактність, PRESENT вразливий до диференціальних атак по побічних каналах та атак, пов’язаних з ключем, на зменшеній кількості раундів (17–26 з 31), як і всі шифри з коротким ключем.

KLEIN — типовий ультралегковаговий SPN-шифр з розміром блоку 64 біти та розмірами ключів 64, 80 і 96 біт (відповідно 12, 16 і 20 раундів). Шифр

можна ефективно реалізувати як у програмному вигляді на платформах сенсорів (що забезпечує гнучкість і нижчу вартість розгортання), так і в компактній апаратній реалізації. Для досягнення компактності використовується операція множення байтів (наприклад, MixColumns) замість зсуву бітів. Розробники заявляють про стійкість KLEIN до різних методів криптоаналізу, проте, як і всі шифри з коротким ключем, він вразливий до атак, пов'язаних з ключем, до 8 раундів зі 12 у версії KLEIN-64 через слабкості шару дифузії та розкладу ключів [32].

Elliptic Curve Cryptography (ECC) — асиметричний шифр, що базується на алгебраїчній структурі еліптичних кривих над скінченними полями. Він використовує скалярне множення, яке включає операції додавання та подвоєння точок. Розмір ключа ECC дорівнює розміру поля, над яким визначена еліптична крива. ECC є перспективним варіантом легковагової криптографії, оскільки вимагає меншого розміру ключа та меншого обсягу пам'яті порівняно з RSA, завдяки чому працює швидше і може бути реалізований на ресурс-обмежених пристроях [33]. Для оптимізації в низькоспоживальних пристроях ECC використовує операції зсуву бітів замість складного множення. Рівень безпеки, який забезпечує 1024-бітний ключ RSA, досягається 160-бітним ключем у ECC.

Blowfish — легковаговий шифр на основі Фейстеля з розміром блоку 64 біти та змінним розміром ключа від 32 до 448 біт. Має 16 раундів структури Фейстеля та використовує великі S-бокси, залежні від ключа. Невеликий ключ ідеально підходить для шифрування на ресурс-обмежених пристроях. Це відкритий алгоритм, розроблений Шнайєром, проти якого не виявлено ефективних атак. Дослідники успішно ламали ключ, але не більше ніж на четвертому раунді, а виявлення ключа — не більше ніж на 14-му раунді [34].

Twofish — блочний шифр з архітектурою Фейстеля, подібний до AES, DES і Blowfish. Має розмір блоку 128 біт, розміри ключів 128, 192 і 256 біт та 16 раундів. Відкритий текст розбивається на два 32-бітні слова, які шифруються через F-бокси. Кожен F-бокс містить шар з чотирьох S-боксів, залежних від

різних ключів, і 4-байтову лінійну трансформацію на основі матриці MDS (Maximum Distance Separable). Для поєднання двох 32-бітних слів використовується Pseudo-Hadamard Transform (PHT). Потім два додаткові 128-бітні підключі XORуються з даними. Twofish досі не зламаний: будь-яка лінійна атака вимагає щонайменше $2^{120},^{8}$ вибраних відкритих текстів, а успішна диференціальна атака може зламати максимум 5 раундів.

HIGHT — шифр на основі Фейстеля з 64-бітним блоком, 128-бітним ключем і 32 раундами. Розроблений спеціально для середовищ з обмеженими ресурсами. Паралельна реалізація підвищує швидкість і дозволяє використовувати алгоритм у RFID-системах. Усі операції — прості обчислення (mod 2^8 або XOR) без використання S-боксів. З точки зору безпеки HIGHT забезпечує достатній рівень захисту, але вразливий до диференціальних і насичувальних атак [35].

Piccolo — ультралегковаговий блочний шифр на основі Фейстеля, придатний для надзвичайно обмежених середовищ, таких як RFID-мітки. Використовує 64-бітний блок, підтримує ключі 80 і 128 біт з 25 і 31 раундами відповідно. Один раунд Piccolo складається з двох функцій: AddRoundKey (XOR з раундовим ключем) та Round Permutation. Кожна з них включає операції SubNibble, MixColumn і SubNibble. Piccolo, як і PRESENT, є апаратно-орієнтованим легковаговим шифром. Він забезпечує ефективний рівень безпеки та стійкий до атак, пов’язаних з ключем, і атак «людина посередині». Тестування показало, що ключ можна відновити лише при зменшенні кількості раундів до 14 і 21 (без whitening) для Piccolo-80 і Piccolo-128 відповідно.

Serpent — блочний шифр на основі SP-мережі з розміром блоку 128 біт і довжиною ключа 128, 192 та 256 біт при 32 раундах. Кожен раунд включає операції змішування ключа, заміщення через S-бокси та лінійну трансформацію. В останньому раунді лінійна трансформація замінюється додатковим змішуванням ключа. Усі 32 раунди використовують 32 різні S-бокси, кожен з яких відображає 4 вхідні біти на 4 вихідні. Розшифрування відрізняється від шифрування: використовується обернена лінійна трансформація та зворотний

порядок ключів. Serpent вважається одним з найбезпечніших алгоритмів (навіть безпечнішим за Rijndael), але має повільну швидкість реалізації. Оскільки пристрої IoT передають дані нечасто, Serpent є одним з ідеальних кандидатів для шифрування в IoT.

2.2 Оцінка енергоспоживання та обчислювальної ефективності алгоритмів шифрування

Термін служби IoT-пристрою є однією з критичних характеристик під час розгортання технологій IoT. Важливо заздалегідь оцінити приблизний час роботи кожного вузла до повного розряду батареї. Цей показник безпосередньо залежить від життєвого циклу пристрою, який включає кілька факторів: режими роботи пристрою (періоди активного та пасивного режимів), топологію мережі Інтернету речей, технологію зв'язку, тип і обсяг передаваних даних, а також механізм безпеки [36].

Життєвий цикл кінцевого IoT-пристрою зображено на рис. 2.1.

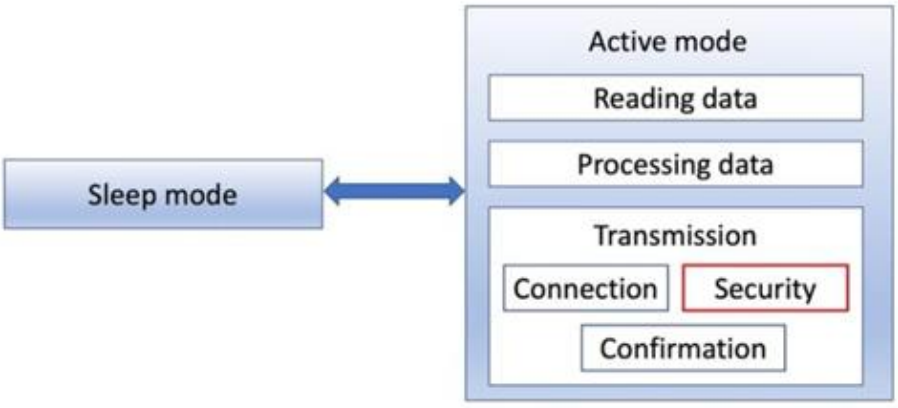

Рисунок 2.1 - Цикл роботи IoT-пристрою.

Під час процесу передачі даних система повинна застосовувати заходи безпеки для забезпечення конфіденційності та цілісності даних. Застосовані заходи безпеки суттєво впливають на термін служби пристрою, залежно від

механізму автентифікації та вибору алгоритму шифрування даних, що передаються. У останньому випадку шифрування/розшифрування потоку даних є найбільш енергоємною діяльністю, пропорційною обсягу самого трафіку.

Для оцінки терміну служби пристрою ми використовуємо вдосконалену модель, яка дозволяє оцінювати термін служби пристрою як функцію від алгоритму шифрування та довжини переданого пакета. Для цього в межах вдосконаленої моделі IoT-пристрою ми вводимо такі позначення:

- E_{IoT} — початкова енергія батареї пристрою (Дж);
- P_{IoT} — загальна потужність, яку споживає пристрій за один цикл (Вт);
- LT_{IoT} — термін служби IoT-пристрою.

Тоді, згідно з [3], термін служби IoT-пристрою можна розрахувати за таким рівнянням:

$$LT_{IoT} = \frac{E_{IoT}}{P_{IoT}} \tag{2.1}$$

Далі, відповідно до [37], традиційне рівняння енергоспоживання для кінцевого IoT-пристрою має вигляд:

$$P_{IoT} = \frac{P_A \cdot t_A + P_S \cdot t_S}{t_c} \tag{2.2}$$

де

- P_A — енергоспоживання в активному режимі, мВт;
- t_A — загальний час перебування в активному режимі, с;
- P_S — енергоспоживання в сплячому режимі, мВт;
- t_S — загальний час перебування в сплячому режимі, с;
- t_c — тривалість одного циклу роботи IoT-пристрою, с.

Як показано на рис. 2.1, активний режим включає процеси зчитування та обробки даних, передачі даних і підтвердження, а також шифрування та

розшифрування. Відповідно, кожен із цих процесів характеризується своєю тривалістю.

Тоді, враховуючи час перебування в активному режимі, енергоспоживання під час активної фази розраховується за таким рівнянням:

$$P_A \cdot t_A = P_D \cdot t_D + P_T \cdot t_T + P_{sec} \cdot t_{sec} \quad (2.3)$$

де

- P_D — середня потужність під час зчитування та обробки даних, мВт;
- t_D — час, витрачений на зчитування та обробку даних, с;
- P_T — середня потужність під час передачі даних і подальшого підтвердження, мВт;
- t_T — час, витрачений на передачу даних і отримання підтвердження, с;
- P_{sec} — загальна потужність на шифрування та розшифрування даних, мВт;
- t_{sec} — загальний час, витрачений на шифрування та розшифрування даних, с (залежить від типу криптографічного алгоритму).

У рівняннях (2.2) і (2.3) величини P_S і P_D є постійними і визначаються особливостями конкретної апаратної реалізації IoT-пристрою.

Потужність P_T залежить від використовуваного стандарту бездротової передачі, частоти вихідних пакетів F, розміру пакета (у бітах) L_{packet} передаваних даних, а також енергії, витраченої на передачу одного біту E_T, і розраховується за таким рівнянням:

$$P_T = E_T \cdot L_{packet} \cdot F \quad (2.4)$$

Відповідно до [20], час t_T, витрачений на передачу даних і отримання підтвердження, дорівнює:

$$t_T = t_{wait} + t_{ch} + t_{tr} + t_{conf} \quad (2.5)$$

де

- t_{ch} — постійний час прослуховування каналу для визначення його зайнятості, що дорівнює 8 періодам символу або 128 мкс;
- t_{conf} — час передачі підтвердження, с;
- t_{wait} — час очікування передачі даних, с, який розраховується як:

$$t_{wait} = W \cdot H \quad (2.6)$$

і залежить від випадкового часового інтервалу W, який є цілим числом, що випадково обирається кожного разу і визначає зайнятість каналу. Наприклад, в усіх редакціях стандарту IEEE 802.15.4 для діапазону 2,4 ГГц один період символу становить W=3 мкс у найкращому випадку та W=7 мкс у найгіршому; H постійно дорівнює періоду 20 символів.

Час, витрачений безпосередньо на передачу даних ttr t_{tr} ttr , можна розрахувати за рівнянням:

$$T_{tr} = \frac{(L_{packet}+L_{field})\cdot 8}{Ch} \quad (2.7)$$

де

- L_{field} — розмір службових полів пакета, байти;
- Ch — швидкість передачі даних каналом, кбіт/с.

У свою чергу, величина $P_{sec} \cdot t_{sec}$ з рівняння (2.7) включає два процеси — шифрування та розшифрування, кожен з яких потребує різного часу залежно від типу алгоритму шифрування. Загальне енергоспоживання на шифрування та розшифрування даних $P_{sec} \cdot t_{sec}$ розраховується як:

$$P_{sec} \cdot t_{sec} = P_{sec\,(enc)} \cdot t_{sec\,(enc)} + P_{sec\,(dec)} \cdot t_{sec\,(dec)} \quad (2.8)$$

де $P_{sec\,(enc)} \cdot t_{sec\,(enc)} + P_{sec\,(dec)} \cdot t_{sec\,(dec)}$ — енергоспоживання на шифрування протягом часу $t_{sec\,(enc)}$ та розшифрування протягом часу $t_{sec\,(dec)}$ відповідно, мВт.

Таким чином, змінні $P_{sec\,(dec)}$ і $P_{sec\,(enc)}$ визначаються за такими рівняннями:

$$P_{Sec(enc)} = V_{enc} \cdot I_{enc} \quad (2.9)$$

$$P_{Sec(dec)} = V_{dec} \cdot I_{dec} \quad (2.10)$$

де $V_{enc} \cdot I_{enc}$ — напруга та струм для процесу шифрування відповідно;

$V_{dec} \cdot I_{dec}$ — напруга та струм для процесу розшифрування відповідно.

Тоді, на основі рівнянь (2)–(10), рівняння (1) можна переписати в такому вигляді:

$$LT_{IoT} = \frac{E_{IoT} \cdot t_c \cdot Ch}{P_D \cdot t_D \cdot Ch + (E_T \cdot L_{packet} \cdot F) \cdot t_T \cdot 8 + P_{sec} \cdot t_{sec} \cdot Ch} \quad (2.11)$$

Розрахунок терміну служби IoT-пристрою за запропонованою моделлю. Для розрахунку терміну служби IoT-пристрою ми використали Arduino Mega 2560 як типовий пристрій IoT. Ця платформа була обрана через її продуктивність і сумісність з широким спектром плат розширення. Оскільки дослідження зосереджене на енергоспоживанні зовнішньої батареї, підключеної до цього пристрою, ємність батарей є визначальним фактором у розрахунках.

Для оцінки продуктивності різних шифрів необхідно виконати два етапи:

1. Експериментальне дослідження обраних алгоритмів шифрування, розгорнутих на двох платах Arduino Mega 2560, що виступають як кінцеві пристрої (передавач і приймач), які обмінюються даними через бездротовий

модуль RF 433 MHz з використанням обраних алгоритмів шифрування, реалізованих за допомогою бібліотек GitHub. Було досліджено загалом десять алгоритмів шифрування (AES, XTEA, HIGHT, KLEIN, ECC, BLOWFISH, Twofish, Serpent, Piccolo та PRESENT) при різних розмірах пакетів у межах одного циклу. Для забезпечення однакового та порівнянного рівня безпеки більшість шифрів було налаштовано з розміром ключа 128 біт, за винятком ECC (160 біт) та KLEIN (96 біт).

Таблиця 2.1.

Вхідні дані для передачі

Параметр, одиниця вимірювання (позначення)	Значення
Початкова енергія вузла (EIoT E_{IoT} EIoT), кДж	20
Розмір пакета (Lpacket L_{packet} Lpacket), байти	50, 100, 200
Частота процесора (F F F), МГц	16
Енергоспоживання в активному режимі (PA P_A PA), мВт	0,02
Енергоспоживання в сплячому режимі (PS P_S PS), мВт	0,003
Швидкість передачі каналу (Ch C_h Ch), кбіт/с	250
Розмір службових полів пакета (Lfield L_{field} Lfield), байти	17
Час сну (ts t_s ts), с	8

Для вимірювання часу шифрування/розшифрування та енергоспоживання ми використовували функцію micros() з бібліотеки Arduino, яка повертає кількість мікросекунд з моменту запуску поточної програми на платі Arduino. Для вимірювання енергоспоживання використовувалися значення напруги V та струму I , отримані за допомогою аналого-цифрового мультиметра METRAHit 16S.

2. Розрахунок терміну служби відповідно до рівняння (2.11) з використанням значень, отриманих на етапі 1, як вхідних даних, а також параметрів, наведених у таблиці 2.1.

Ми використовуємо максимально можливий інтервал у 8 секунд для переведення пристрою в режим глибокого сну (power down sleep mode). Частота процесора — це тактова частота Arduino, яка визначає, скільки операцій плата може виконати за секунду. Стандартна тактова частота для більшості мікроконтролерів Arduino становить 16 МГц, що відповідає 16 мільйонам інструкцій за секунду.

Таблиця 2.2.

Час шифрування та розшифрування для різних розмірів пакетів

Алгоритм	**L = 50 байт**	**L = 100 байт**	**L = 200 байт**
	Шифр., мс	Розш., мс	Шифр., мс
AES	5,76	6,76	8,38
XTEA	6,24	6,24	9,36
HIGHT	8,32	8,34	12,20
KLEIN	5,84	5,88	9,06
ECC	10,84	10,80	13,12
Blowfish	27,70	41,20	43,00
Twofish	25,40	37,40	41,30
Serpent	21,40	38,30	30,70
Piccolo	5,62	3,35	7,74
PRESENT	14,58	30,12	22,46

Етап 1 являв собою експериментальне дослідження заданих алгоритмів шифрування для відображення залежності між часом шифрування/розшифрування та енергоспоживанням як функції довжини пакета. Результати цих експериментів узагальнено в таблицях 2.2 і 2.3.

Таблиця 2.3.

Загальне енергоспоживання криптографічних алгоритмів, мВт

Алгоритм	**L = 50 байт**	**L = 100 байт**	**L = 200 байт**
AES	0,56	0,71	1,20
XTEA	0,61	0,85	1,32
HIGHT	1,21	1,71	2,21
KLEIN	0,65	0,90	1,47
ECC	6,37	8,46	9,30
Blowfish	5,96	7,02	10,64
Twofish	5,57	7,87	10,10
Serpent	5,68	7,99	10,20
Piccolo	0,16	0,66	1,22
PRESENT	4,24	7,60	11,20

Щодо енергоспоживання (таблиця 2.3), результати показали, що деякі шифри зі структурою Фейстеля (Piccolo та XTEA) потребують менше або подібної енергії порівняно з алгоритмами, що використовують структуру SP, що суперечить теоретичному припущенню про менші енергетичні витрати шифрів зі структурою SP-мережі порівняно з Фейстелем. Крім того, що Piccolo є найшвидшим алгоритмом, він також досяг найнижчого енергоспоживання.

Хоча шифри на основі SPN вважаються швидшими у виконанні, ніж шифри Фейстеля, результати показують, що деякі SPN-алгоритми, зокрема Serpent і PRESENT, мають вищий час шифрування та розшифрування, тоді як Piccolo, XTEA та HIGHT працюють відносно швидко серед шифрів Фейстеля.

На основі отриманих значень енергоспоживання (рис. 2.2) та часу шифрування/розшифрування термін служби IoT-пристрою можна розрахувати відповідно до рівнянь (2)–(11). Значення наведено в таблиці 2.4 і на рис. 2.2.

Таблиця 2.4.

Результати розрахунку терміну служби IoT-пристроїв, днів

Алгоритм	**L = 50 байт**	**L = 100 байт**	**L = 200 байт**
	PIoT P_{IoT} PIoT	LTIoT LT_{IoT} LTIoT	PIoT P_{IoT} PIoT
AES	1,12	318	1,46
XTEA	1,21	316	1,72
HIGHT	1,86	269	2,04
KLEIN	1,30	312	1,60
ECC	12,10	113	17,23
Blowfish	11,40	149	14,52
Twofish	11,03	152	15,37
Serpent	11,10	151	15,41
Piccolo	0,22	445	1,30
PRESENT	10,20	178	15,10

Результати показують, що ECC-160 є найбільш енергоємним шифром, тому він найменш бажаний для ресурс-обмежених пристроїв і повинен використовуватися лише у випадках, коли безпека даних є критичною для відповідного процесу, оскільки ECC-160 відповідає 1024-бітному ключу в RSA, тобто забезпечує високий рівень безпеки. Згідно зі значеннями з таблиці 2.1, термін служби вузла з шифруванням ECC становитиме 51–113 днів (2–3,5 місяці) залежно від розміру пакета [38].

Хоча алгоритм HIGHT був спеціально розроблений для RFID-міток з акцентом на низькі ресурсні витрати зв’язку, результати свідчать, що пристрої, які його використовують, матимуть менший термін служби порівняно з алгоритмами AES, XTEA та KLEIN при застосуванні 128-бітного ключа.

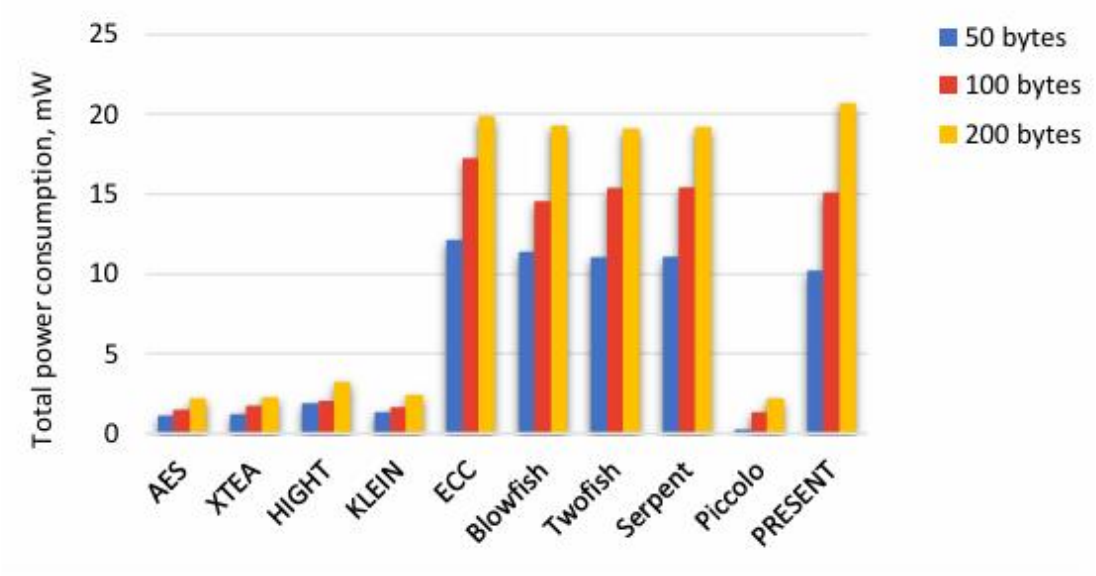

Рисунок 2.2 - Загальне енергоспоживання IoT-пристрою з застосуванням заходів безпеки.

AES, Piccolo-128, XTEA та KLEIN є найбільш бажаними варіантами при поєднанні вимог до енергоспоживання та максимального терміну служби. Завдяки простій структурі та використанню інволютивного S-блоку алгоритми AES і KLEIN призначені головним чином для зниження вартості реалізації, вважаються відносно гнучкими та забезпечують середній рівень безпеки.

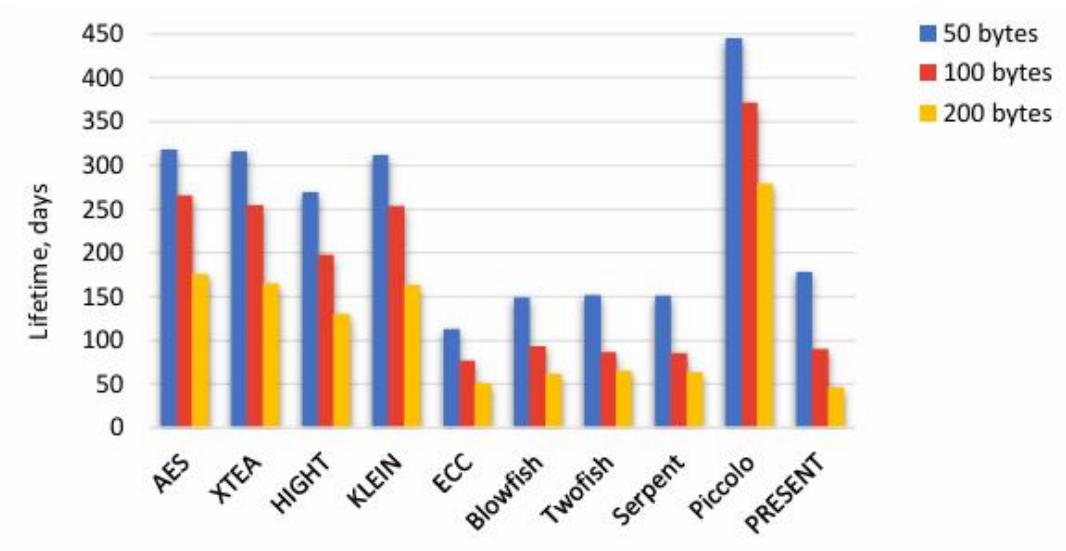

Рисунок 2.3 - Термін служби IoT-пристрою з застосуванням заходів безпеки.

Термін служби вузла з алгоритмами шифрування з цієї групи становитиме 6–10 місяців залежно від розміру корисного навантаження.

Серед цієї групи найвищий результат показав шифр Piccolo на основі Фейстеля і є найвигіднішим вибором з точки зору енергоспоживання. Він дозволяє вузлу працювати близько 15 місяців без заміни батареї при розмірі пакета 50 байт. Piccolo вважається ультралегковаговим шифром, який можна використовувати навіть для RFID з розміром ключа 80 і 128 біт. Можливим напрямком подальших експериментів може бути дослідження переваг використання 80-бітного ключа (для застосувань, де це допустимо) з метою подальшого подовження терміну служби пристрою.

Термін служби IoT-пристрою відображає потенційний життєвий цикл усієї системи. Водночас безпека є одним із ключових елементів системи IoT, оскільки забезпечує конфіденційність та цілісність даних [39]. У роботі запропоновано модель для оцінки терміну служби IoT-пристрою з реалізованим механізмом безпеки з урахуванням використовуваного алгоритму шифрування та довжини пакета. Дослідження також провело глибокий аналіз легковагових блочних шифрів з точки зору їхнього впливу на термін служби та енергоспоживання, базуючись на специфікаціях шифрів та рівні безпеки.

Отримані результати щодо терміну служби можуть допомогти розробникам на етапі розгортання системи IoT та забезпечити кращий графік технічного обслуговування системи. Як показано в роботі, легковагові криптографічні алгоритми доцільно використовувати в певних моделях комунікації IoT, таких як пристрій-пристрій (device-to-device), пристрій-шлюз (device-to-gateway) та хмарна модель пристрій-хмара з живленням від хмари (cloud powered device-to-cloud). Водночас модель пристрій-хмара може також працювати з потужнішими та безпечнішими алгоритмами.

Для порівняльного аналізу різних алгоритмів ми використали платформу Arduino Mega. Було проведено порівняльне дослідження алгоритмів AES, XTEA, HIGHT, KLEIN, Twofish, Blowfish, PRESENT, Serpent та Piccolo з точки зору енергоспоживання та максимального терміну служби. Це дозволило

охарактеризувати енергоефективність цих рішень для подальшого застосування на кінцевих IoT-пристроях. У роботі також розглянуто властивості шифрів на основі мережі Фейстеля та структури SPN. Хоча шифри зі структурою SPN теоретично споживають менше енергії та працюють швидше, ніж шифри на основі Фейстеля, отримані експериментальні результати показали, що деякі SPN-алгоритми, зокрема Serpent і PRESENT, мають суттєвий час шифрування та розшифрування, тоді як Piccolo, XTEA та HIGHT працюють досить швидко серед шифрів Фейстеля.

Результати показали, що ECC-160 споживає максимальну потужність, тому він найменш бажаний для ресурс-обмежених пристроїв. Рекомендується використовувати цей шифр лише у випадках, коли безпека має дуже високий пріоритет, оскільки ECC-160 відповідає 1024-бітному ключу в RSA, тобто забезпечує високий рівень безпеки. Алгоритми Blowfish, Twofish, PRESENT та Serpent також характеризуються високим енергоспоживанням і меншим терміном служби, проте їх можна розглядати для впровадження в системах, де потрібен високий рівень безпеки. AES, Piccolo-128, XTEA та KLEIN є найбільш бажаними варіантами при поєднанні вимог до енергоспоживання, максимального терміну служби та середнього рівня безпеки [40]. Нарешті, ми дійшли висновку, що алгоритм Piccolo є найбільш ефективним і тому є найкращим вибором з точки зору енергоспоживання.

2.3 Методи оптимізації криптографічних алгоритмів для обмежених ресурсів

Запропонований метод являє собою комплексну систему безпеки, яка поєднує різні техніки захисту на основі шифрування. Ця концепція інтегрує симетричне та асиметричне шифрування з удосконаленою системою цифрового підпису, яка забезпечує цілісність даних за допомогою хеш-функції SHA-256. Загалом, запропонований підхід розроблено для забезпечення високоефективного та надійного захисту обчислювальних середовищ.

Важливим критерієм, який слід враховувати під час реалізації рішень на основі криптографічних алгоритмів, є обчислювальна швидкість алгоритмів і пропорційність між швидкістю та продуктивністю. Асиметричні алгоритми (з відкритим ключем) потребують значно більше часу на генерацію ключів [26]. Тому їх швидкість порівняно нижча, ніж у симетричних алгоритмів, таких як Blowfish.

Відповідно, доцільно шифрувати основне повідомлення для передачі за допомогою симетричного підходу, який забезпечує швидшу обробку. У цьому рішенні для шифрування вихідних даних використано удосконалений алгоритм Blowfish.

Загальну схему запропонованого рішення наведено на рис. 2.4. Спочатку вихідні дані хешуються за допомогою алгоритму SHA-256 для забезпечення цілісності даних. Після цього алгоритм шифрування на еліптичних кривих (EC) використовується для створення цифрового підпису та захисту хеш-коду SHA-256 і приватного ключа. EC обрано завдяки його високим характеристикам безпеки та малому розміру ключа, що зменшує час виконання алгоритму. Нарешті, удосконалений алгоритм Blowfish — симетричний алгоритм із швидким часом виконання — шифрує вихідні дані. Детальний опис кожного етапу наведено нижче.

Цифровий підпис. У цій роботі для забезпечення цілісності даних використано хеш-функцію разом із цифровим підписом. Запропоноване рішення застосовує хеш-функції для створення зведення повідомлення перед його підписанням, оскільки класична модель цифрового підпису є вразливою до атак і дозволяє зловмиснику підробити цифровий підпис шляхом зміни процесу перевірки та порушення роботи мережі [28].

Таким чином, за допомогою цього рішення зловмисник може створити лише підроблений цифровий підпис, який не відповідатиме вмісту, пов’язаному з виходом функції зведення повідомлення, і не зможе змінити вміст самого повідомлення. Отже, використання цифрових підписів є потужним інструментом захисту середовища IoT.

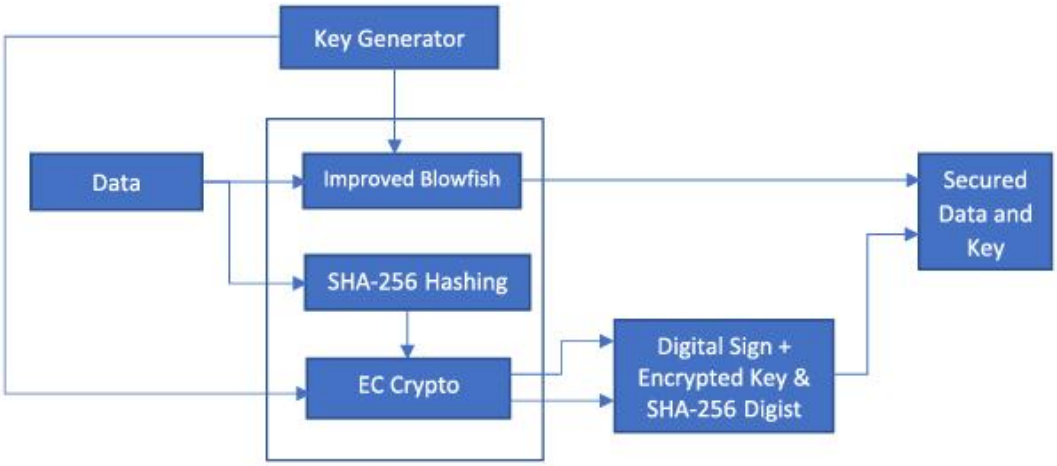

Рисунок 2.4 - Структура запропонованого методу.

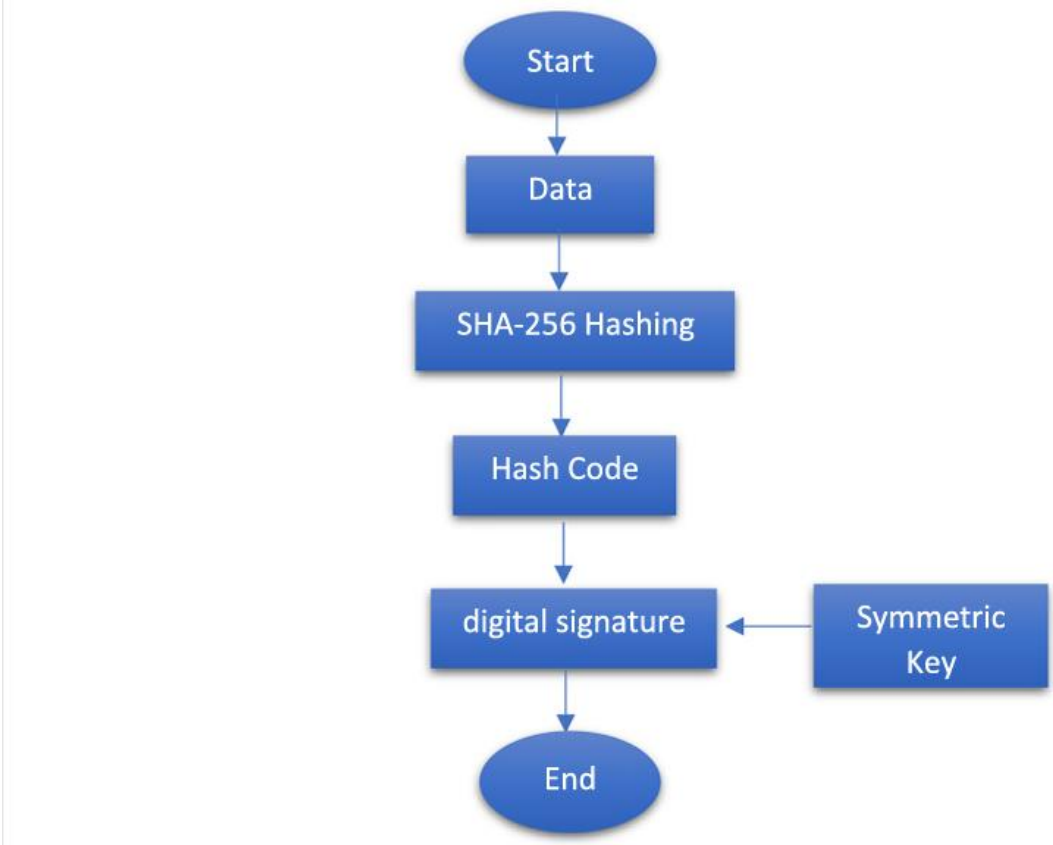

Рисунок 2.5 - Блок-схема цифрового підпису.

Схема роботи запропонованого рішення з використанням хеш-кодів і цифрових підписів зображена на рис. 2.4. Ця діаграма описує різні етапи забезпечення цілісності даних за допомогою хеш-функції SHA-256 та цифрових

підписів. Дотримуючись цієї схеми, користувачі можуть ефективно захистити свої дані та переконатися, що вони не були підроблені чи змінені будь-яким чином.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ АЛГОРИТМІЧНО-

ПРОГРАМНОГО РІШЕННЯ ШИФРУВАННЯ

3.1 Проєктування модуля шифрування даних для IoT-пристрою

Основною метою цього дослідження є представлення безпечного рішення шифрування, яке є ефективним і має низький час виконання. Для досягнення цієї мети модуль шифрування даних використовує вдосконалену версію алгоритму Blowfish. Алгоритм підтримує довжину ключа від 32 до 448 біт і потребує великої кількості підключів (subkeys). Ці підключі повинні бути згенеровані перед початком будь-якого шифрування або розшифрування даних.

Алгоритм Blowfish значно швидший і потребує менше пам’яті порівняно з іншими поширеними алгоритмами шифрування [29]. Він використовує 4 S-бокси, кожен з яких має 256 входів по 32 біти. Процес вибору повторюється для решти входів, де перший і другий байти 32-бітного входу використовуються для вибору входів у першому та другому S-боксах відповідно [30].

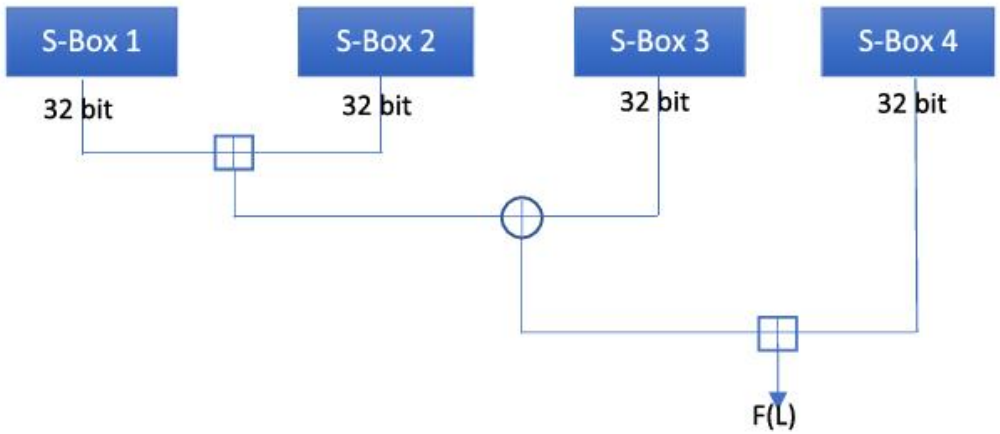

Рисунок 3.1 - Модуль F-функції в традиційному алгоритмі Blowfish.

Процес розшифрування подібний до шифрування, але використовує підключі у зворотному порядку. Оскільки функція F, яка включає операції додавання та зсуву, є основною операцією шифрування в усіх раундах

алгоритму Blowfish і виконується у вигляді модуля, вона займає найбільшу частину часу виконання цього алгоритму.

Тому це дослідження спрямоване на зменшення часу виконання алгоритму Blowfish шляхом модифікації модуля F. Загальний процес модуля F у традиційному алгоритмі Blowfish зображено на рис. 3.1.

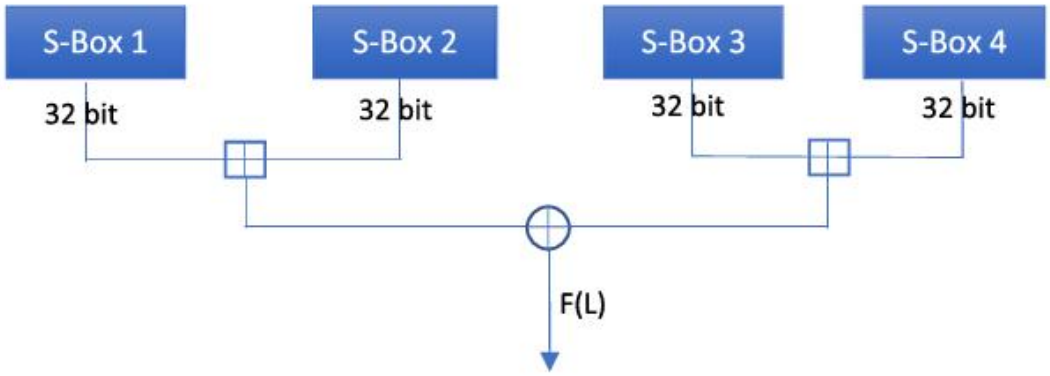

Рисунок 3.2 - Удосконалений модуль F-функції.

Оптимізація часу виконання алгоритму спрямована на зменшення його часової складності. Як видно з рис. 3.2, значення функції F у стандартному.

Застосування паралелізації дає змогу зменшити час виконання кількох операцій до часу виконання лише однієї операції. Оскільки алгоритм має 16 раундів, кожен процес шифрування та розшифрування виконується у 16 разів швидше.

Важливо зазначити, що безпека алгоритму Blowfish базується на міцності його ключів [32]. Модифікація, яка дозволяє виконувати дві операції додавання паралельно, не має негативного впливу на безпеку алгоритму. На рис. 3.2 також показано, як саме виконується ця модифікація.

Крім того, на рис. 3.3 наведено псевдокод процесу шифрування для удосконаленого алгоритму Blowfish, який використовує покращену функцію F.

```
Input data d
Split d into two 32-bit halves: dL, dR
For k = 1 to 16:
        dL = dL XOR Pi
        dR = F(dL) XOR dR
        Swap dL , dR
Next k
dL and dR
dR = dR XOR P17
dL = dL XOR P18
Recombine dL and dR
Function F(dL)
Split dL into four 8bit quarters: a, b, c and d
F(dL) = ((S1,a + S2,b mod 2^32) XOR (S3,c + S4,d mod 2^32))
```

Рисунок 3.3 - Псевдокод процесу шифрування на основі удосконаленого алгоритму Blowfish.

На першому етапі 64-бітний вхід ділиться на дві рівні 32-бітні частини (ліву та праву). Наступним кроком є застосування оператора XOR до першої 32-бітної частини блоку (L) та першого значення P. Потім отримані 32-бітні дані передаються до функції F. Після виконання відповідної операції результат знову XOR'ується з правою 32-бітною частиною (R), яка була отримана в результаті розбиття 64-бітного блоку.

Процес шифрування за допомогою удосконаленого алгоритму Blowfish включає початкову фазу налаштування ключів, після якої виконується кілька раундів шифрування та розшифрування з використанням модифікованої функції F. Під час кожного раунду 64-бітні дані діляться на два 32-бітні блоки L і R, до яких застосовуються різні операції, включаючи XOR, модульне додавання та заміщення через S-бокси. Після кожного раунду блоки L і R міняються місцями перед застосуванням наступного раунду.

Для розшифрування даних застосовується той самий процес у зворотному порядку з використанням ключів у зворотній послідовності.

Забезпечення безпеки симетричного ключа. Як було зазначено, для вирішення проблеми передачі приватного ключа в алгоритмі Blowfish у цьому дослідженні використовується асиметричне шифрування на основі еліптичних кривих. Криптографія на еліптичних кривих (ECC) є різновидом криптографії з відкритим ключем, яка використовує еліптичні криві над скінченними полями [33].

Однією з важливих можливостей цього алгоритму є забезпечення високого рівня безпеки за допомогою 164-бітного ключа, а також вища ефективність порівняно з попередніми методами завдяки меншому енергоспоживанню. Порівняно з іншими методами асиметричного шифрування ECC забезпечує найвищий рівень конфіденційності, низьке енергоспоживання та менші вимоги до пам’яті системи [35]. Еліптична крива описується рівнянням, наведеним на рис. 3.4:

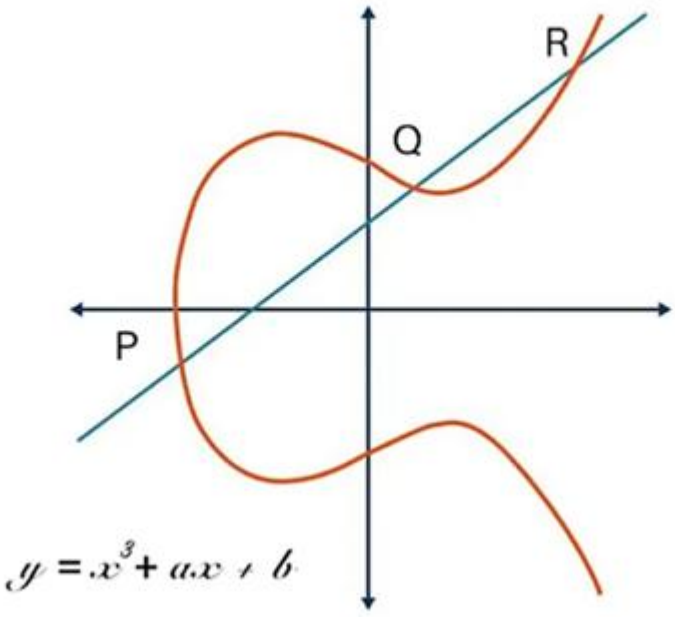

Рисунок 3.4 - Еліптичні криві.

Однією з найважливіших особливостей цього шифрування є виконання

операцій над скінченними полями. Криптографія на основі еліптичних кривих базується на розв’язанні задачі дискретного логарифма на еліптичній кривій, яка належить до NP-складних проблем.

Головною особливістю еліптичної кривої є те, що для знаходження третьої точки, яка лежить на кривій, можна визначити правило додавання двох точок для отримання третьої точки на кривій. Точки на еліптичній кривій і закон додавання цих точок утворюють скінченну абелеву групу.

Для того щоб операція додавання двох точок була коректно визначена, під час опису еліптичної кривої додається спеціальна точка на нескінченності (позначається як O), яка не лежить на самій кривій, але розглядається як точка. Ця точка служить елементом адитивної одиниці (additive identity), тобто додавання будь-якої точки до O повертає ту саму точку.

Кількість точок на еліптичній кривій називається її порядком (order) і включає точку на нескінченності [8].

Вибір фіксованої кривої. Для побудови криптографії на еліптичних кривих обирається велике просте число p, на основі якого створюється скінченне поле GF(p) як множина цілих чисел, менших за p.

Для шифрування на основі еліптичних кривих спочатку обирається випадкове число k у діапазоні [1,n−1]. Слід зазначити, що цей діапазон належить полю, і випадкове число в цьому діапазоні вважається приватним ключем. Далі обчислюється відкритий ключ R=kF, де F — точка на еліптичній кривій.

У цьому випадку обчислення k за відомими точками Q і F має експоненціальну часову складність. Для підвищення безпеки обирається фіксована крива і точка таким чином, щоб порядок фіксованої точки F був великим простим числом. Це значення визначається за допомогою алгоритму Шуфа (Schoff’s algorithm) на основі порядку кривої.

Використання криптографії на еліптичних кривих забезпечує значно менші розміри відкритих ключів і цифрових підписів порівняно з іншими асиметричними алгоритмами, такими як RSA, при збереженні того самого рівня

безпеки. Це пояснюється тим, що якщо порядок фіксованої точки F є простим числом розміром n біт, то обчислення k за відомими kF і F потребує приблизно операцій.

Шифрування за допомогою ECC. Припустимо, що троє користувачів — Alice, Bob, Cathy та David — узгодили еліптичну криву (не конфіденційну) та фіксовану точку кривої F (також не конфіденційну).

Щоб надіслати повідомлення Bob, Alice спочатку повинна отримати відкритий ключ Bob. Отримавши його, вона може зашифрувати повідомлення за допомогою відкритого ключа Bob і надіслати шифротекст Bob.

У криптографії на еліптичних кривих відкритий ключ Bob є точкою кривої , що лежить на тій самій кривій, яку узгодили Alice і Bob. Bob також обрав секретне ціле число Bk , і його відкритий ключ обчислюється як Bp=BkF , де F — фіксована точка на кривій.

Безпека методу базується на припущенні, що обчислення k за відомими F і kF є надзвичайно складним завданням. Оскільки операції в цьому методі виконуються над полем, використання алгоритмів масштабного множення (scaled multiplication) може значно підвищити ефективність шифрування на основі еліптичних кривих.

Відповідно, загальні етапи методу такі:

- Крок 1: Алгоритм хешування MD-5 використовується для створення хеш-коду вихідного матеріалу перед його шифруванням. Це робиться для підвищення ефективності цифрового підпису та перевірки цілісності даних.
- Крок 2: За допомогою цифрового підпису даних та зашифрованого приватного ключа створюється хеш-код, отриманий на попередньому етапі.
- Крок 3: За допомогою шифрування на еліптичних кривих зашифровується приватний ключ алгоритму Blowfish.
- Крок 4: Перед передачею даних цільова інформація шифрується за допомогою симетричного алгоритму шифрування (Blowfish). На цьому етапі симетричний ключ використовується для операції шифрування та генерації

зашифрованого рядка даних із вихідних даних. Після цього зашифровані дані надсилаються разом із зашифрованими рядками, отриманими на попередньому кроці.

- Крок 5: На стороні отримувача застосовується зворотний процес. У цьому випадку отримана інформація розшифровується за допомогою приватного ключа.
- Крок 6: Приватний ключ алгоритму Blowfish використовується для декодування вихідних даних.

Після цього виконується процес перевірки та валідування за допомогою хеш-функції (цифровий підпис).

Оцінка. Реалізацію запропонованого рішення виконано в середовищі розробки Eclipse з використанням Java Development Kit версії 7. У Java Development Kit функції шифрування доступні завдяки двом основним бібліотекам: JCA та JCE. Функції JCA повністю інтегровані з основними API Java і забезпечують більшість базових криптографічних можливостей. JCE використовується для виконання розширених операцій шифрування.

Апаратне забезпечення, що застосовувалося, мало процесор Intel Core 2 Duo з тактовою частотою 2,5 ГГц. Усі оцінки проведено в середовищі Windows.

Для оцінки запропонований метод порівнювали з криптографічними алгоритмами AES, 3DES, DES, RSA, PRESENT та ECDH [19, 20] за штучними критеріями: часом виконання, пропускною здатністю (throughput) та обсягом спожитої пам’яті.

Спочатку рішення оцінювали на невеликому обсязі даних (50 КБ), щоб перевірити їхню роботу при малому розмірі даних. Далі порівняння та оцінку проведено для обсягів даних 1024 КБ (1 МБ) та 2048 КБ (2 МБ).

Критерії оцінки. Для вимірювання ефективності рішення необхідний набір критеріїв, за якими можна оцінити його продуктивність. Для цього було обрано такі критерії:

Час виконання: це час, необхідний для виконання операцій шифрування. У наведеному співвідношенні час виконання визначається за допомогою

системної мітки часу (System timestamp) — таймера в мові Java, який запускається на початку операції шифрування та зупиняється після її завершення. Різниця між цими двома значеннями визначає час реалізації рішення.

Пропускна здатність (Throughput): це кількість задач, оброблених і зашифрованих за одиницю часу. У цьому співвідношенні пропускна здатність отримується шляхом ділення оцінюваного обсягу даних на час виконання, розрахований за попереднім співвідношенням.

Результати та обговорення. Перша оцінка базується на обсязі даних 50 КБ, як показано на рис. 3.5. Час виконання порівняно з алгоритмами RSA, AES, DES, 3DES та PRESENT зменшився на 800, 550, 300, 400 та 800 мілісекунд відповідно. Це означає, що операції шифрування виконуються за менший час. Таке покращення є результатом зменшення часу виконання завдяки розробці та використанню алгоритму Blowfish для шифрування основних даних. Blowfish належить до однієї з найшвидших криптографічних методів у категорії блочних шифрів. Крім того, завдяки зміні в модулі F-функції його швидкість і ефективність значно зросли.

Оскільки асиметричний алгоритм EC у запропонованому рішенні використовується лише для шифрування ключа та зведення хеш-коду, він не мав суттєвого впливу на загальний час виконання.

На рис. 3.6 рішення порівнюються за пропускною здатністю при обсязі даних 50 КБ. Як видно з оцінки, запропонований метод демонструє вищу пропускну здатність порівняно з іншими рішеннями. Відповідно, порівняно з алгоритмами RSA, AES, 3DES, DES та ECDH оптимальність досягла 100 %. Зважаючи на дуже низький час виконання цього рішення під час оцінки, такого рівня пропускної здатності можна було очікувати.

Оскільки розмір даних, що шифруються, може впливати на виконання операції, у другому експерименті обсяг даних збільшили до 1 мегабайта (1024 КБ), щоб порівняти ефективність алгоритмів при цьому обсязі.

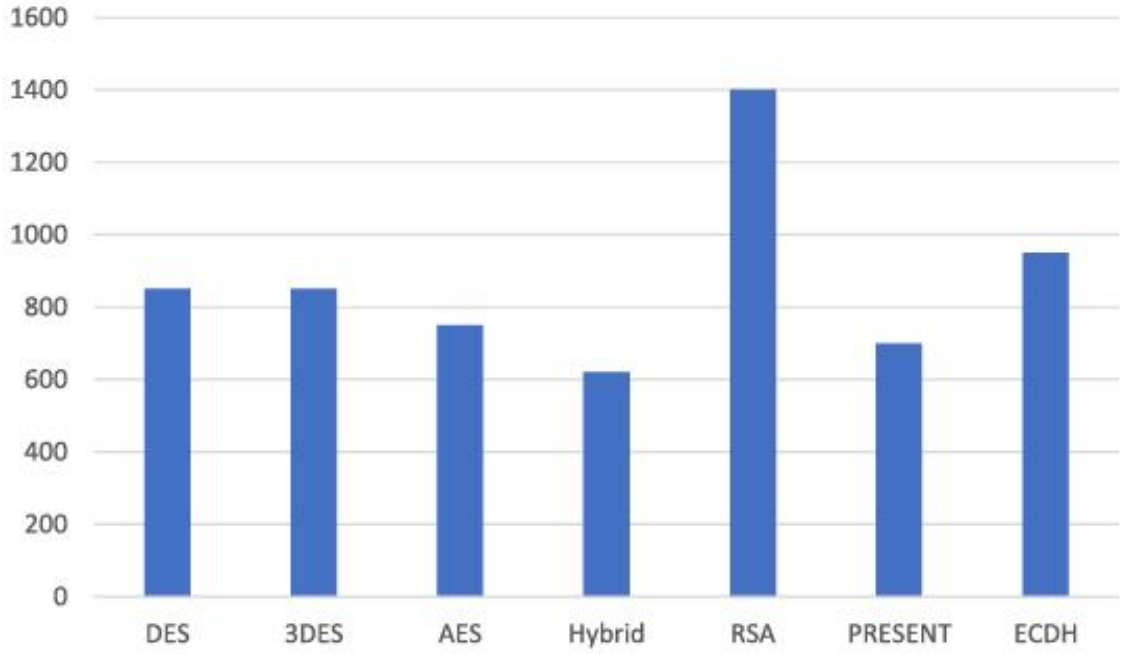

Рисунок 3.5 - Час виконання — мілісекунди (обсяг даних: 50 КБ)

Як видно на рис. 3.5, запропоноване рішення показало кращі результати порівняно з іншими алгоритмами. Час виконання значно зменшився — на 240, 240, 130, 80 та 330 мс порівняно з алгоритмами DES, 3DES, AES, PRESENT та ECDH відповідно. Це свідчить про те, що запропоноване рішення працює швидше за ці алгоритми. Порівняно з асиметричним алгоритмом RSA час виконання зменшився більш ніж на 731 мс, що вказує на швидшу реалізацію рішення.

На рис. 3.6 також показано рівень пропускної здатності рішень. Завдяки нижчому часу виконання передбачувано, що пропускна здатність буде вищою. Відповідно, пропускна здатність у середньому зросла більш ніж на 16 % порівняно з AES, 3DES та DES і більш ніж на 50 % порівняно з асиметричним алгоритмом RSA.

У запропонованому рішенні найбільший час виконання припадає на шифрування основних даних, яке виконується запропонованим алгоритмом. Завдяки паралелізації та вдосконаленню цієї частини рішення значно зменшився час виконання і, як наслідок, зросла пропускна здатність.

Нарешті, в останньому тесті алгоритми оцінювалися при обсязі даних

2048 КБ (2 МБ).

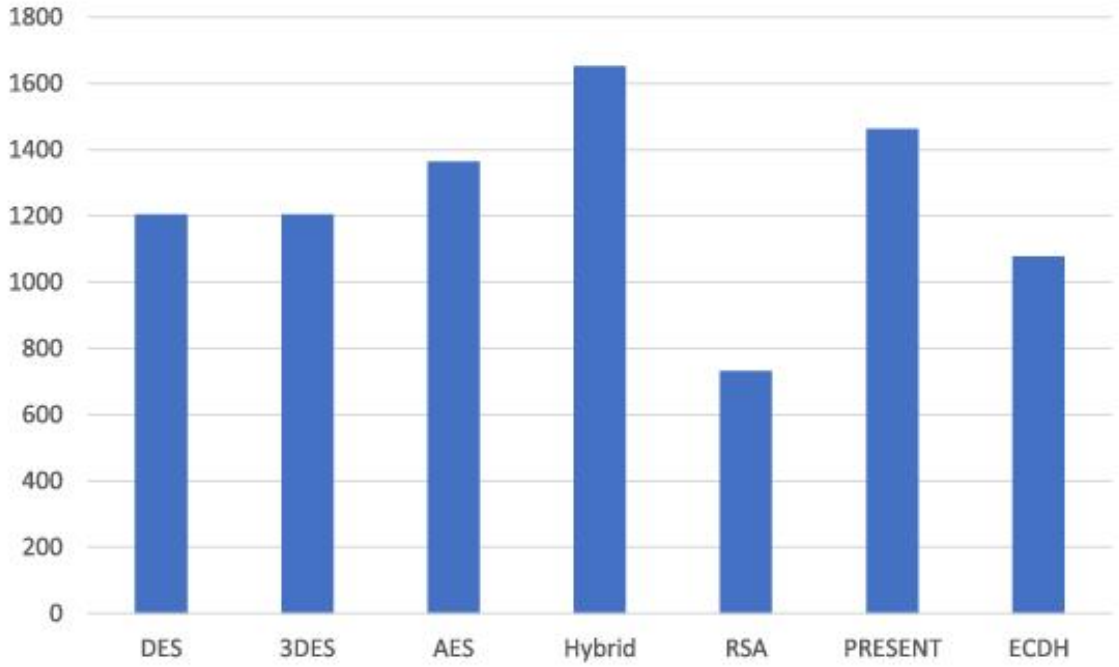

Рисунок 3.6 - Пропускна здатність — кілобайт на секунду (обсяг даних: 50 кілобайт)

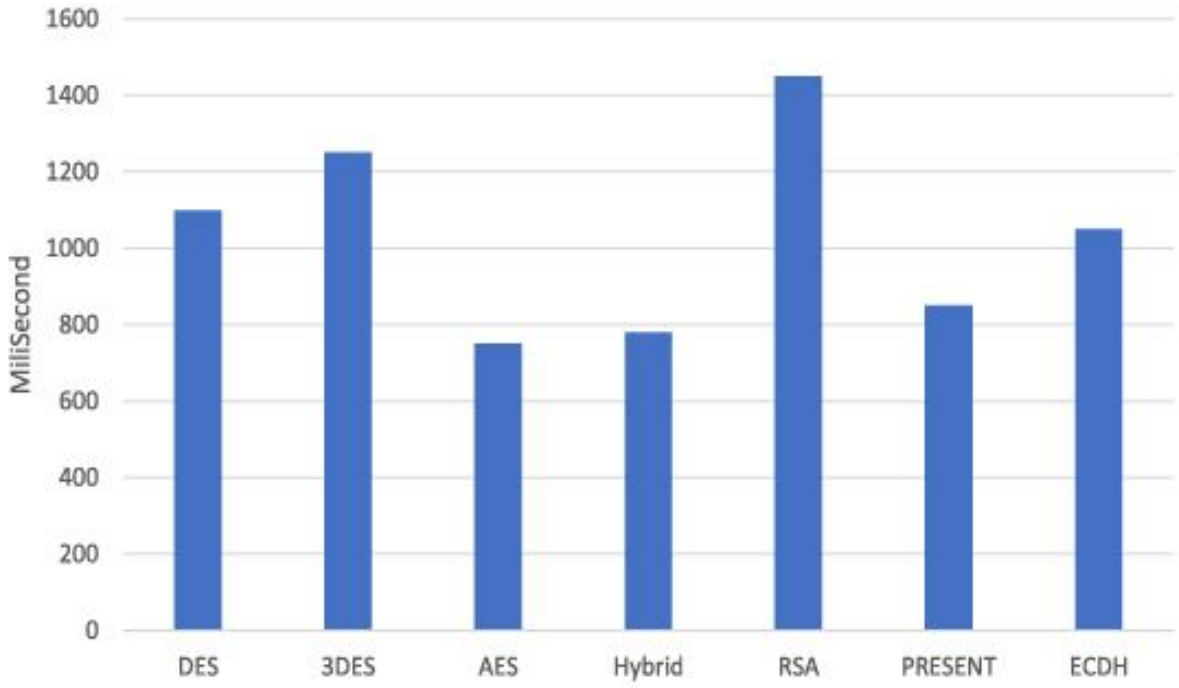

Рисунок 3.7 - Час виконання — мілісекунди (обсяг даних: 1024 КБ)

На рисунках 3.7 і 3.8 чітко видно, що запропоноване рішення є більш

оптимальним. Як видно, зі збільшенням обсягу даних зростає тривалість шифрування, що потребує більше часу. Однак у цій оцінці запропоноване рішення змогло виконати операцію за менший час порівняно з алгоритмами DES, 3DES та RSA. Час виконання порівняно з двома симетричними алгоритмами 3DES і DES зменшився відповідно на 470 та 320 мс, а пропускна здатність стала значно кращою (див. рис. 3.7 і 3.8).

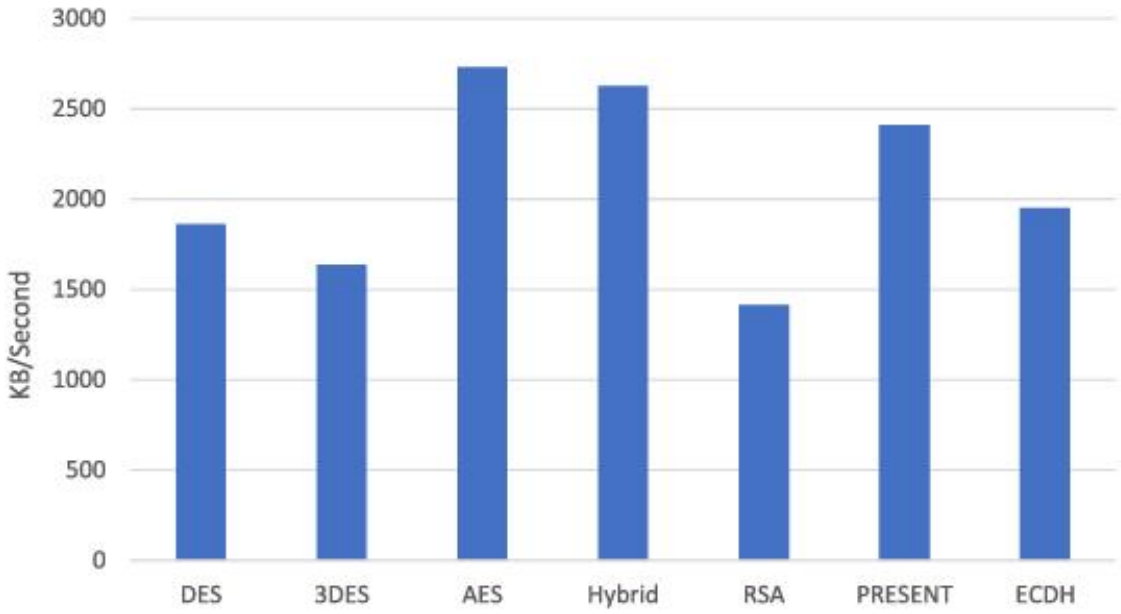

Рисунок 3.8 - Пропускна здатність — кілобайт/секунду (обсяг даних: 1024 кілобайт)

У випадку асиметричного алгоритму RSA час виконання зменшився більш ніж на 670 мс. Однак у цьому експерименті порівняно з симетричним алгоритмом AES оптимізації не досягнуто. Зважаючи на те, що запропоноване рішення пропонує додаткові можливості, такі як цифровий підпис і хешування, порівняно зі стандартним алгоритмом AES, це незначне збільшення часу виконання можна вважати несуттєвим і таким, що не матиме значного впливу на ефективність інфраструктури IoT.

Як видно на рис. 3.8, зі збільшенням розміру даних пропускна здатність рішення також зросла завдяки змінам у F-функції та паралелізації, виконаній у

ядрі шифрування Blowfish. Його пропускна здатність може бути значно вищою, ніж у інших алгоритмів.

Крім того, оскільки в запропонованому методі шифрування даних відділено від шифрування ключів, одночасне використання Blowfish і EC не призвело до суттєвих втрат продуктивності та збільшення часу виконання. У цьому випадку лише приватний ключ і хеш-код шифруються за допомогою EC, і через їхній дуже малий обсяг вони мають мінімальний вплив на час виконання. Основні дані, обсяг яких є значним, шифруються за допомогою вдосконаленого алгоритму Blowfish, який завдяки своїй високій ефективності, часу виконання та пропускній здатності значно перевершує алгоритми 3DES, DES і RSA.

Споживання пам’яті. Кількість пам’яті, яку споживають алгоритми шифрування, також вважається важливою характеристикою. У цьому розділі проведено оцінку та порівняльний аналіз обсягу спожитої пам’яті запропонованим методом із іншими поширеними рішеннями шифрування.

Результати оцінки наведено на рис. 3.8. У цьому тесті для оцінки використано обсяг даних 50 КБ.

Як видно з рис. 3.8 і таблиці 3.1, запропонований метод має найнижчий рівень споживання пам’яті серед усіх розглянутих рішень. Причиною такої високої ефективності є оптимізація криптографічного ядра на основі вдосконаленого алгоритму Blowfish. Це підвищило пропускну здатність і зменшило споживання пам’яті, оскільки в звичайних умовах алгоритм Blowfish є одним із найбільш оптимальних за обсягом займання пам’яті та часом виконання. Крім того, алгоритм EC є одним із найбільш ефективних асиметричних алгоритмів щодо споживання пам’яті завдяки використанню 164-бітного ключа.

Такий низький обсяг вимог до пам’яті не створює жодних проблем для використання запропонованого методу в старих інфраструктурах, а також дозволяє застосовувати його навіть у найменших вбудованих системах.

Таблиця 3.1

Споживання пам’яті в порівнюваних методах

Алгоритм	**Споживання пам’яті (одиниці)**
DES	18,2
3DES	20,7
AES	14,7
RSA	31,0
Hybrid (запропонований)	10,0
PRESENT	19,0
ECDH	15,0

Таблиця 3.2

Криптоаналіз ECC, що ілюструє складність атак

Алгоритм атаки	**Часова складність**	**Просторова складність**
Brute-Force	$O(n) = O(2^{192})$	O(1)
Baby-Step, Giant-Step	$O(\sqrt{n})$	O(1)
Pollard’s Rho	$O(\sqrt{n}) = O\left(2^{\frac{192}{2}}\right)$	O(1)

Таблиця 3.2 узагальнює стійкість різних алгоритмів. За умови, що розмір ключа дорівнює N, у таблиці показано, якою є стійкість алгоритму до злому. Порівняння базуються на часовій складності, необхідній для проведення успішної атаки. Рядок у таблиці ілюструє необхідну довжину ключа для еквівалентного рівня безпеки. Фактичні розміри ключів, що використовуються: 128, 192, 256 біт для AES та 80, 128 біт для Hybrid і PRESENT.

Хоча запропонований метод використовує алгоритм ECC для безпечного обміну приватним ключем, таблиця 3.2 представляє криптоаналіз ECC, що демонструє складність алгоритмів атак [21]. Як видно з таблиці 3.2, часова складність ECC є дуже високою, що робить цей алгоритм та рішення на його основі захищеними від різноманітних атак, зокрема Brute-Force, Baby-Step, Giant-Step та Pollard's Rho.

3.2 Реалізація та оптимізація алгоритмів шифрування

Для підвищення безпеки в IoT-додатках застосовуються різні підходи, такі як контроль приватності та автентифікація. Однак ці підходи завжди мають загрози у вигляді атак. Для посилення рівня безпеки алгоритми шифрування та розшифрування значною мірою вирішують проблему захисту в широкому спектрі IoT-додатків.

До найпоширеніших алгоритмів шифрування, що реалізуються на практиці, належать AES, DES, 3DES та Blowfish.

AES (Advanced Encryption Standard) є блочним шифром на основі раундів, який підтримує розміри ключів 128 біт, 192 біти та 256 біт. Цим ключам відповідає 10, 12 і 14 раундів відповідно. Алгоритм AES було стандартизовано у 2001 році Національним інститутом стандартів і технологій (NIST) під назвою FIPS-197 та включено до стандарту ISO/IEC 18033-3 [19].

У будь-якому комунікаційному додатку цей алгоритм працює у фоновому режимі, де блочний шифр виконується в режимах, таких як Electronic Codebook (ECB), Cipher Feedback (CFB) та Counter (CTR). Однак, за даними літератури, режим CFB є кращим за швидкістю виконання, оскільки потребує шифрування/розшифрування лише в одному напрямку [20].

DES (Data Encryption Standard) є ще одним стандартом шифрування, рекомендованим NIST. DES використовує симетричний блочний шифр, заснований на алгоритмі Lucifer, запропонованому компанією IBM [21].

3DES (Triple DES) є модифікацією DES. Це також симетричний блочний шифр, але дана техніка шифрування застосовує алгоритм DES тричі до кожного блоку даних. Завдяки триразовому застосуванню шифрування 3DES значно підвищує рівень безпеки [21].

Blowfish — це симетричний блочний шифр із змінною довжиною ключа від 32 до 448 біт.

Системний дизайн та експеримент. Цей розділ містить опис апаратного дизайну з короткою характеристикою основних апаратних компонентів і програмного забезпечення, що використовувалося, а також експериментальні результати всіх технік.

Програмні та апаратні інструменти

1. Raspberry Pi 3B+ Raspberry Pi 3B+ — це компактний електронний пристрій із продуктивністю чотириядерного процесора з тактовою частотою 1,4 ГГц. Він підтримує двосмугову бездротову мережу LAN (2,4 ГГц), швидкісний Ethernet та Bluetooth 4.2/BLE.

2. Raspbian OS Raspbian — це вільна операційна система на базі Debian GNU/Linux, розроблена для підтримки апаратного забезпечення Raspberry Pi, забезпечення підключення, комунікації та логічної обробки даних. Raspbian OS містить велику кількість пакетів (понад 35 000) і правильно відформатоване попередньо скомпільоване програмне забезпечення, що значно спрощує встановлення системи на Raspberry Pi.

3. Датчик DHT22 У нашому експерименті датчик DHT22 використовується для вимірювання вологості та температури. Датчик генерує цифрові сигнали, які безпосередньо зчитуються Raspberry Pi.

Експериментальне налаштування

Схема підключення електронних компонентів показана на рис. 3.9, де контакт 1 датчика DHT22 підключено до джерела живлення 3,3 В, контакт 4 — до землі (GND), а контакт 7 — до контакту загального призначення (GPIO) на Raspberry Pi. Крім того, схема містить 10 кОм резистор, підключений між контактами 1 і 7.

Для мережевої комунікації використано другий пристрій Raspberry Pi, який у експерименті виступає як сервер / точка доступу (див. рис. 3.10). Для роботи серверної частини на Raspberry Pi було встановлено Apache та PhpMyAdmin.

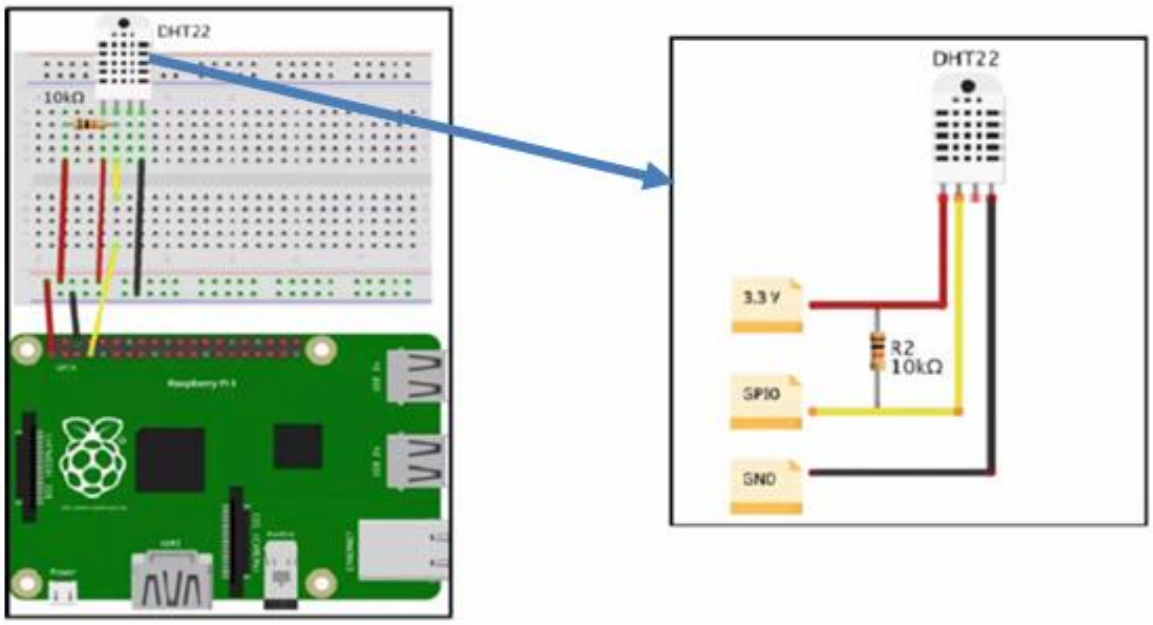

Рисунок 3.9 - Схема підключення компонентів

Рисунок 3.10 - Схема мережевої комунікації

Крім того, Raspberry Pi працював як бездротова точка доступу (Wireless Access Point — WAP) у мережі для короткодіапазонної комунікації. На серверній стороні для обробки кількох TCP/IP-з'єднань використовувався Python-скрипт.

3.3 Експериментальні дослідження та аналіз результатів

Експерименти проводилися для різних алгоритмів шифрування. Перше виконання було проігноровано для всіх алгоритмів шифрування, щоб уникнути можливих помилок налаштування або початкового зміщення (initial bias).

Для алгоритму AES експеримент проводився з розміром блоку даних 320 біт (40 байтів) та трьома різними довжинами ключів: 128 біт, 192 біти та 256 біт. Було оцінено час обробки для кожної довжини ключа, який включав час генерації ключа, час шифрування на стороні датчика та час розшифрування на стороні сервера на Raspberry Pi. Середній час було розраховано шляхом виконання експерименту 10 разів, як показано на рис. 3.11.

З графіка середнього часу виконання AES (рис. 3.11) видно, що генерація ключа потребує приблизно в 6 разів більше часу, ніж шифрування. У всій послідовності захищеної комунікації найменший час виконання має процес розшифрування.

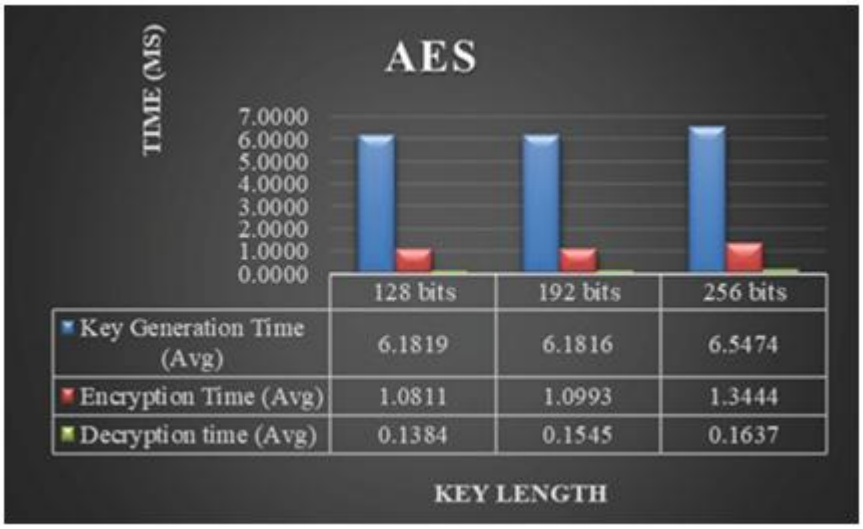

Рисунок 3.11 - Графік середнього часу виконання AES для 10 експериментів

Для DES було застосовано той самий підхід, що й для AES. Крім того, було оцінено продуктивність DES з довжиною ключа 64 біти, як показано на

рис. 3.11. З рис. 3.11 видно, що час генерації ключа є вищим порівняно з часом шифрування та розшифрування, проте різниця не така значна, як у випадку AES.

Рисунок 3.12 - Графік середнього часу виконання DES для 10 експериментів

Для 3DES результати експерименту з довжинами ключів 128 біт і 192 біти представлено на рис. 3.12 з тим самим розміром блоку даних, що й для алгоритмів AES та DES. Тут, серед трьох етапів виконання, найбільший час також припадає на генерацію ключа. Загалом, весь процес шифрування в 3DES відбувається швидше, ніж в AES.

Для алгоритму Blowfish проведено аналогічний експеримент з розміром блоку даних 320 біт (40 байтів) та довжинами ключів 128 біт, 192 біти та 256 біт, як показано на рис. 3.13. З рис. 3.13 видно, що час генерації ключа значно вищий при використанні ключа довжиною 256 біт. Час генерації ключа в Blowfish подібний до AES, проте для інших довжин ключів він показує менші значення часу.

З рис. 3.14 також видно, що запропонований підхід працює швидше, ніж Blowfish при використанні 256-бітного ключа, і швидше, ніж AES для всіх довжин ключів.

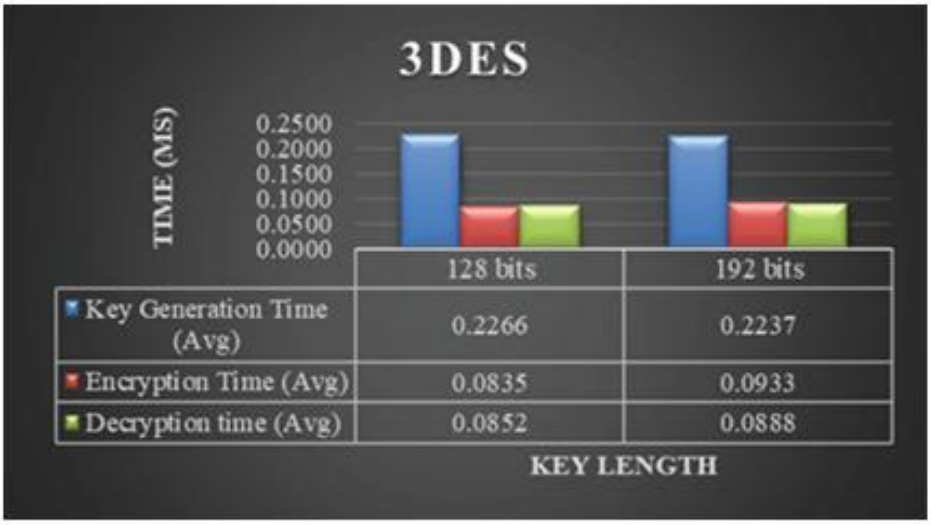

Рисунок 3.13 - Графік середнього часу виконання 3DES для 10 експериментів

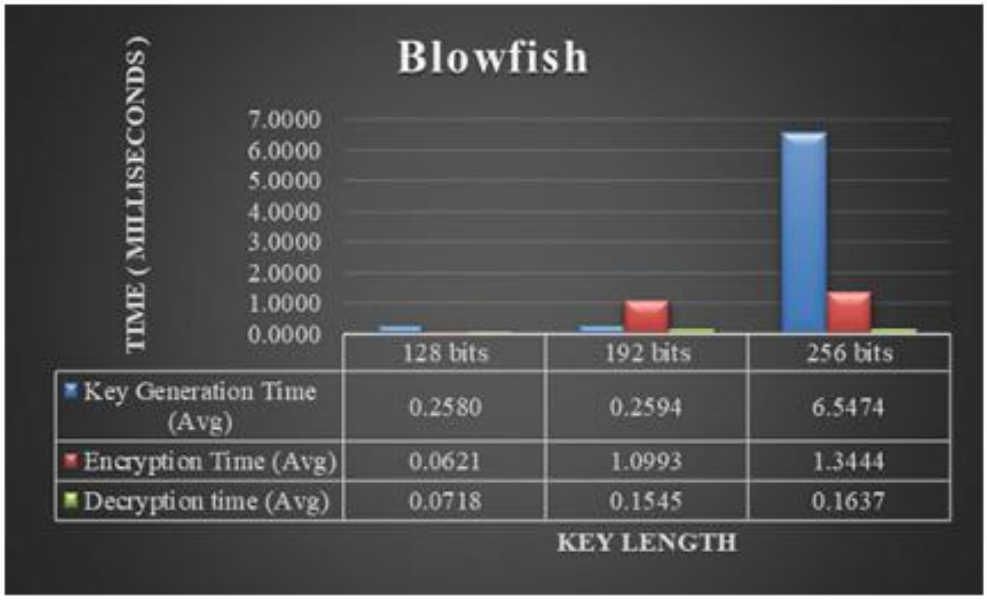

Рисунок 3.14 - Графік середнього часу виконання Blowfish для 10 експериментів

Для порівняльного аналізу алгоритмів шифрування ми використовували такі показники продуктивності, як «Час генерації ключа», «Час шифрування» та «Час розшифрування», що наведено на рис. 3.11, 3.12, 3.13 та 3.14. Середні значення часу після 10 виконань для кожного алгоритму показують, що алгоритм AES має максимальні вимоги до часу, тоді як DES і 3DES демонструють нижчі значення часу для всього процесу шифрування. Алгоритм

Blowfish має нижчі вимоги до часу при використанні коротших ключів (128 біт і 192 біти), проте при довжині ключа 256 біт час виконання значно зростає. У нашому експерименті алгоритм Blowfish не є оптимальним при застосуванні 256-бітного ключа.

Алгоритм DES показав, що загальний час, необхідний для повного процесу шифрування, є нижчим, ніж у AES і Blowfish. Часові вимоги для 3DES є порівнянними з DES. У всіх цих алгоритмах при детальному аналізі окремих етапів було встановлено, що «Час генерації ключа» є найбільшим порівняно з «Часом шифрування» та «Часом розшифрування». Серед трьох етапів найменші витрати часу припадають на «Час розшифрування».

Експеримент також виявив тенденцію до збільшення часу всього процесу шифрування зі зростанням довжини ключа.

Для розрахунку шифротекст генерувався з відкритого тексту за допомогою відповідних алгоритмів шифрування. Пропускну здатність різних алгоритмів зображено на рис. 3.15.

Було встановлено, що алгоритм AES має найнижчу пропускну здатність, тоді як алгоритм Blowfish демонструє найвищу пропускну здатність.

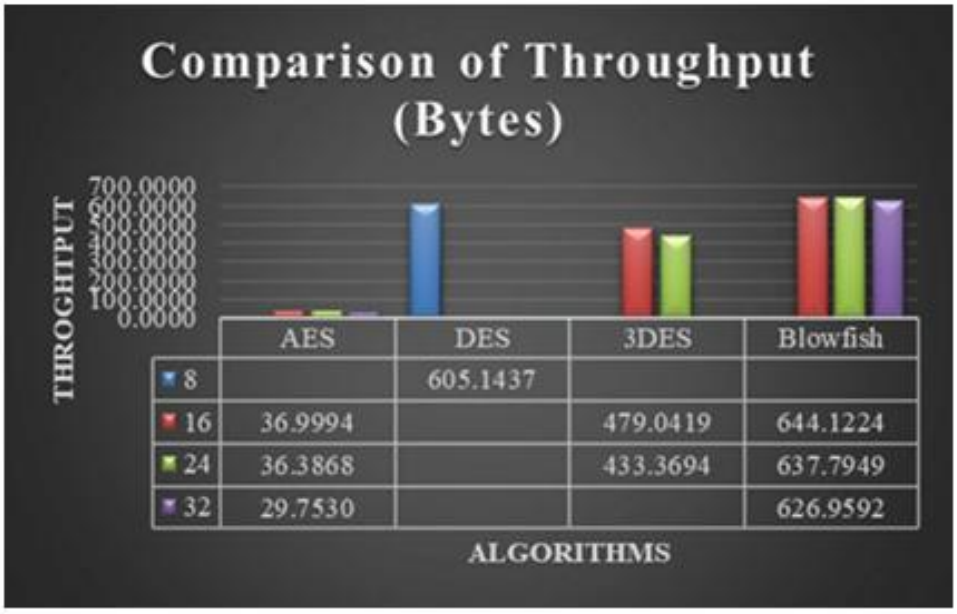

Рисунок 3.15 - Графік порівняння пропускної здатності різних алгоритмів шифрування

Висновки та напрямки подальших досліджень. Система Інтернету речей (IoT) досі потребує надійного механізму захисту даних від потенційних загроз. Для підвищення рівня безпеки застосовуються різні алгоритми шифрування. Однак вибір оптимального алгоритму завжди є складним завданням, оскільки існує компроміс між різними показниками продуктивності системи. Тому виникає суттєва необхідність у збалансуванні цих показників.

У цьому дослідженні проведено порівняльний аналіз алгоритмів шифрування AES, DES, 3DES та Blowfish за такими показниками продуктивності системи, як час виконання процесу шифрування та пропускна здатність (throughput). Аналіз показав наявність компромісу між цими алгоритмами: в одних аспектах кращим є DES, тоді як в інших — Blowfish є кращою альтернативою.

У подальших дослідженнях планується провести більш детальний і критичний аналіз цих алгоритмів з точки зору енергоспоживання та використання пам’яті при різних розмірах даних і довжинах ключів.

ВИСНОВКИ

У ході виконання бакалаврської роботи було розроблено алгоритмічно-програмні рішення шифрування даних для IoT-пристроїв з обмеженими обчислювальними ресурсами, що є комплексним завданням, спрямованим на забезпечення надійного захисту інформації в умовах обмеженої продуктивності, пам’яті та енергоспоживання. У роботі проведено аналіз предметної області, досліджено сучасні підходи до забезпечення безпеки в IoT-системах, а також визначено ключові вимоги до криптографічних алгоритмів для використання в ресурсно-обмежених середовищах.

На початковому етапі було виконано аналіз існуючих криптографічних методів та рішень, що застосовуються в IoT. Виявлено, що традиційні алгоритми шифрування, такі як AES або RSA, хоча й забезпечують високий рівень безпеки, не завжди є оптимальними для мікроконтролерних систем через значні витрати обчислювальних ресурсів. Це обумовило необхідність дослідження та застосування легковагових (lightweight) криптографічних алгоритмів, адаптованих до обмежених умов функціонування IoT-пристроїв.

У другому розділі було розглянуто методи, моделі та алгоритми шифрування, а також проведено оцінку їх ефективності з точки зору обчислювальної складності та енергоспоживання. Особливу увагу приділено оптимізації криптографічних процесів, що дозволяє зменшити навантаження на процесор та знизити споживання енергії без суттєвого зниження рівня безпеки.

У процесі розробки було спроєктовано та реалізовано програмний модуль шифрування даних, орієнтований на використання в IoT-пристроях. Реалізація враховує обмеження апаратних ресурсів та забезпечує ефективну обробку даних у реальному часі. Було використано оптимізовані алгоритмічні підходи, що дозволили досягти балансу між швидкодією, енергоспоживанням і криптографічною стійкістю.

Проведене експериментальне дослідження дозволило оцінити ефективність запропонованих рішень. Результати показали, що розроблений модуль шифрування забезпечує достатній рівень захисту даних при значно

менших витратах ресурсів у порівнянні з класичними криптографічними алгоритмами. Також було підтверджено можливість його використання в умовах обмежених обчислювальних можливостей без суттєвого впливу на продуктивність системи.

Загалом, розроблене рішення відповідає поставленим цілям і завданням. Воно забезпечує ефективне шифрування даних для IoT-пристроїв, має низькі вимоги до апаратних ресурсів та може бути інтегроване в різні IoT-системи. Перевагами запропонованого підходу є зменшене енергоспоживання, висока швидкодія та адаптивність до різних типів пристроїв.

Разом із тим, існують певні обмеження, пов’язані з необхідністю подальшого підвищення рівня криптографічної стійкості та захисту від сучасних атак, таких як побічні канали (side-channel attacks). У майбутньому доцільним є розширення функціональності системи, зокрема впровадження механізмів управління ключами, інтеграції з хмарними сервісами, а також використання апаратного прискорення криптографічних операцій.

Отже, результати роботи можуть бути використані при розробці захищених IoT-систем, що функціонують в умовах обмежених ресурсів, та сприятимуть підвищенню рівня інформаційної безпеки сучасних вбудованих пристроїв.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Akyildiz, I. F., & Jornet, J. M. (2023). Internet of Nano-Things. IEEE Communications Magazine.
2. Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, F. (2024). Internet of Things security: A survey. Journal of Network and Computer Applications.
3. Biryukov, A., Perrin, L., & Udovenko, A. (2023). Lightweight cryptography for IoT. IEEE Security & Privacy.
4. Bogdanov, A., et al. (2022). PRESENT: An Ultra-Lightweight Block Cipher. CHES Conference Proceedings.
5. Cao, Y., et al. (2024). Lightweight cryptography for IoT devices: A survey. ACM Computing Surveys.
6. Chen, J., & Li, X. (2023). Efficient encryption algorithms for resource-constrained IoT devices. IEEE Access.
7. Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE Transactions on Information Theory.
8. Dworkin, M. (2023). Recommendation for Block Cipher Modes of Operation. NIST.
9. Eisenbarth, T., & Kumar, S. (2022). A survey of lightweight cryptography implementations. IEEE Design & Test.
10. Farash, M. S., et al. (2023). Lightweight authentication schemes for IoT. Journal of Information Security.
11. Gubbi, J., et al. (2022). Internet of Things (IoT): A vision, architectural elements. Future Generation Computer Systems.
12. Gupta, B., & Quamara, M. (2023). Security in Internet of Things. Procedia Computer Science.
13. Hatzivasilis, G. (2023). Lightweight cryptography for embedded systems. Journal of Cryptographic Engineering.
14. Hummen, R., et al. (2022). Delegation-based authentication and authorization for IoT. IEEE SECON.

15. Hussain, F., et al. (2024). Lightweight encryption techniques for IoT. IEEE Internet of Things Journal.

16. ISO/IEC 29192-2 (2022). Lightweight cryptography — Block ciphers. ISO Standard.

17. ISO/IEC 29167 (2023). Cryptographic suites for RFID security. ISO Standard.

18. Jing, Q., et al. (2023). Security of IoT systems: A survey. IEEE Communications Surveys & Tutorials.

19. Katz, J., & Lindell, Y. (2020). Introduction to Modern Cryptography. CRC Press.

20. Kumar, P., & Lee, H. J. (2023). Security issues in IoT. Future Generation Computer Systems.

21. Li, S., Xu, L. D., & Zhao, S. (2022). The Internet of Things: A survey. Information Systems Frontiers.

22. Lopez, J., et al. (2023). Lightweight cryptography in IoT: Challenges and solutions. Sensors Journal.

23. Mahmoud, R., et al. (2023). Internet of Things security: Current challenges. IEEE IoT Journal.

24. McKay, K., et al. (2024). NIST Lightweight Cryptography Finalists. NIST Report.

25. Menezes, A., van Oorschot, P., & Vanstone, S. (2018). Handbook of Applied Cryptography. CRC Press.

26. Naor, M., & Yung, M. (2021). Public-key cryptosystems. Journal of Cryptology.

27. NIST (2024). Lightweight Cryptography Standardization Process. https://csrc.nist.gov

28. Perrig, A., et al. (2022). SPINS: Security protocols for sensor networks. Wireless Networks.

29. Ray, P. P. (2023). A survey on IoT cloud platforms. Future Computing and Informatics Journal.

30. Roman, R., Zhou, J., & Lopez, J. (2022). On the features and challenges of IoT security. Computer Networks.

31. Raza, S., et al. (2023). SVELTE: Real-time intrusion detection in IoT. Ad Hoc Networks.

32. Rivest, R. (1992). The MD5 message-digest algorithm. IETF RFC 1321.

33. Sethi, P., & Sarangi, S. R. (2023). Internet of Things: Architectures, protocols, applications. Journal of Electrical Systems.

34. Singh, S., et al. (2024). Advanced lightweight encryption schemes. IEEE Access.

35. Stallings, W. (2022). Cryptography and Network Security: Principles and Practice. Pearson.

36. Suo, H., et al. (2023). Security in IoT: A review. IEEE International Conference on Computer Science.

37. Taneja, J., & Davy, A. (2023). Resource-aware cryptography for IoT. IEEE Transactions on Computers.

38. Wang, W., et al. (2024). Secure communication in IoT systems. IEEE Communications Magazine.

39. Whitmore, A., Agarwal, A., & Da Xu, L. (2022). The Internet of Things—A survey. Information Systems Frontiers.

40. Zhao, K., & Ge, L. (2023). A survey on IoT security. IEEE Transactions on Industrial Informatics.