Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
в.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: Механізм безпеки для інтернет речей

Виконавець: студент IV курсу, групи КБ-42

_____ **Дмитро ЦАРУК**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Андрій ФЕСЕНКО	

Нормоконтроль	Андрій БІГДАН	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
в.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студента **КБ-42** **Царука Дмитра Олексійовича**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Механізм безпеки для інтернет речей

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Інтернет речей, архітектура IoT, NIST, IEEE 802.11i, IETF, GDPR, CCPA, протоколи розподілу ключів, асиметрична криптографія, RSA, ECC, AES-GCM, SHA-256, програмування, Python.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з теорією інтернету речей, хмарних обчислень та їх проблемами інтеграції, хмар як платформ, архітектурами IoT, викликами та загрозами для IoT, вимогами до безпеки протоколів розподілу ключів, принципами асиметричної криптографії, аналізом RSA з ECC, методом Advanced Encryption Standard - Galois/Counter Mode, принципами гібридного механізму захисту.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність полягає в реалізації механізму шифрування та дешифрування за допомогою ECC у поєднанні з Advanced Encryption Standard - Galois/Counter Mode для безпечного обміну інформацією в IoT.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Андрій ФЕСЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Дмитро ЦАРУК (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів випускної кваліфікаційної роботи	Термін виконання робіт (початок-кінець)	Відмітка про виконання
1.	Уточнення постановки задачі	21.11.2022 – 23.11.2022	*виконано*
2.	Аналіз літератури	23.11.2022 – 06.01.2023	*виконано*
3.	Обґрунтування вибору рішення	06.01.2023 – 20.01.2023	*виконано*
4.	Збір даних	20.01.2023 – 10.02.2023	*виконано*
5.	Вибір алгоритму для програмної реалізації	10.02.2023 – 25.02.2023	*виконано*
6.	Вибір методи покращення захисту	25.02.2023 – 18.03.2023	*виконано*
7.	Проведення аналізу отриманих результатів	18.03.2023 – 24.04.2023	*виконано*
8.	Робота над висновками	24.04.2023 – 30.04.2023	*виконано*
9.	Оформлення презентації	30.04.2023 – 05.06.2023	*виконано*

Завдання видав _____ (підпис) Андрій ФЕСЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Дмитро ЦАРУК (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Пояснювальна записка: 101 с., 23 рис., 4 табл., 7 додатків, 101 джерело.

Об’єкт дослідження: процес захисту розподілу ключів для інтернет речей.

Мета роботи: підвищення безпеки середовищ IoT шляхом вирішення проблем, пов’язаних із безпечним розповсюдженням ключів.

Предмет дослідження: методи конфіденційного обміну інформації по відкритим каналам зв'язку в середовищі IoT.

Методи дослідження: теоретичний аналіз, структурний аналіз, порівняння, синтез, моделювання.

Практичне значення роботи полягає в полягає в реалізації механізму шифрування та дешифрування за допомогою ECC у поєднанні з Advanced Encryption Standard - Galois/Counter Mode для безпечного обміну інформацією в IoT.

Результати здійснених у дипломній роботі досліджень потенційно можуть вплинути на розробку та впровадження захищених систем IoT, дозволяючи пристроям IoT безпечно спілкуватися та обмінюватися даними. Запропонований механізм шифрування та дешифрування пропонує практичні рішення для керування ключами та підвищує безпеку обміну інформацією в середовищі IoT.

Ключові слова: інтернет речей, хмарні обчислення, алгоритм шифрування, асиметрична криптографія, протоколи розподілу ключів, RSA, криптографія еліптичних кривих, AES-GCM, гешування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПРК	–	протокол розподілу ключів
AAD	–	Additional Authenticated Data
AES	–	Advanced Encryption Standard
AES-GCM	–	Advanced Encryption Standard - Galois/Counter Mode
AES-NI	–	Advanced Encryption Standard - New Instructions
CCPA	–	California Consumer Privacy Act
DDOS	–	Distributed Denial of Service
ECC	–	Elliptic Curve Cryptography
GDPR	–	General Data Protection Regulation
IETF	–	Internet Engineering Task Force
IV	–	Initialization Vector
NIST	–	National Institute of Standards and Technology
RSA	–	Rivest-Shamir-Adleman
SHA-256	–	Secure Hash Algorithm 256-bit

ВСТУП

Актуальність теми "Механізм безпеки для інтернет речей" обґрунтовується наступними аспектами, які базуються на реальних дослідженнях, прецедентах та статистичних даних:

1. Зростаюча кількість пристроїв IoT: За останні кілька років спостерігається експоненційне зростання кількості підключених до IoT пристроїв, таких як домашні розумні пристрої, медичні пристрої, автомобілі, промислові системи тощо. За даними IDC, кількість підключених до IoT пристроїв може досягти 55,7 мільярда одиниць до 2025 року. Це створює великі виклики для забезпечення безпеки цих пристроїв, оскільки вони можуть бути потенційно вразливими перед атаками та зловживанням даних [1].

2. Загрози та вразливості IoT: Інтернет речей стикається з різноманітними загрозами та вразливостями, такими як атаки з метою перехоплення даних, внесення змін у роботу пристроїв, відмова в доступі до послуг, витік конфіденційної інформації, фізичні атаки на пристрої тощо. Багато з цих загроз можуть мати серйозні наслідки, такі як порушення приватності, втрата фінансових ресурсів, порушення роботи промислових систем, пошкодження репутації та втрата довіри користувачів.

3. Низький рівень безпеки в багатьох пристроях IoT: Багато підключених до IoT пристроїв мають низький рівень захисту. Це може бути пов'язано з різними факторами, такими як відсутність вбудованих захисних механізмів, недостатня увага до оновлення програмного забезпечення, використання застарілих протоколів комунікації, недостатня свідомість користувачів щодо заходів безпеки та багато іншого. Це робить ці пристрої вразливими перед різними видами кібератак, що може призвести до серйозних наслідків.

4. Зростаюча кількість кібератак на пристрої IoT: За останні роки відбувся значний приріст кібератак на пристрої IoT. Зловмисники активно використовують вразливості цих пристроїв для здійснення різноманітних атак, таких як DDoS-атаки, крадіжка даних, шпигунство, програми вимагачі та інше. Згідно з дослідженням

"Threats Report 2021" від компанії Nokia, кількість кібератак на пристрої IoT зросла в середньому на 50% у 2020 році [2].

5. Потенційні наслідки безпекових вразливостей IoT: наприклад, вразливості в системах управління промисловими процесами можуть призвести до виробничих аварій та зупинки виробництва, що може мати серйозний вплив на економіку та життя людей, в медичних пристроях можуть ставити під загрозу здоров'я та безпеку пацієнтів, в домашніх розумних пристроях можуть використовуватися для незаконного вторгнення у приватне життя користувачів, крадіжки особистих даних та порушення їхньої конфіденційності. Крім того, кібератаки на IoT можуть мати вплив на критичну інфраструктуру, таку як електроенергетика, транспортні системи, водопостачання та інші, що може викликати серйозні наслідки для суспільства.

6. Зростання свідомості щодо важливості забезпечення безпеки в IoT: Останнім часом спостерігається зростання свідомості про важливість забезпечення безпеки в контексті розвитку IoT. Багато компаній, виробників пристроїв IoT, враховують аспекти безпеки на ранніх етапах проектування та розробки. Також з'являється більше наукових досліджень, спрямованих на виявлення вразливостей та розробку захисних механізмів для пристроїв IoT.

7. Застосування нових технологій для забезпечення безпеки в IoT: Для забезпечення безпеки в IoT застосовуються різні технології. Наприклад, блокчейн технології можуть забезпечувати децентралізовану реєстрацію даних, що може захистити від несанкціонованого доступу та маніпуляції з даними. Технології шифрування можуть захистити комунікацію між пристроями в IoT від перехоплення та розкриття інформації. Машинне навчання та штучний інтелект можуть використовуватися для виявлення аномальної активності та вразливостей в системі в реальному часі, що може сприяти швидкому виявленню та вирішенню проблем безпеки [3].

8. Регуляторні заходи та стандарти: Забезпечення безпеки в IoT також вимагає регуляторних заходів та стандартів. Багато країн та регіонів вже прийняли або розробляють законодавство, що регулює аспекти безпеки в IoT, такі як захист даних, захист від кібератак, стандарти безпеки пристроїв тощо. Це сприяє покращенню

безпеки в IoT шляхом встановлення мінімальних вимог та стандартів безпеки для виробників, операторів та користувачів.

9. Приклади реальних загроз та інцидентів: Низка реальних загроз та інцидентів, пов'язаних з безпекою в IoT, також підкреслює актуальність цієї теми. Наприклад, атака Mirai, яка сталася у 2016 році, використовувала вразливості в пристроях IoT, щоб створити мережу зомбі-ботнету, яка використовувалася для здійснення масштабних кібератак на веб-сервери. Інші приклади включають компрометацію систем "розумного будинку", витоки особистих даних зі смарт-пристроїв, крадіжки конфіденційної інформації з медичних пристроїв тощо [4].

Ці фактори демонструють актуальність теми безпеки в контексті розвитку IoT. Відсутність належного захисту може призвести до серйозних наслідків, таких як втрата конфіденційності, порушення приватності, кібератаки, фінансові втрати, втручання у функціонування критичних інфраструктур та інші негативні наслідки. Дослідження, статистичні дані та прецеденти підтверджують, що забезпечення безпеки в IoT є надзвичайно важливим аспектом розвитку цієї технології.

Основна мета дипломної роботи на тему "Механізм безпеки для інтернету речей" полягає в розробці ефективного механізму захисту для вимог до безпеки, з урахуванням аналізу загроз та вразливостей, вимог до безпеки, огляду існуючих рішень.

Для досягнення основної мети, необхідно виконати наступні конкретні завдання:

1. Виконати аналіз концепцій типів архітектур, хмарних обчислень, хмари як платформи, проблем інтеграції в хмарні обчислення та викликів для IoT.
2. Проаналізувати ризики для IoT, зокрема інцидентів та статистичних даних, щоб обґрунтувати необхідність впровадження захищених ПРК для підтримки цілісності та конфіденційності даних, що передаються в мережах IoT.
3. Провести аналіз стандартів та правил, які регулюють вимоги безпеки для ПРК у контексті IoT.
4. Дослідити застосування асиметричної криптографії для IoT.

5. Проаналізувати фундаментальні концепції, що лежать в основі асиметричної криптографії, включаючи використання відкритих і закритих ключів, а також дослідити різні алгоритми шляхом розгляду математичних формул та схем, таких як RSA та ECC, які використовуються в IoT.

6. Дослідити різні аспекти ECC і RSA, де алгоритми розглядаються всебічно. Аналіз проводитиметься шляхом порівняльних таблиць згідно існуючих досліджень з точки зору деяких показників.

7. Проаналізувати метод покращення захисту обраного протоколу розподілу ключів після порівняння за різними показниками та обґрунтувати доцільність реалізації Advanced Encryption Standard - Galois/Counter Mode в ECC.

8. Реалізувати та описати механізм шифрування та дешифрування за допомогою ECC у поєднанні з Advanced Encryption Standard - Galois/Counter Mode для захисту зв'язку IoT.

9. Оцінити вплив розробленого механізму безпеки на ресурси пристроїв IoT.

Об’єктом дослідження є процес захисту розподілу ключів для інтернету речей

Предметом цього дослідження є методи конфіденційного обміну інформації по відкритим каналам зв’язку в середовищі IoT. Дослідження вивчатиме існуючі протоколи, криптографічні принципи, що лежать в їх основі, і їх придатність для унікальних характеристик та обмежень мереж IoT.

Для досягнення мети дипломної роботи будуть використані різні методи дослідження, включаючи теоретичний аналіз, структурний аналіз, порівняння, синтез та моделювання.

Практичне значення роботи полягає реалізації механізму шифрування та дешифрування за допомогою ECC у поєднанні з Advanced Encryption Standard - Galois/Counter Mode для безпечного обміну інформацією в IoT. Результати здійснених у дипломній роботі досліджень потенційно можуть вплинути на розробку та впровадження захищених систем IoT, дозволяючи пристроям IoT безпечно спілкуватися та обмінюватися даними. Запропонований механізм шифрування та

дешифрування пропонує практичні рішення для керування ключами та підвищує безпеку обміну інформацією в середовищі IoT.

Апробація тематики дипломної роботи була сформована у вигляді тез та подані для доповідання та обговорення на XII Міжнародній науковій конференції "Інформація, комунікація, суспільство – 2023". Тези були опубліковані в збірнику "Інформація, комунікація, суспільство 2023: Матеріали XII-ї Міжнародної наукової конференції ІКС-2023" на сторінках 40-41.

РОЗДІЛ 1
АНАЛІЗ ЗАГРОЗ ТА ВЛАСТИВОСТЕЙ В ІНТЕРНЕТ РЕЧАХ

1.1 Огляд сучасного стану розвитку інтернет речей

IoT — це популярна технологія, яка контролює підключені розумні гаджети [5]. IoT стає однією з найбільш зростаючих сфер, і останнім часом використовується в багатьох програмах, таких як розумний дім пристроїв [6], які є прикладом таких програм, як інтелектуальні персональні помічники, захисне обладнання, таке як інтелектуальне особисте спорядження, деякі корисні пристрої IoT і програми для промислової допомоги тощо.

Пристрої, які використовуються в IoT зазвичай малопотужні і мають обмежену обробну здатність, у той час як компонент розрахунку виконується на внутрішньому хмарному сервері. Деякі дослідження створили фреймворк розпізнавання обличь як IoT-додаток з обчислювальним сервером у двох різних інфраструктурах: локальній межі хмари поблизу клієнта та комерційній хмарній платформі [7, с. 3]. Реалізація частини обробки в крайовому вузлі або шлюзі також може значно зменшити кількість пакетів даних і, як наслідок, затримку мережі. Їхній час обробки створеної ними системи, а також затримка мережі були протестовані та порівняні.

IoT підключає фізичні пристрої до цифрового світу, але це лише початок. Найважливішим є питання про те, що робити з усією цією інформацією та як використовувати дані з пристроїв. Використання інформаційно-комунікаційних технологій (ІКТ) у системах управління логістикою стало важливим фактором посилення конкурентоспроможності логістики в багатьох галузях. Можливості IoT збільшують здатність системи приймати рішення добре поінформованим способом, які враховують обмін даними IoT. Обробка та аналіз даних є найважливішим питанням про те, що робити з усією цією інформацією та як використовувати дані з пристроїв.

1.1.1 Хмарні обчислення

Хмарні обчислення розширюють можливості IoT. У цьому підрозділі буде розкрита архітектура IoT і особливості хмарної платформи IoT та її взаємодія з трьома основними моделями хмарних обчислень: IaaS (інфраструктура як послуга), Paas (Платформа як послуга) і SaaS (програмне забезпечення як послуга). SaaS (Відстеження як послуга), DBaaS (База даних як послуга), VSaaS (Відеоспостереження як послуга) та інші інтелектуальні сервіси та програми є одними з нових сценаріїв, доступних завдяки інтеграції хмари та IoT.

Різноманітні продукти компаній, дослідницькі проекти та проекти з вільно доступним вихідним кодом у різних сферах хмарних обчислень та інтеграції IoT: Nimbits, Thing Speak, Paraimpu, Device Cloud [8] і Sensor Cloud [9]. Щоб спілкуватися з об’єктами IoT з обмеженими ресурсами, використовуються веб-сервіси в архітектурному стилі Presentational State Transfer (REST) [10], протоколи веб-передачі Constrained Application Protocol (COAP) [11] і Message Queue Telemetry Transport (MQTT). Для обмежених мереж IoT і хмарна інтеграція використовують такі мережеві протоколи, як IPv6 через бездротову персональну мережу з низьким енергоспоживанням (6LoWPAN) і IPv6 через Bluetooth з низьким енергоспоживанням. IEEE 802.15.4, IEEE 802.11ah, Z-Wave, Wireless HART, Bluetooth і Zigbee — це протоколи канального рівня для пристроїв IoT, які використовуються для зв’язку на короткій відстані [12, с. 1-5].

Окрім сільського господарства [13], охорони здоров’я, розумного міста [14], відеоспостереження, розумного дому та інтелектуальних лічильників, IoT та хмарні обчислення мають багато інших застосувань. IoT і хмарна інтеграція включає в себе кілька викликів і проблем, як стандартизація зв’язку між машинами (M2M) і взаємодію, потужність та енергоефективність пристроїв для передачі та обробки даних, великі дані, створені кількома пристроями, безпека та конфіденційність, методологія інтеграції, ціноутворення та виставлення рахунків, мережеві комунікації, зберігання тощо. інтегровані програми, а також пов’язані з ними виклики та проблеми. Вбудовані пристрої мають обмежені можливості зберігання, потужності та

обчислень і вимагають інтеграції вбудованих пристроїв із великими пулами ресурсів, такими як хмара. Очікується, що інтеграція цієї технології принесе величезне зростання багатообіцяючих додатків IoT зараз і в майбутньому.

1.1.2 Хмара як платформа

Програми IoT повинні враховувати різні фактори, залежно від контексту програми. Зібрані дані, що генеруються різними пристроями, потребують обробки, інтерпретації та зберігання, а рішення, пов'язані з реалізацією IoT-програм, мають вирішальне значення для їх успіху. Однак, виникають проблеми щодо надійності, продуктивності, безпеки та конфіденційності. Подібні проблеми вже відомі в галузі мобільних обчислень. Модель хмарних обчислень має свої переваги та недоліки. Ці обмеження призводять до інтеграції з хмарними обчисленнями, які мають практично необмежену потужність для зберігання та обчислювальних операцій. Основні хмарні IoT-платформи, що лідирують на світовому ринку, включають Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform [15]. У цьому документі описується інтелектуальна платформа, яка може використовуватися для швидкого прототипування в системах управління холодовим ланцюгом з особливим акцентом на безпеку. Розглядаються такі теми, як збір даних у реальному часі, локальна обробка даних та передача їх на безпечну хмарну платформу, залишаючи подальшу обробку даних та їх візуалізацію для подальшого розвитку [16]. Екосистема IoT включає розумні пристрої, які мають підтримку інтернету і збирають, передають та обробляють дані з навколишнього середовища за допомогою вбудованих систем, а саме процесорів, датчиків та комунікаційних обладнань. За допомогою підключення до IoT-шлюзу або іншого периферійного пристрою, пристрої IoT можуть обмінюватися даними з датчиків. Дані з датчиків можуть бути відправлені до хмари для аналізу або оброблятись локально. Ці пристрої можуть взаємодіяти між собою та реагувати на отриману інформацію. Люди можуть взаємодіяти з пристроями для налаштування, надання інструкцій та отримання даних, але багато роботи виконуються самими пристроями без прямого втручання людини [12, с. 2].

1.1.3 Тенденції розвитку інтернет речей

IoT зараз обертається навколо швидкого розвитку технологій і, за прогнозами, відіграватиме значну роль у найближчі роки. Багато дослідників звернули увагу на IoT, щоб виявити проблеми, пов'язані з його дизайном, архітектурою та безпекою. Наприклад, стаття містить вичерпний огляд різних концепцій і застосувань IoT та його розширень. Автори обговорюють проблеми та можливості IoT. У статті також представлені деякі приклади існуючих і потенційних систем IoT у різних сферах, таких як охорона здоров'я, сільське господарство, розумні міста тощо. Основна метою було висвітлити важливість і переваги IoT і пов'язаних з ним технологій для покращення якості життя та підвищення ефективності та продуктивності різних процесів [17]. Різноманітність мов, протоколів і стандартів, а також відсутність згоди щодо того, який найкраще працює для певних рівнів IoT, є однією з багатьох ключових труднощів. IoT не має єдиної платформи стандартизації; а отже неоднорідність пов'язаних речей спричиняє її різницю. Наприклад, Міжнародний союз електрозв'язку (МСЕ) наразі підключає IoT до «інформації, яка забезпечує розширені послуги шляхом з'єднання речей (фізичних і віртуальних) на основі існуючих і нових сумісних інформаційних і комунікаційних технологій. IoT — це інтеграція кількох об'єктів, мереж і протоколів зв'язку, дротових і бездротових датчиків для отримання технологій і комунікаційних рішень [18, 19].

IoT відноситься до мережевого взаємозв'язку кожного разу, що робить їх розумними пристроями. IoT – це нова тема технологічного, соціального та економічного значення. Споживчі товари, споживчі товари тривалого користування, автомобілі та вантажівки, промислові та комунальні компоненти, датчики тощо. Як уже згадувалося. Програма, яка автоматично або напівавтоматично інтегрує реальні об'єкти та місця з інтернетом. Визначення IoT – це з'єднання та обмін даними між пристроями реального та віртуального світу. Ці пристрої включають ПК, ноутбуки, планшети, КПК, смартфони, холодильники, телевізори, розумні будинки, автомобілі тощо, але також поширюються на все, включаючи тварин. Це підключення запускається датчиком, який отримує дані від об'єктів або інтернету. Існує багато

методів, які були запропоновані та застосовані до цих питань. Наприклад, у статті обговорюється інтеграція LoRaWAN з мережами 5G і проводиться представлення оцінки продуктивності запропонованої архітектури [20]. У наступній статті розглядається реалізація алгоритму інтелектуального керування чергою для вузькосмугових послуг IoT у мережах 4G/5G. Алгоритм визначає пріоритет трафіку на основі вимог до якості обслуговування (QoS) і призначений для підвищення продуктивності мережі, та обговорюється використання методів машинного навчання для оптимізації продуктивності алгоритму [21].

1.1.4 Архітектура інтернет речей

Архітектура IoT відноситься до загальної структури системи IoT, включаючи її різні рівні, компоненти та протоколи зв'язку. Показуючи, як вони пов'язані та взаємодіють один з одним. Він забезпечує структуру для проектування та розгортання систем IoT, уможливлюючи інтеграцію фізичних пристроїв, датчиків і програмних додатків у єдину систему, яка може збирати, обробляти та аналізувати дані [22]. На даний момент узгодженої еталонної архітектури не існує, і було доведено, наскільки складно створити еталонну архітектуру, незважаючи на численні зусилля зі стандартизації [23].

Система IoT характеризується своєю вертикальною фрагментацією для всіх можливих застосувань, кожна з яких спирається на численні та різні змінні та специфікації дизайну. На рисунку 1.1 показано найпоширеніші використовувані архітектури IoT.

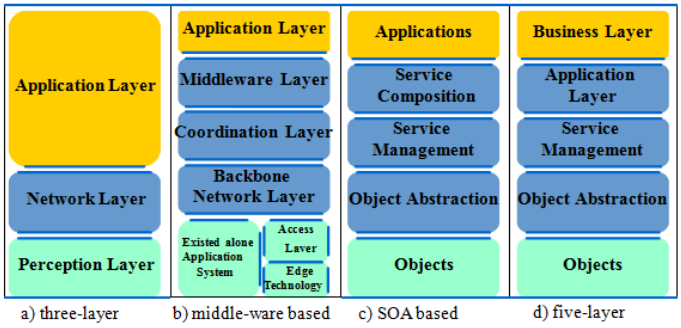

Рисунок 1.1 – Поширені архітектури IoT [22]

1.1.4.1 Трирівневі архітектури IoT

Трирівнева архітектура IoT є спрощеною версією архітектури IoT, що складається з трьох рівнів: рівня сприйняття, мережі та рівня додатків [24, с. 3]. На рисунку 1.2 показано схему трирівневої архітектури:

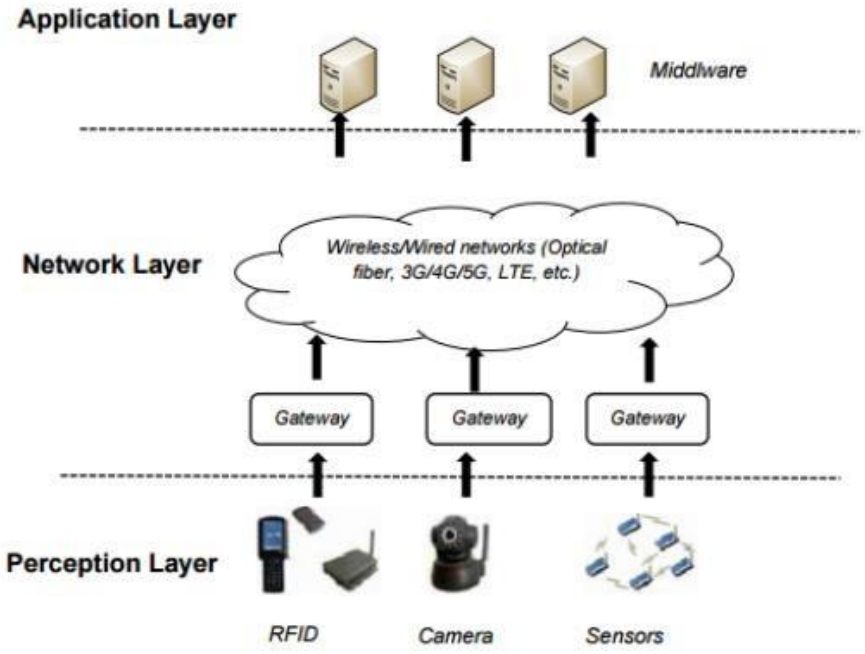

Рисунок 1.2 – Трирівневі архітектури IoT

Рівень сприйняття: рівень сприйняття, також званий рівнем ресурсів, головним чином спрямований на збір даних із фізичного середовища за допомогою датчиків та інших пристроїв збору даних. Цей рівень включає різні датчики, такі як датчики температури, вологості, руху та освітленості. Рівень сприйняття також включає в себе приводи, які дозволяють пристроям реагувати на зібрані дані та RFID-мітки, які широко використовуються в харчовій промисловості.

Мережевий рівень: мережевий рівень з’єднує дані рівня сприйняття з прикладним рівнем. Він охоплює різні комунікаційні технології, такі як Wi-Fi, Bluetooth, Zigbee тощо. Мережевий рівень також включає шлюзи як посередники між

рівнем сприйняття та прикладним рівнем, де шлюзи виконують фільтрацію, агрегацію та стиснення даних перед передачею даних на прикладний рівень [25].

Рівень додатків: відповідає за розробку та розгортання додатків, які дозволяють користувачам взаємодіяти з системою IoT. Він включає різні програми, такі як аналіз даних, прогнозне обслуговування та віддалений моніторинг. Додатки дозволяють користувачам віддалено контролювати та контролювати пристрої, автоматизувати різні процеси та приймати зважені рішення на основі зібраних даних.

1.1.4.2 Сервісно-орієнтована архітектура

SoA (сервісно-орієнтована архітектура) — це підхід до проектування, який використовує слабозв’язані служби для забезпечення зв’язку та обміну даними між різними пристроями та системами в системі IoT [26, с. 2].

У цій архітектурі кожен пристрій або система в мережі IoT розглядається як постачальник або споживач послуг. Послуги, що надаються кожним пристроєм або системою, доступні через стандартизовані інтерфейси, які можуть використовувати інші пристрої або системи [26, с. 2].

Архітектура IoT на основі SoA складається з трьох рівнів:

Рівень пристроїв складається з сенсорних пристроїв, які збирають дані з середовища та взаємодіють з іншими пристроями в мережі.

Рівень сервісу, який надає набір сервісів, які надаються через стандартизовані інтерфейси. Інші пристрої або системи можуть використовувати ці послуги в мережі.

Рівень інтерфейсу складається з програм, які використовують послуги, які надає рівень сервісу. Ці програми можуть надавати інформацію, аналітику або керувати діями, спираючись на дані, зібрані з мережі IoT [26, с. 7].

Ця архітектура сприяє взаємодії та гнучкості, дозволяючи різноманітним пристроям і системам спілкуватися один з одним за допомогою стандартних інтерфейсів. Це також дозволяє створювати нові служби та програми, які можуть використовувати дані, зібрані з мережі IoT. Ця архітектура сприяє горизонтальній фрагментації системи IoT [26, с. 9]. На рисунку 1.3 зображена архітектура SoA IoT.

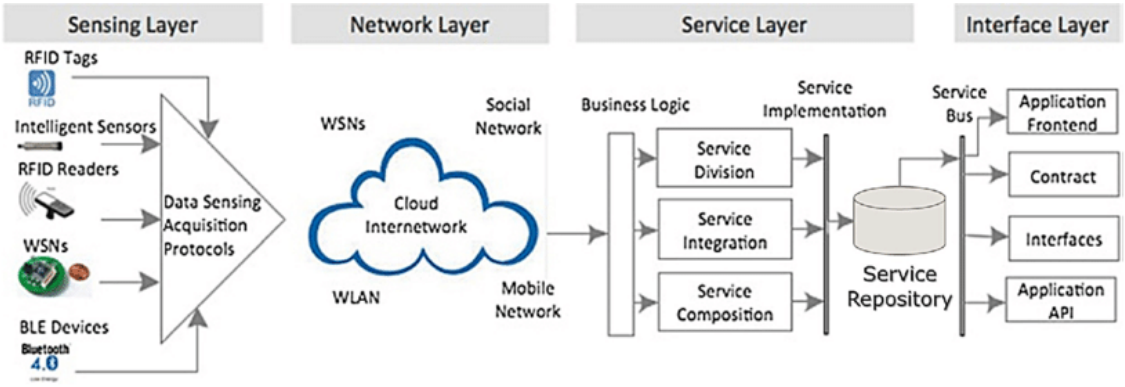

Рисунок 1.3 – Архітектура SoA IoT

1.1.4.3 Проміжне програмне забезпечення IoT на основі послуг

Архітектури на основі проміжного програмного забезпечення (Middleware) IoT відносяться до підходу до проектування, який використовує програмне забезпечення проміжного програмного забезпечення для полегшення зв'язку та обміну даними між пристроями та системами в екосистемі IoT [27].

Проміжне програмне забезпечення діє як перехідний рівень між рівнями пристрою та додатками, надаючи послуги та API (інтерфейси програмування додатків), які дозволяють пристроям і програмам взаємодіяти [27]. На рисунку 1.4 наведена схема проміжного програмне забезпечення IoT на основі послуг.

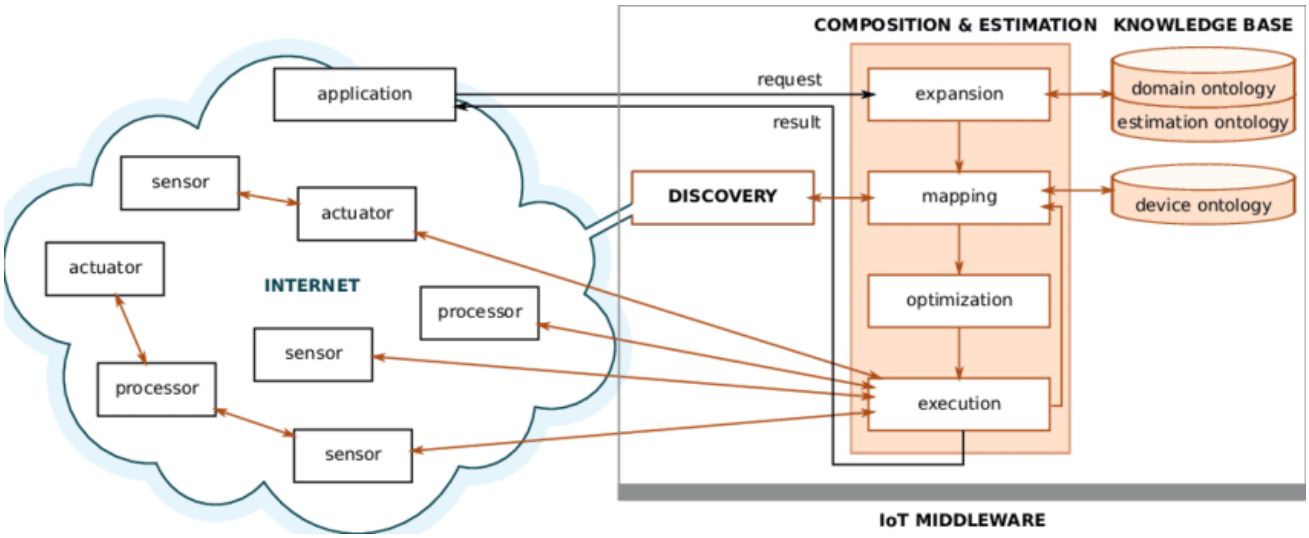

Рисунок 1.4 – Проміжне програмне забезпечення IoT на основі послуг [27]

Існують різні типи архітектур IoT на основі проміжного ПЗ, зокрема:

- Архітектура проміжного програмного забезпечення, орієнтованого на повідомлення (MOM): ця архітектура використовує систему обміну повідомленнями для передачі даних між пристроями та програмами. Пристрої надсилають повідомлення до центрального брокера повідомлень, який потім пересилає їх до відповідної програми на основі попередньо визначених правил [28].
- Архітектура проміжного програмного забезпечення для публікації/підписки: ця архітектура дозволяє пристроям публікувати дані в центральному посереднику, який потім пересилає його всім програмам, які підписалися на цей конкретний потік даних [29].
- Архітектура проміжного програмного забезпечення для віддаленого виклику процедур (RPC): Ця архітектура дозволяє пристроям викликати віддалені процедури, розміщені на інших пристроях або серверах, дозволяючи їм обмінюватися даними та виконувати різні функції [30].
- Архітектура проміжного програмного забезпечення, орієнтована на дані. Ця архітектура зосереджена на управлінні даними в екосистемі IoT, надаючи такі функції, як фільтрація даних, агрегація та зберігання [31].

1.1.4.4 П’яти-рівнева архітектура

Згідно з дослідженням [32], п’яти-рівнева архітектура IoT має деякі спільні рівні, як показано рисунку 1.5.

1. Рівень пристрою: складається з об’єкта та сенсорного пристрою.

Типи датчиків:

- RFID
- 2D-штрих-код-інфрачервоний датчик, в залежності від техніки розпізнавання товару.

Сенсорні пристрої отримують дані від об’єкта в цьому рівні. Розташування, температура, напрямок, рух, вібрація, прискорення, вологість, хімічні зміни в повітрі

тощо можна включити в ці дані. Зібрані дані потім надсилаються на мережевий рівень для безпечної передачі в систему обробки даних.

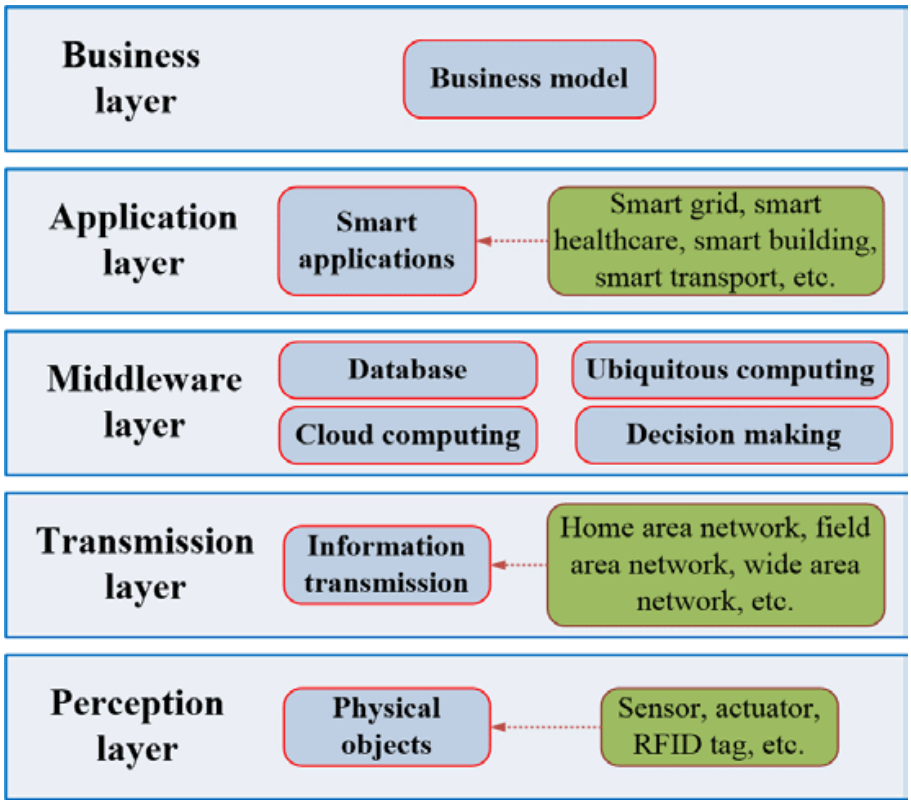

Рисунок 1.5 – Рівні інтернету речей

2. Мережевий рівень: мережевий рівень бере на себе відповідальність за передачу даних від пристроїв, які мають датчики, і передачу їх до системи обробки інформації. І це залежить від сенсорних пристроїв, можна використовувати різні типи методів передачі, будь то бездротові чи дротові, а використовувані технології можуть бути 3G, Wi-Fi, Bluetooth, UMTS ZigBee, інфрачервоний зв’язок тощо.

3. Рівень проміжного програмного забезпечення: на цьому рівні існує дуже багато типів послуг, реалізованих пристроями IoT. Лише інші пристрої, які реалізують такий самий тип служби, підключатимуться до кожного пристрою та спілкуватимуться з ним. Цей шар відповідає за керування послугами та має підключення до бази даних. Отримує дані від мережевого рівня та зберігає їх у базі даних. Потрібно виконувати масштабні розрахунки перед обробкою даних і приймати автономні рішення на основі результатів.

4. Рівень програми: цей рівень, який часто називають рівнем інтерфейсу, забезпечує повний контроль. Дані датчика обробляються враховуючи інформацію, отриманих від датчика. Приклади включають розумні міста, розумне землеробство, розумні будинки, інтелектуальні транспортні системи та інші програми.

5. Бізнес-рівень: цей рівень відповідає за керування всією системою IoT, включаючи програми та служби. Бізнес-моделі, графіки, блок-схеми тощо на основі даних, отриманих із прикладного рівня. Справжній успіх технології IoT також залежить від надійної бізнес-стратегії.

1.1.5 Хмарні послуги та їх взаємозв'язок з інтернетом речей

Більшість програм та даних надаються з хмари, що є основою для різних послуг через інтернет. Хмарні послуги включають ресурси і програми, такі як мережі, бази даних, сховища даних, сервери та програмне забезпечення. З'єднані з інтернетом електронні пристрої стають частиною IoT, що призводить до великого обсягу генерованої інформації. IoT потребує інтеграції з хмарними обчисленнями, щоб впоратися з обробкою, зберіганням та комунікацією даних. Таким чином, IoT може використовувати безліч можливостей та ресурсів хмари для своєї ефективної роботи [12, с. 3].

Хмара може забезпечити наступні послуги для IoT:

- Управління IoT;
- Впровадження додатків і сервісів;
- Забезпечення проміжного шару між речами та додатками, приховуючи всі деталі.

З іншого боку, хмара може отримати вигоду від IoT за рахунок:

- Нарощування;
- Надання нових послуг;
- Ефективного та доступного рішення для підключення, відстеження та керування в будь-який час і в будь-якому місці за допомогою спеціальних порталів та інтегрованих програм;

- Доступності високошвидкісної мережі, яка забезпечує ефективний моніторинг і контроль віддалених мереж, їх координацію, зв’язок і доступ до згенерованих даних у режимі реального часу. Все це вплине на майбутні розробки додатків, де збір, обробка та передача інформації створить нові проблеми, особливо в багатохмарне середовище.

Переваги інтеграції:

- Обмін даними та програмами;
- Доставка персоналізованих додатків;
- Застосування автоматизації збору та розповсюдження даних за низькими витратами.

1.1.6 Виклики для інтернет речей

IoT має багато проблем, які автори намагаються вирішити та знайти рішення, які могли б усунути та покращити його впровадження, щоб ви могли створити величезну спільноту з мільярдів або трильйонів «речей», пов’язаних і спілкуючись один–одним, мають справу з багатьма технічними та прикладними проблемами. Ці виклики можна класифікувати в загальний клас [33], такий як секретна глобальна співпраця, етика, суспільство контролю, стеження, згода та життя, кероване даними, бізнес-моделі, нові валюти в IoT та довіра, технологічні виклики та пошук ідеального балансу між планування зверху вниз і інновації знизу вгору.

Також існують інша важлива проблема, як відсутність регулювання. Полягає вона в тому, що державне регулювання часто відстає від поточного стану технологій. Зі стрімким розширенням IoT, яке відбувається щодня, уряд відстає, а підприємства часто залишаються без важливої інформації, яка їм потрібна для прийняття рішень. Нові хвилі технологій сумно відомі тим, що мають безліч конкурентів, які змагаються за домінування на ринку, IoT не є винятком. Це може бути гарною новиною, оскільки вона дає споживачам більше можливостей, але це також може призвести до неприємних проблем із сумісністю. Проблеми сумісності виникають у сфері домашніх сітчастих мереж. Bluetooth вже давно є галузевим стандартом для

сумісності пристроїв IoT. Фактично, він був названий на честь Гаральда Блютуза, стародавній правитель, який відзначився тим, що об’єднав ворогуючі племена.

Однак, коли справа доходить до домашньої автоматизації за допомогою сітчастої мережі, з’явилися різні конкуренти, включаючи такі протоколи, як Zigbee і Z-Wave, щоб кинути виклик пропозиціям сітчастої мережі Bluetooth. Можуть знадобитися роки, щоб ринок достатньо заспокоївся, щоб увінчати єдиний глобальний домашній стандарт IoT. Деякі нерозривні проблеми, такі як обмежена пропускна здатність, і, нарешті, очікування клієнтів. З такою великою конкуренцією на ринку IoT клієнти, які не отримують того, що отримують хочу, не вагаючись, піду за цим в іншому місці. Клієнти очікують плавнішого та сучаснішого досвіду, тому підприємства, які бажають вийти на цю конкурентну та винахідницьку сферу, повинні бути готові до ринку, який ніколи не зупиняється. IoT — це захоплююча сфера з великим потенціалом змінити те, як ми живемо, працюємо та граємо. Однак для того, щоб IoT залишався безпечним і продуктивним, технологічний сектор, уряд і споживачі повинні погодитися з питаннями безпеки та продуктивності [12, с. 3].

1.2 Дослідження загроз та вразливостей в інтернеті речей

IoT — це швидкозростаюча мережа підключених пристроїв і систем, включаючи все: від інтелектуальних термостатів і переносних пристроїв до промислового обладнання та медичних пристроїв. Хоча цей взаємозв’язок може запропонувати значні переваги, він також представляє безліч нових вразливостей і потенційних загроз:

1. Загрози для пристроїв IoT: Пристрої IoT уразливі до різних форм атак, таких як зловмисне програмне забезпечення та програми-вимагачі. Хакери можуть використовувати невиправлені вразливості в мікропрограмі або операційній системі пристроїв IoT, щоб отримати контроль над пристроями або викрасти конфіденційну інформацію. Пристрої IoT також можна використовувати як точку входу для проникнення в інші системи в мережі.

a. У 2016 році ботнет Mirai заразив сотні тисяч пристроїв IoT, включаючи маршрутизатори, IP-камери та відеореєстратори, для запуску широкомасштабних розподілених атак типу «відмова в обслуговуванні» (DDoS) [4]. Ботнет - мережа скомпрометованих комп'ютерів або пристроїв IoT, під контролем зловмисника. Вони використовуються для шкідливих дій, таких як DDoS, спам та крадіжка облікових даних. Ботнети створюються через вразливості, шкідливе ПЗ або соціальну інженерію. Зловмисник керує ботами через сервер C&C, вказуючи їм, що робити. Захист від ботнетів включає оновлення ПЗ, міцні паролі та моніторинг активності [34, с. 1-6]. Повертаючись до прецеденту з Mirai, ця атака призвела до широкого збою в роботі інтернет-сервісів, зокрема Twitter, Reddit і Netflix. Згідно звіту Symantec Internet Security Threat Report, який містить огляд загроз для пристроїв IoT, що зловмисники дедалі частіше атакують пристрої IoT і що кількість атак IoT зросла на 600% між 2016 і 2019 роками. У звіті також підкреслюється, що зловмисники все частіше використовують пристрої IoT як спосіб отримати доступ до інших систем на мережу, а у 2020 році на пристрої IoT припадало 33% усіх заражень шкідливим програмним забезпеченням [35].

b. У 2019 році атака програм-вимагачів LockerGoga торкнулася Norsk Hydro, норвезького виробника алюмінію, спричинивши зупинку виробництва та фінансові збитки на суму близько 40 мільйонів доларів. Атаку сприяло використання вразливості в системі VPN, яка потім поширилася на промислові системи управління та пристрої IoT [36].

c. У 2020 році було виявлено варіант зловмисного ПЗ ботнету IoT під назвою «Fbot», який заражав маршрутизатори, розумні домашні пристрої та інші пристрої IoT для запуску DDoS-атак [37].

d. У 2019 році дослідники виявили вразливість у пристроях Amazon Echo, яка дозволяла зловмисникам віддалено змушувати пристрій постійно перезавантажуватися, роблячи його непридатним для використання [38]. Схему принципу роботи шкідливого програмного забезпечення у простоях Amazon Echo наведено на рисунку 1.6.

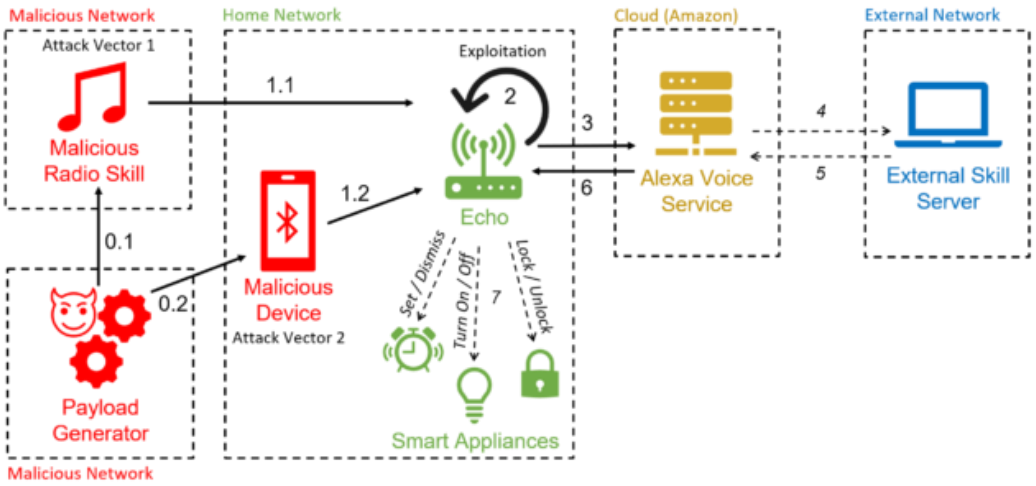

Рисунок 1.6 – Схема експлуатації вразливості у Amazon Echo

2. Ризики конфіденційності та безпеки даних: Пристрої IoT збирають величезну кількість даних, часто включаючи особисту інформацію, таку як імена, адреси та дані кредитної картки, а також GPS координати. Деякі пристрої IoT, такі як дрони та автономні транспортні засоби, покладаються на GPS для навігації та відстеження місцезнаходження. Однак сигнали GPS можна легко підробити, що може призвести до втрати цих пристроїв, дезорієнтації або навіть зіткнення з іншими об’єктами. У 2019 році дослідники продемонстрували, як спуфінг GPS можна використовувати, щоб взяти під контроль дрон і змусити його злетіти з курсу [39]. Ці дані можуть бути перехоплені, витоку або викрадені зловмисниками, які можуть використовувати їх для крадіжки особистих даних або інших зловмисних цілей. Крім того, пристрої IoT можуть використовувати слабкі або застарілі протоколи шифрування, що полегшує зловмисникам перехоплення та дешифрування конфіденційних даних під час передачі.

a. У 2017 році Equifax, агентство кредитної звітності, зазнало витоку даних, у результаті якого було розкрито особисту інформацію понад 147 мільйонів осіб, включаючи номери соціального страхування, дати народження та адреси. Порушення пов’язано з уразливістю в пристрої IoT, який використовується для моніторингу та керування веб-трафіком [40].

b. У 2019 році дослідження, опубліковане в Journal of Medical Internet Research, виявило, що деякі популярні програми для відстеження здоров’я та фізичної

форми, такі як Fitbit і MyFitnessPal, були вразливі до порушень безпеки та конфіденційності. Дослідження показало, що ці додатки передають конфіденційні дані користувачів, включаючи імена, дати народження та розташування GPS, у незашифрованому вигляді через інтернет [41]. Також раніше було доведено, що пристрої Fitbit були вразливі до злому через реалізацію Bluetooth Low Energy (BLE) [42]. Подібним чином у 2018 році дослідники виявили, що MyFitnessPal був уразливим до атак із використанням міжсайтових сценаріїв (XSS) [43].

c. IoT_reaper, також відомий як IoTroop, — це ботнет, націлений на вразливі пристрої IoT. Це було виявлено наприкінці 2017 року дослідниками з Check Point, фірми з кібербезпеки. Було виявлено, що ботнет активно заражає пристрої IoT, включаючи маршрутизатори, IP-камери та цифрові відеореєстратори (DVR), використовуючи вразливості в їх мікропрограмі та паролях за замовчуванням. Вважається, що IoT_reaper заразив понад мільйон пристроїв у всьому світі та здатний запускати DDoS-атаки та інші типи атак. Ботнет був описаний як більш складний, ніж ботнет Mirai, який також був націлений на пристрої IoT у 2016 році. Однією з причин того, що IoT_reaper є більш складним, є те, що він використовує більш просунуту систему однорангового зв'язку (P2P), ніж інші ботнети. Це ускладнює виявлення та закриття ботнету дослідникам безпеки та правоохоронним органам. На відміну від інших бот-мереж, які покладаються на централізований сервер керування (C&C) для видачі команд зараженим пристроям, IoT_reaper використовує децентралізовану систему P2P. Кожен заражений пристрій спілкується з іншими зараженими пристроями в ботнеті, обмінюючись інформацією та інструкціями. Це означає, що навіть якщо один пристрій у ботнеті вимкнено, інші можуть продовжувати працювати. Окрім DDoS-атак, IoT_reaper також здатний запускати атаки для майнінгу криптовалют і викрадення конфіденційної інформації, такої як облікові дані та номери кредитних карток [44].

d. У 2018 році команда дослідників з Каліфорнійського університету в Сан-Дієго виявила вразливість у популярному пристрої IoT з голосовим керуванням, яка дозволила їм отримати криптографічний ключ, який використовується для захисту зв'язку між пристроєм і пов'язаною з ним програмою. Дослідники використовували

техніку під назвою акустичний криптоаналіз, яка передбачає аналіз звуку, який видає пристрій, щоб отримати інформацію про секретний ключ [45].

3. Погано налаштовані пристрої: Багато пристроїв IoT встановлюються та налаштовуються нетехнічними користувачами, які можуть не мати необхідних знань або навичок, щоб належним чином захистити їх. У деяких випадках паролі за замовчуванням або інші налаштування безпеки залишаються незмінними, що полегшує доступ зловмисникам. Подібним чином слабкі або неіснуючі механізми автентифікації також можуть призвести до несанкціонованого доступу та неправильного використання пристроїв IoT.

a. У 2019 році ФБР випустило попередження щодо пристроїв IoT, які використовують облікові дані за замовчуванням, заявивши, що «простий пошук в інтернеті за маркою та номером моделі вашого пристрою разом зі словами «пароль за замовчуванням» може дозволити хакерам отримати доступ до вашого пристрою. ." Подібним чином, атака ботнету Mirai була сприяла використанню використання імен користувачів і паролів за замовчуванням на пристроях IoT [46].

b. Також було досліджено, що деякі розумні домашні пристрої використовують слабкі або застарілі протоколи шифрування, що робить їх вразливими до перехоплення та дешифрування конфіденційних даних. Подібним чином виявлено, що деякі пристрої IoT використовують слабкі або взагалі відсутні механізми автентифікації, що полегшує зловмисникам отримання неавторизованого доступу.

4. Вразливості ланцюга поставок: Пристрої IoT часто складаються з компонентів і програмного забезпечення від кількох постачальників, що ускладнює відстеження та перевірку безпеки кожного компонента. Зловмисники можуть використовувати вразливості в компонентах або програмному забезпеченні, інтегрованому в пристрої IoT, піддаючи ризику всю систему. У 2020 році звіт Ponemon Institute показав, що 56% організацій зазнали атаки на ланцюг поставок за попередні 12 місяців. Атаки на ланцюг постачання включають у себе зловмисників, які втручаються в компоненти або програмне забезпечення під час виробничого процесу, що призводить до скомпрометованих пристроїв [47].

5. Ризики фізичної безпеки: пристрої IoT вразливі до фізичного втручання та крадіжки, що може призвести до витоку даних та інших інцидентів безпеки. Наприклад, згідно статті з BBC News, хакерська група під назвою Predatory Sparrow стверджувала, що спричинила пожежу в будівлі, зламавши систему опалення будівлі. У статті пояснюється, що група випустила відео, щоб підтвердити свою історію. Також у статті [48] розповідається про першу перевірену програму-вимагач для розумних термостатів, яка була продемонстрована на DEF CON 2016. В іншій статті пояснюється, що зловмисник може підвищити тепло та заблокувати термостат IoT [49].

Більшість наведених прикладів ілюструють необхідність надійних заходів безпеки, включаючи безпечні протоколи розповсюдження ключів (ПРК). Через обмежені ресурси пристрої IoT часто використовують легкі криптографічні алгоритми для захисту каналів зв'язку. Хоча ці алгоритми ефективні, вони також більш сприйнятливі до атак. Поширеним вектором атаки є перехоплення ключів під час розподілу, що призводить до компрометації всієї системи. Це робить безпеку ключових протоколів розповсюдження ключовим аспектом безпеки IoT. Статистика останніх досліджень також підкреслює вразливість основних протоколів розповсюдження для пристроїв IoT. Згідно з опитуванням Ponemon Institute у 2020 році, 97% виробників пристроїв IoT вважають, що безпеку їхніх пристроїв можна покращити. Крім того, 71% виробників пристроїв IoT повідомили, що їхні пристрої були скомпрометовані в минулому. Більш того, лише 21% респондентів сказали, що вони мають повну інвентаризацію всіх IoT-пристроїв сторонніх виробників у своїй мережі, і лише 29% сказали, що вони мають централізовану програму управління ризиками IoT, 9% респондентів відповіли, що їхні організації дуже ефективно зменшують ризики, пов'язані з IoT, на підприємстві. Лише 15% респондентів кажуть, що їхні організації використовують стандартизований підхід для оцінки ризиків IoT для всіх третіх сторін. Лише 13% респондентів стверджують, що їхні організації використовують автоматизовані інструменти для постійного моніторингу ризиків IoT. Лише 12% респондентів кажуть, що їхні організації оцінюють ефективність діяльності з управління ризиками IoT. Лише 11% респондентів кажуть, що їхні

організації використовують ключові індикатори ризику (KIP) для відстеження ризиків IoT [47].

1.3 Вимоги до безпеки протоколів розподілу ключів для інтернет речей

ПРК є важливим компонентом безпеки пристроїв IoT, оскільки вони допомагають забезпечити безпечну передачу конфіденційних даних між пристроями. Існує низка правил, стандартів і законів, розроблених для регулювання безпеки ПРК для пристроїв IoT, зокрема:

1. Інструкції Національного інституту стандартів і технологій (NIST): NIST розробив вичерпні вказівки щодо захисту пристроїв IoT, включаючи вказівки щодо керування та розповсюдження ключів. Схема, наведена в Додатку З, показує стадії розробок публікацій, досліджень, рекомендацій NIST для IoT. Пристрої IoT часто розгортаються в середовищах, які є менш безпечними, ніж традиційні IT-мережі, і тому вони більш вразливі до атак, які використовують слабкі методи керування ключами. Щоб вирішити цю проблему, рекомендації NIST щодо ПРК для пристроїв IoT охоплюють низку тем:

- Використання безпечних ПРК, щоб забезпечити безпечну передачу ключів між пристроями. Це включає використання протоколів, які забезпечують наскрізне шифрування, наприклад протокол TLS, захищених каналів зв’язку для розповсюдження ключів, таких як SSL або VPN. Ці канали мають використовувати надійні алгоритми шифрування та повинні бути захищені брандмауерам та іншими механізмами безпеки.
- Використання надійних алгоритмів шифрування для захисту конфіденційних даних. Це включає використання передових стандартів шифрування, таких як AES, який забезпечує надійне шифрування як для симетричних, так і для асиметричних ключів. Інструкції рекомендують використовувати AES-128 або AES-256, які вважаються найбезпечнішими доступними стандартами шифрування.
- Реалізація ефективних практик керування ключами, включаючи використання функцій виведення ключів і безпечне зберігання ключів. Це гарантує,

що ключі не будуть легко скомпрометовані або вкрадені. Рекомендацією є використання функції виведення ключів, такі як PBKDF2 або «bcrypt», які створені для того, щоб зловмисникам було складніше вгадати ключі, або підбір ключів. Рекомендації також рекомендують використовувати безпечне сховище ключів, наприклад апаратні модулі безпеки (HSM) або довірені платформні модулі (TPM).

- Безпечна генерація ключів. Пристрої IoT повинні використовувати безпечні методи генерації ключів, наприклад генератори випадкових чисел або апаратні генератори ключів. Це допомагає запобігти передбачуваному створенню ключів і зменшує ризик компрометації ключа. Інструкції рекомендують використовувати апаратні генератори випадкових чисел, які вважаються найбезпечнішим методом генерації ключів.
- Регулярна ротація ключів. Пристрої IoT повинні регулярно змінювати ключі, щоб запобігти використанню зламаних ключів. Це включає реалізацію автоматичної ротації ключів, яка допомагає забезпечити регулярне оновлення ключів. Рекомендації рекомендують використовувати короткочасні ключі, які мають меншу ймовірність зламу та можуть допомогти зменшити вплив зламу ключа.
- Сумісність. Рекомендації NIST підкреслюють важливість забезпечення сумісності ключових протоколів розподілу з іншими пристроями та платформами. Це допомагає гарантувати, що пристрої IoT можуть безпечно спілкуватися з іншими пристроями та платформами, і зменшує ризик компрометації ключа через проблеми взаємодії. Інструкції рекомендують використовувати стандартизовані протоколи розповсюдження ключів, такі як протокол обміну ключами в інтернеті (IKE) або протокол Kerberos [50; 51; 52].

2. Стандарт IEEE 802.11i: Стандарт IEEE 802.11i — це протокол бездротової безпеки, який забезпечує безпеку бездротових мереж, у тому числі тих, що використовуються в пристроях IoT. Він забезпечує такі функції безпеки, як шифрування, автентифікація та керування ключами. Стандарт IEEE 802.11i визначає чотири протоколи керування ключами: Pre-Shared Key (PSK), Extensible Authentication Protocol (EAP), Simultaneous Authentication of Equals (SAE) і 4-way

handshake. Кожен із цих протоколів має власні вимоги до безпеки, які необхідно виконати, щоб забезпечити безпеку процесу розповсюдження ключів [53; 54].

3. Стандарти IETF (Internet Engineering Task Force): IETF розробила низку стандартів, пов’язаних із безпекою пристроїв IoT, включаючи стандарти для керування та розповсюдження ключів. Ці стандарти включають протокол Datagram Transport Layer Security (DTLS), який забезпечує безпеку програм на основі датаграм, і CoAP, який забезпечує легкий протокол для зв’язку між пристроями IoT. Також IETF відповідає за розробку стандартів і протоколів для інтернету, включно з тими, що стосуються безпеки IoT. Одним із таких стандартів є "Телеметричний транспортний протокол черги повідомлень (MQTT) версії 5.0", який містить вимоги безпеки для протоколів розповсюдження ключів для IoT. Відповідно до стандарту MQTT 5.0, протоколи розповсюдження ключів для IoT повинні відповідати наступним вимогам безпеки:

- Автентифікація: протоколи розподілу ключів повинні забезпечувати механізм автентифікації пристроїв і користувачів.
- Конфіденційність: протоколи розповсюдження ключів мають гарантувати, що дані шифруються та передаються безпечно, щоб вони не могли бути перехоплені та прочитані сторонніми особами.
- Цілісність: протоколи розповсюдження ключів повинні гарантувати, що дані не були підроблені або змінені під час передачі.
- Незаперечність: протоколи розповсюдження ключів повинні гарантувати, що передача даних може бути відстежена назад до їх джерела, і що відправник не може заперечити, що надіслав дані.
- Контроль доступу: протоколи розподілу ключів повинні забезпечувати механізми контролю доступу до пристроїв і даних IoT.
- Керування ключами: протоколи розповсюдження ключів повинні включати механізми керування ключами, включаючи генерацію, розповсюдження, відкликання та оновлення.
- Стійкість: протоколи розподілу ключів повинні бути розроблені таким чином, щоб протистояти атакам і швидко відновлюватися після будь-яких інцидентів

безпеки. IETF також опублікував кілька відповідних стандартів, наприклад "Протокол безпеки транспортного рівня (TLS) версії 1.3", який забезпечує безпечний канал зв'язку для пристроїв IoT, і "Протокол обміну ключами в інтернеті версії 2 (IKEv2)", який надає безпечний ключ. обмін на пристрої IoT. Загалом IETF зробила значний внесок у безпеку IoT, а її стандарти та протоколи надають цінні рекомендації щодо розробки та впровадження безпечних протоколів розподілу ключів для IoT [55].

4. Загальний регламент захисту даних (GDPR) є нормативним актом, що регулює захист персональних даних в Європейському Союзі (ЄС). GDPR встановлює вимоги щодо безпеки зберігання та передачі персональних даних, включаючи шифрування та протоколи розповсюдження ключів. Стаття 32 GDPR визначає вимоги безпеки для обробки персональних даних, включаючи захист від несанкціонованого доступу та змін. Стаття 5 вимагає відповідної безпеки обробки персональних даних, а стаття 25 вимагає впровадження заходів безпеки з самого початку. Стаття 35 вимагає проведення оцінки впливу на захист даних (DPIA) при обробці персональних даних, включаючи використання пристроїв IoT. Організації, що обробляють персональні дані через IoT, повинні дотримуватися цих вимог. Додатково, стаття 30 вимагає ведення обліку всіх видів обробки даних. GDPR надає велику увагу безпеці персональних даних, зокрема даним, які обробляються через IoT. Організації повинні виконувати вимоги безпеки GDPR, включаючи заходи безпеки, захист даних та проведення DPIA для оцінки і зменшення ризиків [56].

5. Закон Каліфорнії про конфіденційність споживачів (CCPA): CCPA є комплексним законом, що захищає особисту інформацію жителів Каліфорнії. Вимоги CCPA стосуються безпеки зберігання та обробки особистої інформації. Для IoT, CCPA вимагає захисту особистих даних через розділ 1798.81.5(a) та 1798.81.5(b). Розділ 1798.81.5(a) вимагає, щоб пристрої та послуги IoT були розроблені з урахуванням конфіденційності та безпеки інформації. Розділ 1798.81.5(b) вимагає використання розумних функцій безпеки для захисту особистих даних. Загалом, безпека ключових протоколів IoT зосереджена на забезпеченні конфіденційності, цілісності та доступності даних. Відповідність цим вимогам є важливою для захисту конфіденційності та безпеки пристроїв IoT і переданих даних [57].

1.4 Висновки за Розділом №1

У даному розділі було проведено ряд досліджень та отримано наступні результати:

1. Проведено аналіз концепцій типів архітектур, хмарні обчислення, хмару як платформу, проблем інтеграції в хмарні обчислення та викликів для IoT.
2. Проаналізовано ризики для IoT, а саме інцидентів та статистичних даних, тим самим обґрунтовано необхідності впровадження захищених ПРК для підтримки цілісності та конфіденційності даних, що передаються в мережах IoT.
3. Проведено аналіз стандартів та правил, які регулюють вимоги безпеки для ПРК у контексті IoT.

РОЗДІЛ 2
ФОРМУВАННЯ КРИТЕРІЇВ ДО МЕХАНІЗМІВ ЗАХИСТУ

2.1 Поняття протоколу розподілу ключів

Протоколи розподілу ключів (ПРК) — це криптографічні процедури, які дозволяють користувачам встановлювати безпечні канали зв’язку. Ці протоколи включають генерацію та обмін ключами сеансу та автентифікацію повідомлень. У деяких випадках ПРК залучають довірену третю сторону, наприклад Центр розповсюдження ключів (ЦРК), щоб допомогти створити та розповсюдити ключі [58, с. 46].

Метою ПРК в контексті IoT є забезпечення безпечного зв’язку між пристроями з обмеженою обчислювальною потужністю, пам’яттю та енергетичними ресурсами. Використовуючи спрощені протоколи, пристрої IoT можуть створювати безпечні канали для передачі даних і команд, мінімізуючи вплив на продуктивність пристрою та час автономної роботи. Наприклад, Аліса та Боб, два пристрої IoT, потребують безпечного обміну конфіденційними даними. Вони починають із узгодження спільного протоколу, генерації ключів сеансу та обміну ними за допомогою шифрування. Потім вони використовують ці ключі для симетричного шифрування даних. Кожен пристрій перевіряє автентичність отриманих повідомлень, щоб переконатися, що вони надходять від передбачуваного відправника та не були підроблені. Використовуючи протоколи розповсюдження ключів, Аліса та Боб можуть безпечно обмінюватися даними, забезпечуючи конфіденційність та цілісність обмінюваної інформації [58, с. 80-81].

ПРК мають вирішальне значення в системах IoT для забезпечення безпеки та конфіденційності пристроїв і даних. Використовуючи ефективні та легкі протоколи, пристрої IoT можуть створювати безпечні канали зв’язку, які захищають від прослуховування, втручання та інших форм кібератак.

2.2 Асиметрична криптографія

Асиметрична криптосистема, також відома як криптографія з відкритим ключем, використовує різні ключі для різних операцій у криптосистемі, таких як шифрування та дешифрування. Це дозволяє відкрити відкритий доступ до одного з ключів, відомого як відкритий ключ, без шкоди для секретності відповідного закритого ключа.

У традиційній моделі асиметричного шифрування операція шифрування використовує відкритий ключ для обчислення зашифрованого тексту із повідомлення. Зворотна операція дешифрування використовує відповідний закритий ключ для відновлення вихідного повідомлення. Схему операції шифрування та дешифрування в асиметричній криптографії наведено на рисунку 2.1.

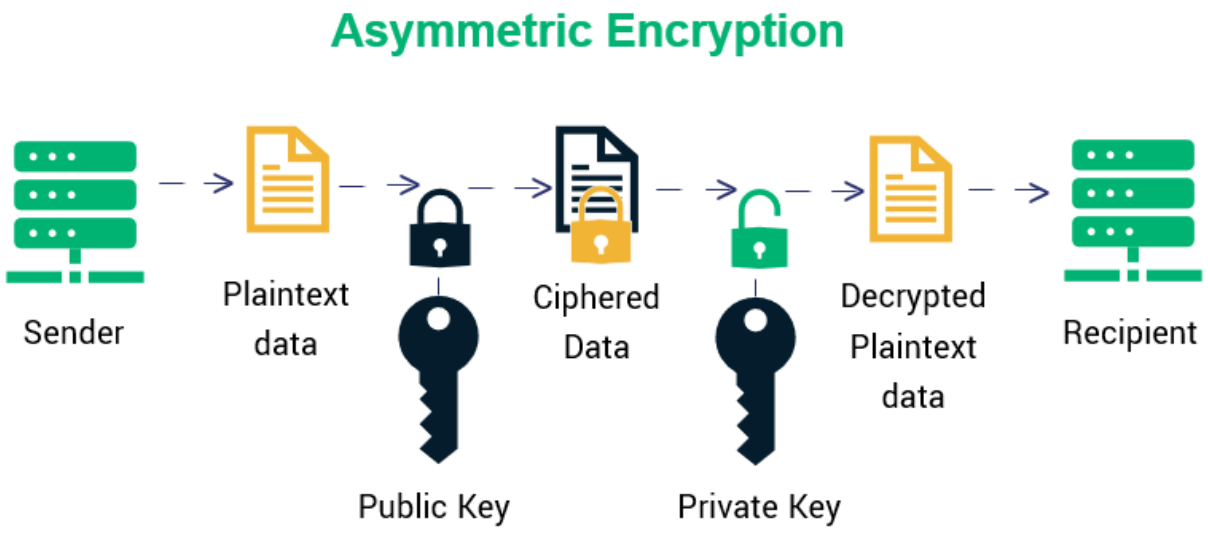

Рисунок 2.1 – Асиметрична криптографія

Сучасне використання асиметричної криптографії виходить за рамки шифрування та дешифрування та включає схеми цифрового підпису та протоколи узгодження ключів. Ці схеми передбачають використання ключа, який можна оприлюднити для певних операцій, тоді як інші вимагають, щоб відповідний секретний ключ залишався конфіденційним.

Асиметричні криптосистеми знаходять широке застосування, особливо в сценаріях, коли сторони спочатку не мають зручного засобу для обміну секретами. Примітні приклади включають шифрування Діффі-Хеллмана, RSA і криптографію еліптичних кривих (ECC) [59, с. 46].

2.3 Криптографічний алгоритм Рівеста-Шаміра-Адлемана

RSA – це криптографічний алгоритм, який використовує асиметричне шифрування. Цей алгоритм отримав свою назву на честь трьох винахідників: Рона Рівеста, Аді Шаміра і Леонарда Адлемана, і був запропонований в 1978 році. Розробники успішно втілили ідею односторонніх функцій зі секретом у даному алгоритмі. Ключі для шифрування та розшифрування є функціями двох великих простих чисел, кожне з яких має більш як 100...200 десяткових цифр. Відновлення відкритого тексту з шифрованого тексту та відкритого ключа еквівалентно факторизації числа на два великі прості множники. Схему роботи алгоритму RSA наведено на рисунку 2.2.

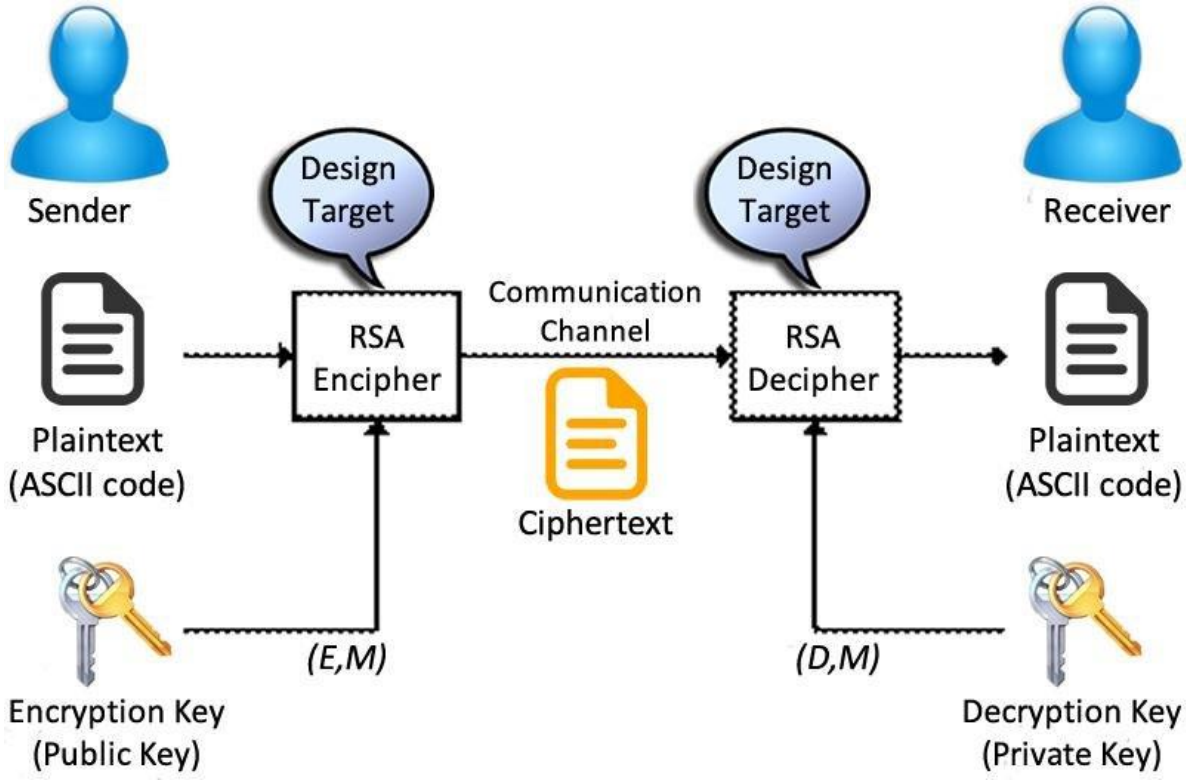

Рисунок 2.2 – Схема роботи алгоритму RSA

Алгоритм RSA протистояв численним криптоаналітичним викликам протягом багатьох років. До них відносяться атаки на синхронізацію, атаки на бічні канали, ”padding oracle” атака [60], атаки грубої сили, загрози від квантових обчислень тощо. Незважаючи на ці проблеми, RSA залишається безпечним, якщо його впроваджено з відповідними розмірами ключів і контрзаходами [61; 62, с. 55]. Принцип алгоритму RSA:

Для генерації двох ключів використовуються два випадкові великі прості числа, p і q. Для максимальної безпеки рекомендується обирати p і q однакової довжини. Обчислюється добуток: $n = p \times q$.

Потім випадково обирається ключ шифрування, e, такий, що e і $(p-1) \times (q-1)$ є взаємно простими. Нарешті, застосовується розширений алгоритм Євкліда для обчислення ключа розшифрування, d, такого, що

$$ed \equiv 1 mod((p-1) \times (q-1)), \tag{2.1}$$

або

$$d = e^{-1} mod((p-1) \times (q-1)). \tag{2.2}$$

Потрібно зазначити, d і n також є взаємно простими. Числа e і n є публічним ключем; число d є приватним ключем. Два простих числа, p і q, більше не потрібні. Вони мають бути викинуті, однак ніколи не розголошуватись.

Для шифрування повідомлення m спочатку розбивається на числові блоки, менші за n (за допомогою двійкових даних обирається найбільша степінь числа 2, менша за n). Іншими словами, якщо як p, так і q є простими числами з 100 цифрами, то n матиме приблизно 200 цифр, а кожен блок повідомлення, m_i, повинен бути трохи меншим за 200 цифр. (Якщо потрібно зашифрувати фіксовану кількість блоків, доповнюється кілька нулів зліва, щоб забезпечити, що вони завжди будуть меншими за n). Зашифроване повідомлення, c, складається з блоків повідомлень такого ж розміру, c_i, приблизно однакової довжини. Формула (2.3) шифрування:

$$c_{\text{i}} = m_{\text{i}}^{\text{e}} mod n \tag{2.3}$$

Для розшифрування повідомлення береться кожний зашифрований блок c_i і обчислюється формулою (2.4).

$$m_{\text{i}} = c_{\text{i}}^{\text{d}} mod n, \tag{2.4}$$

Оскільки

$$c_i{}^d (mod n) \equiv m_i{}^{ed} (mod n) \equiv \left(m_i{}^{k(p-1)(q-1)+1}\right) mod n \equiv \left(m_i * m_i{}^{k(p-1)(q-1)}\right) mod (p \times q) \equiv m_i. \tag{2.5}$$

Повідомлення так само може бути зашифроване з використанням d і розшифроване з використанням e. Вибір є довільним [62, с. 55-58; 63, с. 620-622].

Стійкість алгоритму RSA базується на складності розкладання великих чисел на множники [63, с. 1; 64, с. 620-622]. Проте, це твердження не є математично доведеним і є гіпотетичним. Існує можливість, що існують інші методи криптоаналізу RSA, які не вимагають розкладання чисел на множники. На прикладі атаки грубої сили, був проведений процес імплементування даного методу зламу в алгоритм RSA. В підсумку, ”атака грубої сили” менш ефективна загроза для RSA, але все ще загрожує системі [63, с. 1]. Для забезпечення безпеки RSA важливо використовувати великі випадкові числа та дотримуватись рекомендацій щодо параметрів, ускладнюючи криптоаналітичні атаки та забезпечуючи надійний захист даних [62, с.59].

Крім того, RSA можна використовувати для створення та перевірки цифрових підписів, що дозволяє перевірити автентичність та цілісність документів [62, с. 94]. Технологія використання електронного цифрового підпису передбачає наявність мережі абонентів, які обмінюються електронними документами. У цьому випадку кожен абонент використовує окрему пару ключів – К1 і К2 – для створення електронного підпису. Секретний ключ К2 відомий тільки користувачу, а його

ідентифікаційний номер ID та ключ К1 розміщуються в загальнодоступному каталозі для інших абонентів мережі. Це дозволяє будь-якому абоненту мережі перевіряти правильність цифрового пімвадпису документів, які вони отримують від власника. Значення ідентифікаційного номера використовуються у деяких алгоритмах створення підпису. Найпоширенішою системою для створення електронного підпису є система, яка базується на алгоритмі RSA. Узагальнена схема для створення та перевірки цифрового підпису RSA [65] показана на рисунку 4.2.

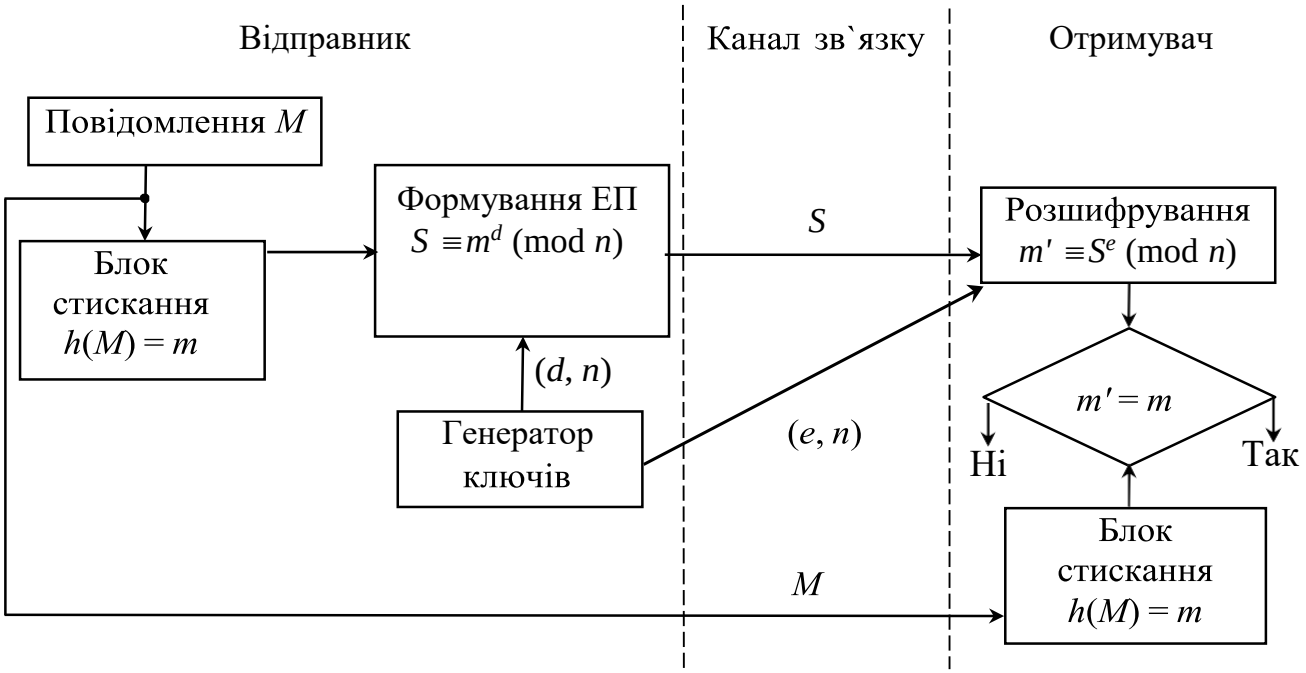

Рисунок 2.3 – Узагальнена схема цифрового підпису RSA

Пояснення схеми:

1. Вибираються великі прості числа p і q;
2. обчислюється число $n = p \times q$;
3. функція Ейлера $\varphi(n) = (p - 1)(q - 1)$;
4. $e < \varphi(n)$, e(відкритий ключ К1) – взаємно просте з $\varphi(n)$;
5. обчислюється число d (К2 = d), обернене до e.
6. Ключ (n, e) розміщується у відкритому каталозі, а ключ d зберігається у власника документа.

7. Якщо відправник хоче підписати повідомлення *M* перед надсиланням, то спочатку текст *M* стискається за допомогою геш-функції *h*, отримуючи ціле число *m*: $h(M) = m$.

8. Потім відправник зашифровує *m* своїм секретним ключем *d*: $S \equiv m^d \pmod n$.

9. Пара чисел *(M, S)* відправляється отримувачу як електронний документ *M* та підписується підписом *S*.

10. Адресат, отримавши підписаний документ *(M, S)*, перевіряє значення m двома способами:

a. відновлює геш-значення *m'*, використовуючи криптографічне перетворення підпису *S*, використовуючи відкритий ключ *e*:

$m' \equiv S^e \pmod n$.

b. одержувач обчислює значення гешування отриманого повідомлення *M* за допомогою тієї ж геш-функції *h*: $h(M) = m$.

11. Якщо обидва значення збігаються $m' = m$, тобто дотримується рівність $S^e \pmod n = h(M)$, тоді отримувач визнає пару *(M, S)* справжнім підписом документа.

У схемі підпису RSA використовуються геш-функції з родини MD. Відкритий ключ e виступає як ідентифікатор підписувача [62, с. 94-95].

Однак, обчислення ключів RSA для цифрового підпису вимагає дотримання багатьох умов, щоб уникнути можливості фальсифікації підпису. Навіть теоретично не можна допускати такі ризики під час підписання важливих документів. Алгоритм цифрового підпису RSA вразливий до мультиплікативної атаки, яка дозволяє криптоаналітику створювати підписи без знання секретного ключа, використовуючи результати гешування вже підписаних документів [62, с. 96]. Однак, важливо, щоб геш-функції не мали колізій, тобто не було можливості знайти дві різні вхідні дані, які дають однакові геш-значення. Криптографічно стійка геш-функція називається такою, для якої не існує ефективного алгоритму, що знайде колізії або знайде вхідні дані, що дають заданий геш. Відсутність колізій сама по собі не гарантує практичну стійкість геш-функцій [66, с. 141]. Важливою властивістю є відсутність кореляції, коли неможливо знайти пару вхідних даних, для яких вага Хеммінгу геш-значень

буде меншою, ніж вага Хеммінгу для іншої пари даних. Свобода від кореляції є сильнішою вимогою для криптографічної стійкості геш-функцій, ніж відсутність колізій [62, с. 90].

2.4 Криптографія еліптичних кривих

ECC базується на математиці еліптичних кривих і особливо корисний у сценаріях, коли обчислювальні ресурси обмежені.

Генерація ключа в ECC включає вибір конкретної еліптичної кривої та базової точки на цій кривій. Закритий ключ, який представляється цілим числом, обирається випадковим чином в певному діапазоні. Відкритий ключ, що представляє собою точку на еліптичній кривій, отримується шляхом множення базової точки на закритий ключ [62, с. 70].

Загальний вид кубічного рівняння для еліптичних кривих:

$$y^2 + axy + by = x^3 + cx^2 + dx + e, \qquad (2.6)$$

де a, b, c, d, e – дійсні числа, що задовольняють певним умовам. Визначення еліптичної кривої також включає елемент в нескінченності, який позначається як точка в нескінченності і є нульовим елементом. Дане рівняння має назву кубічний або рівнянням третього порядку, бо найвищий ступінь в них дорівнює трьом [62, с. 70-72].

У реальних криптосистемах використовується рівняння: $y^2 \equiv x^3 + ax + b \pmod p$, де $a, b \in GF(p)$. Група $E(GF(p))$ складається з усіх точок (x, y), де $x, y \in GF(p)$ і вони задовольняють рівняння, включаючи точку в нескінченності O. На рисунку 2.4 наведено приклад вигляду еліптичної кривої.

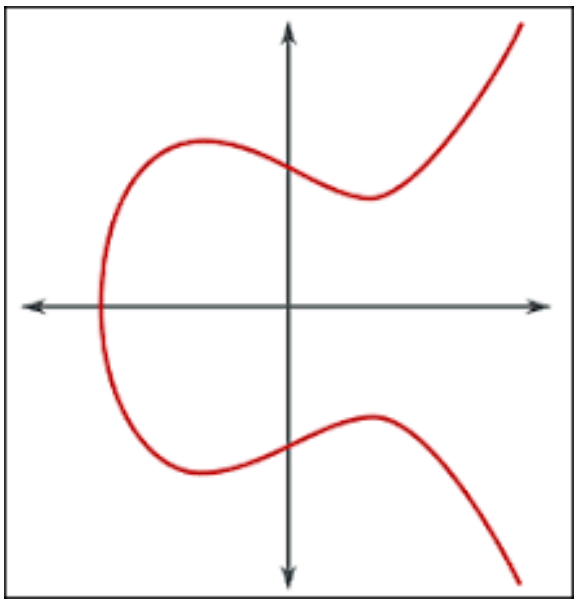

Рисунок 2.4 – Приклад еліптичної кривої

Множина $E_p(a, b)$ складається з усіх точок (x, y), де $x \geq 0$, $y < p$, і вони задовольняють рівняння $y^2 \equiv x^3 + ax + b \ (mod\ p)$, включаючи точку в нескінченності. Кількість точок в $E_p(a, b)$ позначається як $\#E_p(a, b)$ і має значення для криптографічних застосувань еліптичних кривих.

Операція додавання над точками з $E(GF(p))$ алгебраїчно може бути описана наступними правилами:

1. $P + O = O + P = P$.

2. У випадку $P = (x, y)$, *то* $P + (x, -y) = O$. Точка $(x, -y)$ є оберненою до точки P і позначається як $-P$, і $(x, -y)$ лежить на еліптичній кривій і належить $E_p(a, b)$.

3. Якщо $P = (x_1, y_1)$ і $Q = (x_2, y_2)$, то $P + Q = (x_3, y_3)$, де координати x_3 та y_3 визначаються відповідно до правил:

$$x_3 \equiv \lambda^2 - x_1 - x_2 (mod p); \tag{2.7}$$

$$y_3 \equiv \lambda(x_1 - x_3) - y_1 (mod p), \tag{2.8}$$

де λ виконується при умові:

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \text{ якщо } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, \text{ якщо } P = Q. \end{cases} \quad (2.9)$$

Кутовий коефіцієнт січної, проведеної через точки $P = (x_1, y_1)$ та $Q = (x_2, y_2)$, позначається як λ. Коли $P = Q$, січна перетворюється на дотичну, що пояснює наявність двох формул для обчислення λ [62, с. 72].

Рекомендації до вибору параметрів еліптичної кривої, які використовуються у криптографічних задачах, при виборі коефіцієнтів a, b та модуля p мають наступний вигляд: головним критерієм вибору є нездатність здійснити певні атаки, які можуть бути спрямовані на певні класи кривих. Існує альтернативний підхід, який полягає у систематичному конструюванні кривої з певними властивостями, що зазвичай є ефективнішим з обчислювальної точки зору. Для реалізації цього підходу існують спеціальні методи, але отримані криві фактично вибираються з обмеженого набору, що може викликати підозри на наявність деяких специфічних властивостей, які можуть стати основою для розробки алгоритмів злому в майбутньому.

Кроки процесу формування випадкової кривої:

1. Випадково обирається просте число p. Бітова довжина числа p, позначена як $t=|log\ p|+1$, повинна бути достатньою для унеможливлення застосування загальних методів знаходження логарифмів на кривій, які мають високу обчислювальну складність $T(2^{t/2})$. На сьогоднішній день величина $t = 128$ біт (чотири машинні слова на 32-бітових комп'ютерах) є недостатньою, оскільки існують повідомлення про злам відповідних кривих. Іншим міркуванням є те, що стійкість шифру на еліптичній кривій повинна бути не меншою, ніж у блокового шифру AES (Advanced Encryption Standard). Стійкість AES забезпечується повною довжиною ключа, яка становить 128, 196 або 256 біт. Оскільки стійкість шифру на еліптичній кривій визначається величиною $t/2$, довжина еліптичних модулів кривих повинна складати відповідно 256, 392 і 512 біт.

2. Обираються випадкові числа a і b таким чином, щоб вони не були кратними модулю p і рівняння $E_p(a, b)$ не дорівнювало нулю. Параметр b не впливає

на обчислення композиції точок, тому іноді рекомендується вибирати випадкове значення лише для b, встановлюючи його як невелике ціле число. Наприклад, стандарт FIPS 186-2 США рекомендує використання кривих з параметром $a=-3$, що спрощує обчислення.

3. Визначається кількість точок на кривій n, що позначається як $\#E_p(a, b)$. Важливо, щоб n мав великий простий дільник q, найкраще, якщо n = q. Якщо n має малі множники, то на кривій $E_p(a, b)$ існує багато малих підмножин з власними генераторами, і алгоритм Поліга-Хеллмана швидко обчислює логарифми на кривій через логарифми в цих малих підмножинах. Якщо пошук кривої з $n = q$ займає багато часу, можна допустити $n = h \times q$, де h – невелике число. Стійкість криптосистеми на еліптичній кривій визначається кількістю елементів q у підмножині точок кривої, а не модулем p. Але якщо множник h є невеликим числом, то q має той же порядок, що і p. Якщо n не задовольняє вимогам, проводимо процес обчислення з кроку 2.

4. Проводиться процес перевірки нерівності $(p^k - 1) \bmod q \neq 0$ для всіх k, $0 < k < 32$. Якщо нерівності не виконуються, проводимо процес обчислення з кроку 2. Ця перевірка запобігає можливості атаки Менезеса-Окамото-Ванстона (MOV-атаки) і виключає аномальні криві та криві з $\#Ep(a, b) = p - 1$. Метод MOV та певні види кривих дають змогу спростити обчислення логарифмів на кривій.

5. Перевірка нерівності $q \neq p$. Якщо нерівність не виконується, повторюється крок 2. Криві з $q = p$ є аномальними та являють собою ефективні методи обчислення логарифмів.

6. Отримуються параметри кривої для криптографічних застосувань: параметри p, a, b, кількість точок n та розмір підмножини точок q. Зазвичай проводиться пошук точку G – генератор підмножини q. Якщо $q = n$, то будь-яка точка (крім O) є генератором. Якщо $q < n$, випадковим чином обираються точки G', доки не знаходиться $G = [n/q]G' \neq O$. Щоб отримати випадкову точку на кривій, береться випадкове число $x < p$, обчислюється $e \equiv (x^3 + ax + b) \bmod p$ і проводиться пошук квадратного кореня, як показано у формулі (2.10).

$$y = \sqrt{e} \bmod p \qquad (2.10)$$

Якщо корінь існує, то отримується точка *(x, y)*, інакше береться інше число *x*.

Завдання, яке вирішує криптоаналітик при використанні криптосистеми на основі еліптичних кривих, називається завданням дискретного логарифмування на еліптичній кривій. Це завдання полягає в пошуку унікальної точки *x* такої, що $P = xQ$, де *P* і *Q* – точки на еліптичній кривій порядку *n* [62, с. 74-76].

Шифрування ECC базується на обміні ключами Діффі-Хеллмана [62, с. 67], який дозволяє двом сторонам установлювати спільний секрет через незахищений канал [63, с. 686-688].

У початковому етапі вибирається просте число p параметрів *a* і *b* для еліптичної кривої, що утворюють еліптичну групу точок $E_p(a, b)$. З цієї групи обирається генеруюча точка *G*. Щоб забезпечити стійкість криптосистеми, найменше число *n*, для якого $nG = O$, повинно бути простим. Параметри $E_p(a, b)$ та *G* криптосистеми є загальними для всіх учасників системи. Обмін ключами між учасниками A та B відбувається наступним чином:

1. Учасник A обирає особистий ключ k_a, менше за *n*, далі генерує відкритий ключ $Y_a = k_aG$, який є точкою з $E_p(a, b)$.
2. Аналогічно, учасник B обирає особистий ключ k_b і обчислює відкритий ключ $Y_b = k_bG$.
3. Учасник A генерує секретний ключ $K = k_aY_b$, а учасник B генерує секретний ключ $K = k_bY_a$.

Обидва ключі дають один і той же результат, як наведено у формулі (2.11).

$$k_aY_b = k_a(k_bG) = k_b(k_aG) = k_bY_a. \qquad (2.11)$$

Слід зауважити, що загальний секретний ключ представляється парою чисел. Якщо цей ключ використовується як сеансовий ключ для шифрування, потрібно згенерувати одне значення з цієї пари чисел. Наприклад, можна використовувати координату x або певну функцію від *x*.

Є різні підходи до шифрування/розшифрування на основі еліптичних кривих. Наприклад, першим завданням є зашифрування відкритого тексту *M*, яке буде представлено координатами *x* і *y* точки P_M. Зашифрований текст C_M буде складатися з пари точок: $C_M = (rGP_M + rY_b)$, де *r* – випадкове ціле число, вибране учасником А.

Шляхом до розшифрування шифротексту C_M є множення першої точки в парі на секретний ключ та віднімання результату від другої точки:

$$P_M + rY_b - k_b(rG) = P_M + r(k_bG) - k_b(rG) = P_M \quad (2.12)$$

Учасник А маскує повідомлення P_M, додаючи до нього rY_b. За допомогою "підказки", яку має учасник В, можна відновити оригінальне повідомлення, прибравши маску $P_M + rY_b - k_b\ (rG) = P_M$, маючи особистий ключ k_b. Знайдення значення *r* за відомими *G* і *rG* є складною задачею, відомою як проблема логарифмування на еліптичній кривій.

Безпека криптографічного підходу на основі еліптичних кривих залежить від складності вирішення проблеми логарифмування, тобто знаходження значення *r* за відомими *rP* і *P* [62, с. 76-77]. Один з найшвидших відомих методів для розв'язання цієї проблеми є розширений метод Полларда [67]. Також були проведені кроки з розробки алгоритму, який покращує продуктивність множення еліптичної кривої за рахунок скорочення операцій і усунення попереднього обчислення. Практичні переваги алгоритму, включаючи підвищену швидкість і ефективність, роблять його добре придатним для середовищ з обмеженими ресурсами, а його математичні пояснення та схеми дають цінну інформацію в області ефективних операцій з еліптичною кривою [68].

2.5 Переваги та недоліки існуючих рішень

У 2004 році було порівняно RSA та ECC з точки зору характеристик продуктивності, таких як розмір ключа, час виконання (продуктивність генерації ключів, продуктивність генерації підпису та продуктивність перевірки підпису).

Отримані результати доводять, що генерація ключів у RSA відбувається значно повільніше, ніж у ECC. Також було визначено, що ECC швидше, ніж RSA, створює цифровий підпис, але повільніше перевіряє цифровий підпис (особливо з великою довжиною ключа). Таким чином, RSA може бути найкращим вибором для додатків, які вимагають перевірки повідомлень частіше, ніж генерування підпису [69].

У [70] є оглядова стаття, де було порівняно RSA та ECC на основі аналізу даних, зібраних із технічних звітів та дослідження літератури. Згідно з цим документом, ECC має менший коефіцієнт вартості, ніж RSA. Крім того, ECC порівняно з RSA може забезпечити однаковий рівень безпеки з меншими розмірами ключів. Отже, коли обчислювальне навантаження не збільшується, ECC більше пропонується для підвищення безпеки та вищої швидкості.

Дослідження впровадження ECC у вбудовану в програму iOS було проведено в [71], щоб порівняти показники ефективності ECC у бездротовому середовищі з RSA. З цього документа він визначає, що RSA порівняно з ECC має в десять разів більше обчислювальних витрат, ніж ECC. Розмір пар ключів і параметрів систем в ECC менше, ніж у RSA. Оскільки на тому самому рівні безпеки RSA потребує ключа набагато більшого розміру, ECC може значно заощадити пропускну здатність, ніж RSA. Згідно з цим документом, генерація ключів ECC є швидшою, ніж RSA, і ECC набагато ефективніша для невеликих пристроїв порівняно з RSA.

Також були представлені деякі проблеми безпеки при проектуванні захищених вбудованих систем і провели порівняння між RSA та ECC у [72]. Результат порівняння цього документу показує значну різницю між RSA та ECC з точки зору часу виконання. Базуючись на цьому документі, ECC, застосовуючи менші ключі та надаючи вищий рівень безпеки, може заощадити витрати на продуктивність, такі як споживання пам’яті, витрати на обчислення та потужність обробки. ECC може бути реалізований на менших чіпах для швидшої та швидкої роботи з криптографією з меншими витратами, через що пристрій виділяє незначну кількість тепла та споживає менше енергії. Різниця в розмірі ключа RSA зробила його менш придатним для систем реального часу, тоді як ECC із меншим розміром ключа має досить складну криптографію та підходить для обмежених вбудованих систем у реальному часі.

Крім того, зламати ECC у близько 10 000 разів складніше порівняно з аналогічним 2048-бітним RSA. Однак автори також зазначили, що ECC має певні обмеження щодо зменшення резервного живлення батареї, меншої потужності процесора та малої пам’яті, що ускладнює ефективне впровадження.

У [73] загальне впровадження ECC для систем керування Smart Parking для оптимізації паркувальних місць у місті забезпечило рішення, яке захищає конфіденційність користувачів. Автор виділяє деякі переваги ECC:

- ECC перевершує RSA у стриманому режимі середовища щодо споживання енергії, вимог до пам’яті та часу обчислень.
- ECC досягає такого ж рівня безпеки, як і RSA з меншими розмірами параметрів.
- ECC використовує менші розміри повідомлень, що призводить до коштують дешевше і можуть бути краще доставлені.

Згодом, як пункт у цьому протоколі, який можна обговорити, є попереднє завантаження використовуваної еліптичної кривої, включаючи її параметри, у пам’ять пристрою. Таким чином, втручання та фізичні атаки можуть скомпрометувати еліптичну криву та її налаштування [73, с. 4].

У дослідженні [74] перевірили ECC-BROSMAP (протокол захищеного мобільного агента на основі широкомовної передачі), застосовуючи Scyther [75, с. 1], а потім порівняли його з BROSMAP з точки зору вартості обчислень і часу виконання. Згідно з цим документом, час виконання в ECC із розмірами ключів 224 і 256 приблизно вдвічі і в чотири рази швидший, ніж у RSA 2048 і 3072 відповідно, а витрати на обчислення також у ECC ефективніші, ніж у RSA. ECC-BROSMAP може забезпечити той самий рівень безпеки, що й RSA-BROSMAP, але з більшою ефективністю та легкий. ECC-BROSMAP усунув асиметричне шифрування, використовуючи ключі меншого розміру, і застосував лише симетричне шифрування для поєднання з ключами ECC.

Було також представлено порівняльний аналіз RSA та ECC у [76], де порівнюється проміжок часу шифрування та дешифрування в RSA та ECC на трьох зразках вхідних даних (8, 64 та 256 біт). Згідно з їхніми експериментами, було

помічено, що ECC є більш застосовним та ефективним у дешифруванні, але в шифруванні є повільним, тоді як RSA є більш придатним у шифруванні та повільним у дешифруванні. Цей аналіз також показує, що ECC може займати менше пам’яті. Загалом на основі їх результатів було встановлено, що ECC перевершує RSA з точки зору безпеки та ефективності роботи з меншими параметрами та більше підходить для пристроїв з обмеженим ресурсом.

В роботі [77] було запропоновано схему безпеки на основі ECC для запобігання атакам підслуховування в хмарних середовищах, а потім порівняли її з характеристиками RSA. На основі цієї статті визначено, що запропонована система з використанням ECC перевершує RSA і є набагато швидшою у практичному застосуванні. Запропонована схема на основі ECC набагато швидша, ніж RSA, як для шифрування, так і для дешифрування, і корисна для захисту приватних даних користувачів. Це також чудовий варіант для механізму захисту від перехоплювачів у хмарних службах зберігання даних. Загалом, коли кількість користувачів і їхні дані постійно зростають, ECC чудово підходить для використання в хмарних службах зберігання.

Інше порівняння між алгоритмами представлено в [78]. Результати, отримані в цьому документі, також визначають, що ECC є більш ефективним, ніж RSA, щодо часу виконання (шифрування та дешифрування) і часу генерації ключа. Міра вимог до пам’яті, необхідна в ECC, менша, ніж RSA.

У 2018 році дослідник порівняв алгоритм цифрового підпису еліптичної кривої (ECDSA) і RSA щодо розміру ключа, споживання енергії, середнього часу під час виконання на вузлі IoT з обмеженими ресурсами. Результати, отримані в цій статті, показують, що ECDSA має набагато кращі результати щодо енергоефективності та часу відгуку, а також підходить для захисту пристроїв IoT з обмеженими ресурсами [79].

У [80] було запропоновано порівняння ECDSA та RSA як двох найбільш використовуваних алгоритмів аутентифікації TLS (Transport Layer Security). Результати визначають величезні відмінності в однаковому рівні безпеки між RSA та ECC з точки зору розміру ключа та споживання енергії. ECC перевершує RSA щодо

значень даних пропускна здатність і споживання енергії. Якщо потрібен вищий рівень безпеки, розмір ключа в RSA буде непрактичним, але ECDSA може забезпечити розгортання IoT.

Також було представлено у дослідженні [81] інше порівняння RSA та ECC у шлюзі проти туману та двох кінцевих пристроях проти туману. Досягнуті результати засвідчили, що ECC перевершує RSA з точки зору пропускної здатності даних і споживання енергії на всіх рівнях безпеки.

Наприклад, при рівні безпеки 128 біт RSA споживає вдвічі більше енергії, ніж криві secp256r1 і secp256k1. У sepc256k1 «k» означає Koblitz, а «r» у sepc256r1 — випадковий. Еліптична крива Кобліца має деякі унікальні особливості, які роблять групову операцію більш ефективною [82]. Також зроблено висновок, що в той час як рівень безпеки підвищується, розміри ключів, необхідні для ECC, збільшуються більш лінійно, ніж розміри, необхідні для RSA. Як згадувалося в попередній статті, криві ECC – це залежність кривої та платформи від реалізованих оптимізацій (крива secp256r1 мала кращі результати порівняно з secp224r1) [81].

У 2018 році представили CoAP, що використовує ECC, і порівняли переваги ECC з RSA. Результати, отримані в цій статті, виявили, що запропонований безпечний CoAP з ECC може подолати CoAP із застосуванням RSA щодо економії енергії на 47%. Крім того, при застосуванні RSA енергоспоживання батареї відбувається швидше, ніж CoAP [83].

У [84] схема реалізації RSA була запропонована Каеді та ін. протистояти атакам аналізу потужності. У цій статті згадуються деякі недоліки RSA в IoT. Слабкі сторони, такі як додаткове обчислювальне навантаження, високе енергоспоживання, численні контрзаходи в бічному каналі, які спричиняють потребу в додатковому обсязі пам’яті, збільшення часу виконання, також потребують TRNG (генератор справжніх випадкових чисел), який обчислює фактори засліплення, на завершення RSA є не підходить для підходу IoT.

Відповідно до статті [85], автори порівняли продуктивність двох алгоритмів шифрування, ECC і RSA, в пристроях IoT. Був проведений аналіз згідно робіт та

статей, частина яких була описана в розділі 3.4.1, за наступними показниками ефективності:

- Вимоги до пам’яті: різні методи шифрування потребують різного розміру пам’яті для впровадження. Вимоги до пам’яті залежать від кількості операцій, які виконує алгоритм, розміру ключа, використовуваних векторів ініціалізації та типу послуг. Використовувана пам'ять впливає на вартість системи. Бажано, щоб необхідна пам'ять була якомога меншою [85, с. 13].
- Споживання енергії: повна енергія, необхідна для алгоритму шифрування/ дешифрування. Він оцінюється відповідно до пропускної здатності алгоритмів шифрування/дешифрування.
- Розмір ключа: у методології шифрування керування ключами є важливим аспектом, який визначає спосіб шифрування даних. Втрата зображення та коефіцієнт шифрування базується на цій вирішальній довжині. Симетричний алгоритм використовує змінну довжину ключа, яка є довшою. Кожен алгоритм використовує певне число довжини ключа, яке використовується як початкове число в блоці процесу [85, с. 13].
- Час генерації підпису: час генерації підпису, який використовує закритий ключ для створення цифрового підпису [86].
- Час перевірки підпису: Перевірка підпису — це техніка для порівняння підписів і підтвердження особи (використовується банками, спецслужбами та високопоставленими установами) [62, с. 91]. Час перевірки означає час перевірки підпису, коли користувач отримує доступ до системи [87].
- Час генерації та виконання ключа: цей час відноситься до часу, необхідного функції генерації ключа для створення ключів. Усі ці функції виробляють різний час залежно від розміру текстових файлів і довжини ключа в будь-якому алгоритмі [88, с. 444].
- Час шифрування: це повний час, необхідний для створення зашифрованого тексту з відкритого тексту. Цей час використовується для обчислення пропускної здатності зашифрованого алгоритму (забезпечує швидкість шифрування).

- Час дешифрування: це загальний час, необхідний для створення відкритого тексту із зашифрованого тексту. Цей час використовується для обчислення пропускної здатності дешифрованого алгоритму (забезпечує швидкість дешифрування) [89].

2.5.1 Вимоги до пам’яті

Згідно вимог до пам’яті, на тому самому рівні безпеки ECC потребує менше використання пам’яті, ніж RSA. Згідно таблиці 2.1, ECC пропонує той самий рівень безпеки, що й RSA, при цьому використовуючи менше пам’яті.

Таблиця 2.1

Вимоги до пам’яті для RSA та ECC

Стаття	Рівень безпеки	Розмір ключа		Вимоги до пам’яті(байт)	
		RSA	ECC	RSA	ECC
Модель безпеки для збереження конфіденційності медичних великих даних у хмарі охорони здоров’я з використанням обчислювальної системи Fog із криптографією на основі пар	80	512	106	157	108
	112	768	132	236	117
	128	1024	160	313	125
	160	2048	210	621	140

2.5.2 Споживання енергії

У Додатку Б, Додатку В класифіковано показники енергоспоживання, зібрані для ECC і RSA на різних рівнях безпеки та з різними розмірами ключів.

ECC із використанням меншого розміру ключа та споживання енергії долає RSA. Оскільки ECC економить пропускну здатність більше, ніж RSA і перевершує RSA на 47% з точки зору економії енергії, є більш ефективним. ECC може споживати менше ресурсу батареї та обчислювальної потужності [90]. Щоб деталізувати це, у додатку Б, ECC із 192-бітним розміром ключа забезпечує той самий рівень безпеки, однак вимагає трохи більше 9 МВт-год порівняно з 17,86 МВт- год у RSA. Коли рівень

безпеки підвищився до 128, розмір ключа в RSA зріс у три рази з 1024 до 3072, у той час як розмір ключа ECC збільшився на 64 біти, а його енергоспоживання майже в чотири рази більше, ніж у ECC. Крім того, для приблизно однакового енергоспоживання (RSA з приблизно 21,55 мВт/год і ECDSA з 22,26 мВт/год) ECDSA з алгоритмом ECC забезпечує вищий рівень безпеки, ніж RSA (192- розрядний рівень безпеки в ECDSA порівняно з RSA з алгоритмом ECC 112-біт). Важливим висновком ECC є те, що рівень безпеки, реалізований кривою, ніколи не пропорційний її продуктивності. Як показано в додатку Б, крива з secp256 споживала менше енергії та показала вищі значення пропускної здатності порівняно з кривою secp224r1, яка є найслабшою. Причина цього полягає в тому, що існує оптимізація програмного забезпечення, яка була реалізована для прискорення алгоритмів з точки зору математичних операцій, оскільки продуктивність ECC залежить від кривої та платформи. Ця оптимізація включає в себе розробку спеціалізованих алгоритмів та використання апаратних ресурсів, що дозволяють ефективніше виконувати необхідні обчислення. У Додатку В можна побачити, що в обох алгоритмів, за рахунок підвищення частоти з 80 МГц до 240 МГц, енергоспоживання зменшується. Це свідчить про те, що оптимізація програмного забезпечення, спрямована на покращення продуктивності ECC, також має позитивний вплив на енергоефективність системи. За таких умов ECC на кожному рівні безпеки та частоті споживає менше енергії та працює краще, ніж RSA.

2.5.3 Розмір ключів

Для успішної реалізації алгоритму криптографії, одним з першочергових критеріїв є вибір розміру ключів. Важливо враховувати, що більші розміри ключів забезпечують вищий рівень безпеки, але водночас є більш витратними з точки зору обчислювальних ресурсів та пам'яті. Тому, щоб досягти найменшого можливого розміру ключа при збереженні високого рівня захисту, необхідно ретельно вибирати параметри ключів [91]. У випадку RSA, рекомендований розмір ключів постійно зростає, починаючи від 1024 біт і досягаючи 15360 біт, для забезпечення належної

надійності криптографії. З іншого боку, алгоритм ECC може забезпечити такий же рівень безпеки та криптографічної стабільності з меншими розмірами ключів. ECC вдосконалює безпеку, зменшуючи обчислювальні вимоги і ефективніше використовуючи ресурси. Завдяки своїм меншим розмірам ключів, ECC стає особливо привабливим для пристроїв IoT, які мають обмежену пам'ять або обчислювальну потужність. Крім того, ключі меншого розміру можуть забезпечити швидшу рукостискання SSL, що прискорює завантаження веб-сторінок, а також потужніший рівень захисту [92]. Відповідно до таблиці 2.2, порівнюючи алгоритми з однаковим рівнем безпеки, можна побачити, що ECC вимагає меншого розміру ключа порівняно з RSA. Це означає, що при використанні ECC можна досягнути економії ресурсів та більш ефективного використання криптографічних алгоритмів, забезпечуючи при цьому необхідний рівень захисту для системи.

Таблиця 2.2

Розмір ключів для RSA та ECC

Рівень безпеки	Розмір ключа		Протокол прикладного рівня CoAP з урахуванням безпеки для IoT із використанням еліптичної криптографії	Криптографія еліптичної кривої для вбудованих систем реального часу в мережах IoT	ECC для вбудованих систем реального часу в мережах IoT
	RSA	ECC			
80	1024	160-233	+	+	+
112	2048	224-255	+	+	+
128	3072	256-383	+	+	+
192	7680	384-541	+	+	+
256	15360	512+	+	+	+

2.5.4 Генерація підпису та час перевірки підпису

ECC скорочує час створення та перевірки підпису порівняно з RSA. Більші ключі RSA займають утричі більше часу для підпису. RSA потребує значно більше часу для підпису з довшими ключами. Але при перевірці підпису RSA є ефективнішим. Час перевірки RSA не залежить від довжини ключа, тоді як ECC вимагає часу в різних діапазонах ключів. В таблиці 2.3 представлено порівняння часу генерації підпису та перевірки підпису між RSA та ECC. Як видно, час генерації підпису в RSA та ECC мало відрізняється на рівнях безпеки 80 і 192. Однак, при використанні більших розмірів ключів (15360 і 571 для RSA та ECC відповідно) RSA займає близько трьох разів більше часу, ніж ECC. RSA з великим розміром ключа потребує значно більше часу. Втім, RSA перевершує ECC у перевірці підпису. Для RSA час перевірки не залежить від довжини ключа, тоді як ECC відстає у всіх діапазонах ключів та зростає лінійно. Тому RSA підходить для ситуацій, де важлива саме перевірка повідомлень, а не генерація підпису.

Таблиця 2.3

Створення підпису та час перевірки підпису

Дослідження	Рівень безпеки	Розмір ключа		Час генерації підпису		Час перевірки підпису	
		RSA	ECC	RSA	ECC	RSA	ECC
1. Порівняльне дослідження RSA та ECC і впровадження ECC у вбудовані системи 2. Безпека даних у хмарному сховищі за допомогою алгоритму ECC 3. Порівняння продуктивності еліптичної кривої та цифрових підписів RSA	80	1024	163	0,01	0,15	0,01	0,23
	112	2240	233	0,15	0,34	0,01	0,51
	128	3072	283	0,21	0,59	0,01	0,86
	192	7680	409	1.53	1.18	0,01	1,80
	256	15360	571	9.20	3.07	0,03	4.53

2.5.5 Час генерації та виконання ключа

Згідно даного критерію, час генерації та виконання ключа в ECC менший, ніж у RSA, і зростає лінійно з розміром ключа, а в RSA зростає експоненціально. Додаток Д вказує на те, що операція з відкритим ключем в RSA є швидшою, тоді як ECC в операції з закритим ключем набагато швидше, ніж у RSA. Для операцій із закритим ключем із рівнем безпеки 80-біт і 112-біт RSA приблизно в 14 і 40 разів повільніше, ніж ECC відповідно. Під час роботи з відкритим ключем ECC повільніше, ніж RSA. Крім того, виходячи з Додатку Д, №2, оскільки ECC-BROSMAP застосовує симетричне шифрування; отже, час виконання ECC менший, ніж RSA, крім оптимізації продуктивності. З іншого боку, час створення запиту (мобільною програмою оренди автомобіля) і підготовки результату (сервером) в ECC нижчий порівняно з RSA. Через кількість асиметричних операцій, що застосовуються в RSA на основі BROSMAP; які роблять ECC швидшим, ніж RSA, під час створення запиту. Тоді як час дешифрування RSA трохи швидший, ніж ECC. Під час дешифрування результатів RSA-BROSMAP потребує одного асиметричного процесу, одного симетричного процесу та трьох гешів, тоді як ECC-BROSMAP потребує однакових геш-чисел і двох симетричних операцій. Завдяки оптимізації коду компілятор у JAVA виконує друге та третє шифрування в ECC швидше, ніж перше симетричне шифрування. Це дає порівняльні результати з RSA на основі BROSMAP, навіть якщо текст двічі зашифровано в ECC, що має знизити продуктивність [74]. У Додатку Д можна побачити результати порівняння генерації ключів для алгоритмів ECC і RSA. В рядках № 4 і 5 видно, що ECC виявляється значно швидшим і отримує кращі показники порівняно з RSA. Зокрема, RSA виявляється близько в 15 разів повільнішим, ніж ECC. Це пояснюється тим, що для генерації ключів RSA потребується виділяти ресурси для обчислення простих чисел, що є витратним обчислювальним процесом, тоді як ECC може генерувати пару відкритих і закритих ключів швидше і без необхідності обчислення простих чисел. Крім того, RSA повільніший за ECC в 15 разів. Генерація ключів RSA потребує обчислення простих чисел, що витратно, тоді як ECC може генерувати ключі швидше без цього

обчислення. Час генерації ключів ECC збільшується лінійно з їх розміром, у відмінність від RSA, де цей час зростає експоненційно [93]. RSA обробляє 450 запитів на секунду з середнім часом відповіді 150 мс, тоді як ECC відповідає за 75 мс на таку ж кількість запитів. ECC має відмінний час відгуку під час обміну даними між сервером і робочим столом [92]. Таким чином, ECC є більш ефективним та швидким алгоритмом для генерації ключів та обробки запитів на шифрування та розшифрування порівняно з RSA.

2.5.6 Час шифрування та дешифрування

З точки зору часу шифрування та дешифрування, загальний час шифрування та дешифрування в ECC менший, ніж у RSA. На основі результатів, що RSA більш корисний у шифруванні даних, тоді як ECC більш ефективний у дешифруванні. У Додатку Е загальний час операцій шифрування та дешифрування в ECC менший, ніж у RSA. Однак, ECC потребує більше часу, ніж RSA для шифрування, але він дуже ефективний, ніж RSA, у дешифруванні, тоді як RSA дуже ефективний у шифруванні, але повільний, ніж ECC у дешифруванні. Але загалом ECC ефективніший, ніж RSA.

2.6 Висновки за розділом №2

У даному розділі було проведено ряд досліджень та отримано наступні результати:

1. Досліджено застосування асиметричної криптографії для IoT.
2. Проаналізовано фундаментальні концепції, що лежать в основі асиметричної криптографії, включаючи використання відкритих і закритих ключів, а також дослідження різних алгоритмів шляхом розгляду математичних формул та схем, таких як RSA та ECC, які використовуються в IoT.
3. Досліджені різні аспекти ECC і RSA, де алгоритми розглядаються всебічно. Аналіз проводився шляхом порівняльних таблиць згідно існуючих досліджень з точки зору різних показників.

РОЗДІЛ 3
РОЗРОБКА МЕХАНІЗМУ БЕЗПЕКИ

3.1 Advanced Encryption Standard - Galois/Counter Mode

Для покращення протоколу ECC, буде проводитись процес імплементування гібридного методу шифрування разом з Advanced Encryption Standard - Galois/Counter Mode (AES-GCM).

GCM — це режим роботи з блоковим шифруванням, який забезпечує конфіденційність і автентифікацію джерела даних. Операція шифрування з автентифікацією GCM має чотири входи: секретний ключ, вектор ініціалізації (IV), відкритий текст і вхід для додаткових автентифікованих даних (AAD). Він має два виходи: зашифрований текст, довжина якого ідентична відкритому тексту, і тег автентифікації [94, с. 2].

Архітектура AES розроблена паралельно завдяки вищій продуктивності шифрування та дешифрування AES-GCM. Паралельна архітектура розширення ключа розроблена, оскільки використовується більший розмір ключа. На рисунку 3.2 показано, що розмір ключа становить 256 біт, що призведе до виконання 14 раундів заміни та необхідних перестановок. Розмір ключа залежить від бажаного рівня безпеки. Стандартна кількість трансформаційних патронів AES-256 становить чотирнадцять патронів. Як показано в таблиці 3.1, AES шифрує 128-бітний відкритий текст або розшифровує 128-бітний зашифрований текст, багаторазово застосовуючи те саме циклічне перетворення кілька разів залежно від розміру ключа.

Коли реалізований розмір ключа більший, шифрування або дешифрування триватиме більше часу. Основною функцією шифрування AES є розклад ключів. Пропонований розмір ключа цього проекту становить 256 біт. Таким чином, запропонована конструкція AES є паралельною, що може оптимізувати час, витрачений на виконання розширення ключа та шифрування AES. Для алгоритму AES довжина вхідного блоку та вихідного блоку становить 128 біт. Розроблена

довжина ключа становить 256 біт, що відображає кількість раундів, які потрібно виконати, 14 раундів. Чим більше раундів потрібно виконати, тим більше часу буде потрібно. Таким чином, паралельні шляхи даних призначені для підвищення продуктивності AES з точки зору швидкості виконання. У Додатку Ж показана архітектура, розроблена в 256-бітному ключі.

Таблиця 3.1

Відповідність довжини ключа, розміру блоку та номеру раунду

AES	Довжина ключа (32-біт)	Розмір блоку (32-біт)	Номер раунду
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

У MixColumns кожен стовпець стану містить поліном із чотирьох членів. Завдяки зменшенню XOR-гейтів на критичних шляхах, архітектура MixColum має бути більш ефективною. Стандартні поліноміальні рівняння першого стовпця показані у формулі (3.1):

$$s'_{0,c} = \{[02] \cdot s_{0,c}\} \oplus \{[03] \cdot s_{1,c}\} \oplus s_{2,c} \oplus s_{3,c}. \quad (3.1)$$

Після розкладання воно виглядає як формула (3.2),

$$s'_{0,c} = [02]s_{0,c} \oplus [02]s_{1,c} \oplus s_{1,c} \oplus s_{2,c} \oplus s_{3,c} \quad (3.2)$$

Функція множення «*[02]*» у формулі (3.2) в шістнадцятковій формі представлена як XTime, як показано на рисунку 3.1.

Після розширення рівняння критичний шлях перетворення MixColumn дорівнює 4 з XOR-гейтом. Ефективна архітектура реалізації MixColumn може бути отримана шляхом мінімізації кількості XOR-гейтів.

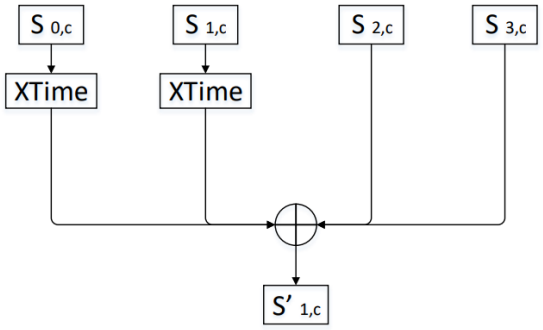

Рисунок 3.1 – MixColumn перетворення.

Рівняння можна спростити та переписати як формула (3.3):

$$s'_{0,c} = [02]\{s_{0,c} \oplus s_{1,c}\} \oplus \{s_{1,c} \oplus s_{2,c}\} \oplus s_{3,c} \quad (3.3)$$

Як показано на рисунку 3.2, критичний шлях MixColumn зменшено, три критичні шляхи об'єднані в XOR для генерації виходу MixColumn. Тому потрібен один XTime з трьома XOR-гейтами.

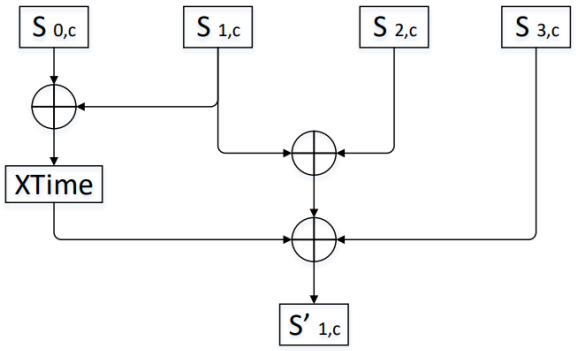

Рисунок 3.2 – Ефективність перетворення MixColumn

AES-GCM має два основні компоненти, механізм AES і функцію GHASH, як показано на рисунку 3.3. У геш-підключі для функції GHASH 128-бітний «нульовий» блок як вхід зашифровано, а геш-підключ зберігатиметься в реєстрі. 96-бітний IV додається з $0^{31}\|1$ і генерується Nonce у лічильнику 1, у той же час 32-бітова функція збільшення застосовується для формування наступного блоку лічильника. Nonce діє як вхід шифрування AES і генерує проміжне геш-значення *Yi*. Коли він отримує

додаткові дані автентифікації, AAD і до них додається зашифрований текст, функція GHASH генерує тег автентифікації. Та сама функція автентифікованого шифрування та автентифікованого дешифрування. Вихід автентифікованого дешифрування — тег автентифікації, згенерований, повинен відповідати тегу шифрування. Якщо не збігається, функція дешифрування не працює [95. с. 2-5].

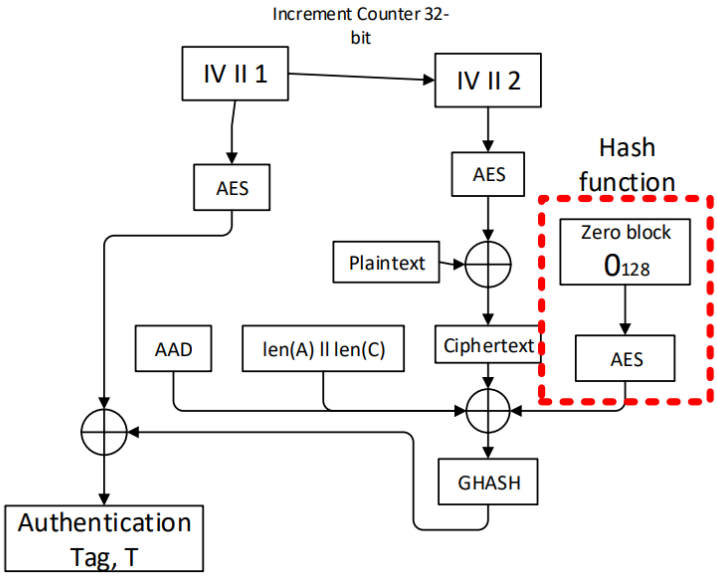

Рисунок 3.3 – AES-GCM шифрування та дешифрування.

GCM доведено захищений від зловмисників, які можуть адаптивно вибирати відкриті тексти, зашифровані тексти, ICV і поле AAD за стандартними криптографічними припущеннями (приблизно, вихід основного шифру під випадково вибраним ключем не відрізняється від випадково вибраного виводу). По суті, це означає, що, якщо використовується в межах призначених параметрів, розрив GCM передбачає розрив основного блочного шифру [96]. Найважливішим фактором безпеки є те, що IV ніколи не повторюється для даного ключа. Частково це вирішується шляхом заборони використання AES-GCM під час використання статично налаштованих ключів.

Коли IKE [97] використовується для встановлення нових ключів між двома одноранговими об’єктами, для двох потоків трафіку встановлюються окремі ключі. Якщо інший механізм використовується для встановлення нових ключів (той, який

встановлює лише один ключ для шифрування пакетів), тоді існує висока ймовірність того, що однорангові вузли виберуть однакові значення IV для деяких пакетів.

Реалізації, які дозволяють використовувати той самий ключ для шифрування та дешифрування пакетів з одним і тим же вузлом, повинні гарантувати, що два однорангові вузли призначають різні значення солі асоціації безпеки. Інше зауваження полягає в тому, що, як і в будь-якому режимі шифрування, безпека всіх даних, захищених відповідно до певної асоціації безпеки, дещо знижується з кожним повідомленням. Щоб захиститися від цієї проблеми, реалізації повинні згенерувати новий ключ перед шифруванням 2^{64} блоків даних заданим ключем. Однак, що неможливо досягти цієї межі при використанні 32-розрядних порядкових номерів [96, с. 2].

3.2 Розроблення механізму безпеки на основі встановлених вимог

Початок розроблення механізму проводиться з вибору бібліотек для програмної реалізації покращення механізму безпеки протоколу розподілу ключів ECC.

`from tinyec import registry`: цей рядк імпортує модуль `registry` з бібліотеки `tinyec`, яка забезпечує операції з еліптичною кривою.`import hashlib` імпортує модуль `hashlib`, який забезпечує криптографічні геш-функції для гешування даних.`import secrets`: цей рядок імпортує модуль `secrets`, який використовується для генерації випадкових чисел і безпечних маркерів.

`import binascii`: цей рядок імпортує модуль `binascii`, який надає функції для перетворення двійкових даних у ASCII (ASCII — це 8-бітний код. Тобто він використовує вісім бітів для представлення літери чи пунктуації. Двійковий код із восьми цифр, наприклад *1101 10112*, може зберігатися в одному байті пам’яті комп’ютера.) представлення і навпаки.

`from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes`: ці рядки імпортують необхідні класи з модуля `cryptography.hazmat.primitives.ciphers`, включаючи `Cipher`, `algorithms` і `modes` . Ці класи використовуються для шифрування AES.

`from cryptography.hazmat.backends import default_backend`: цей рядок імпортує серверну частину за замовчуванням для бібліотеки `cryptography` (рисунок 3.4).

```
from tinyec import registry
import hashlib
import secrets
import binascii

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
```

Рисунок 3.4 – Бібліотеки та імпорти

def generate_encryption_keys(pub_key) є функцією під назвою generate_encryption_keys, яка приймає відкритий ключ (pub_key) як вхідні дані. Ця функція виконує шифрування за допомогою криптографії на основі еліптичної кривої (ECC) і AES-GCM. На рисунку 3.5 показана функція шифрування.

```
def generate_encryption_keys(pub_key):
    private_key = secrets.randbelow(curve.field.n)
    public_key = private_key * curve.g

    shared_key = pub_key * private_key
    secret_key = derive_256_bit_key(shared_key)

    ciphertext, nonce, auth_tag = encrypt_AES_GCM(message, secret_key)

    return (ciphertext, nonce, auth_tag, public_key)
```

Рисунок 3.5 – Функція шифрування

def generate_decryption_key(private_key, ciphertext, nonce, auth_tag, public_key): цей рядок визначає функцію під назвою generate_decryption_key, яка приймає приватний ключ (private_key), зашифрований текст, nonce, тег автентифікації та відкритий ключ зашифрованого тексту як вхідні дані. Ця функція виконує дешифрування за допомогою ECC і AES-GCM. На рисунку 3.6 показана функція дешифрування.

```
def generate_decryption_key(private_key, ciphertext, nonce, auth_tag, public_key):
    shared_key = public_key * private_key
    secret_key = derive_256_bit_key(shared_key)

    plaintext = decrypt_AES_GCM(ciphertext, nonce, auth_tag, secret_key)
    return plaintext
```

Рисунок 3.6 – Дешифрування

def encrypt_AES_GCM(message, secret_key): цей рядок визначає функцію під назвою encrypt_AES_GCM, яка приймає повідомлення (message) і секретний ключ як вхідні дані. Ця функція, яка наведена на рисунку 3.7, шифрує повідомлення за допомогою AES-GCM.

```
def encrypt_AES_GCM(message, secret_key):
    aes_key = secret_key[:16]
    nonce = secrets.token_bytes(12)
    cipher = Cipher(algorithms.AES(aes_key), modes.GCM(nonce), backend=default_backend())

    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(message) + encryptor.finalize()
    auth_tag = encryptor.tag

    return ciphertext, nonce, auth_tag
```

Рисунок 3.7 – AES-GCM шифрування

def decrypt_AES_GCM(ciphertext, nonce, auth_tag, secret_key): цей рядок визначає функцію під назвою decrypt_AES_GCM, яка приймає зашифрований текст, nonce, тег автентифікації та секретний ключ як вхідні дані. Ця функція, яка наведена на рисунку 3.8 розшифровує зашифрований текст за допомогою AES-GCM.

```
41  def decrypt_AES_GCM(ciphertext, nonce, auth_tag, secret_key):
42      aes_key = secret_key[:16]
43      cipher = Cipher(algorithms.AES(aes_key), modes.GCM(nonce, auth_tag), backend=default_backend())
44
45      decryptor = cipher.decryptor()
46      plaintext = decryptor.update(ciphertext) + decryptor.finalize()
47      return plaintext
```

Рисунок 3.8 – AES дешифрування

def derive_256_bit_key (point): цей рядок, який наведений на рисунку 3.9, визначає функцію під назвою derive_256_bit_key, яка приймає точку ECC (point) як вхідні дані. Ця функція гешує координати x і y точки ECC за допомогою SHA-256 і повертає 256-бітний ключ, отриманий із гешу.

SHA-256 (Secure Hash Algorithm 256-bit) — широко поширена криптографічна геш-функція, яка належить до сімейства SHA-2. Він приймає вхідне повідомлення довільної довжини та створює 256-бітне геш-значення фіксованого розміру. Сила геш-функції полягає в її стійкості до різноманітних криптографічних атак, включаючи атаки прообразу, атаки другого прообразу та атаки зіткнення.

Геш-функція вважається стійкою до прообразу, якщо обчислювально неможливо знайти будь-яке вхідне повідомлення, яке гешує до певного вихідного геш-значення. У контексті похідного ключа стійкість прообразу гарантує, що отримати вихідний спільний секретний ключ із похідного 256-бітного ключа практично неможливо. Геш-функція SHA-256 має високий рівень стійкості до прообразів, що робить її придатною для виведення ключів.

Також вона є стійкою до другого прообразу означає, що з огляду на вхідне повідомлення обчислювально неможливо знайти інше вхідне повідомлення, яке дає таке саме геш-значення. У коді функція derive_256_bit_key() отримує ключ із загального секретного ключа за допомогою гешу SHA-256. Стійкість до другого прообразу SHA-256 гарантує, що навіть якщо зловмисник знає похідний ключ, обчислювально неможливо знайти інший спільний секретний ключ, який створив би той самий похідний ключ. Ця властивість підвищує безпеку процесу отримання ключа.

SHA-256 є стійкою до зіткнень, або колізій. Це означає, що обчислювально неможливо знайти два різних вхідних повідомлення, які створюють однакове геш-значення. Хоча стійкість до зіткнень не має прямого відношення до процесу отримання ключа в наданому коді, вона є важливою властивістю для загальної безпеки геш-функцій. Це запобігає зловмиснику від пошуку двох різних спільних

секретних ключів, які створюють той самий похідний ключ, що потенційно може призвести до вразливості безпеки.

Використання геш-функції SHA-256 у функції derive_256_bit_key() гарантує, що похідний ключ має такі бажані властивості, як стійкість до прообразу, стійкість до другого прообразу та стійкість до зіткнень. Ці властивості необхідні для підтримки безпеки та цілісності отриманого ключа, що, у свою чергу, посилює загальну безпеку розподілу ключів і схеми шифрування.

Важливо відзначити, що криптографічні геш-функції, такі як SHA-256, широко вивчаються та пройшли ретельний аналіз криптографічної спільноти. Властивості міцності та безпеки SHA-256 добре задокументовані та широко прийняті на практиці [62, с. 82-86].

```
def derive_256_bit_key(point):
    sha = hashlib.sha256(int.to_bytes(point.x, 32, 'big') + int.to_bytes(point.y, 32, 'big'))
    return sha.digest()
```

Рисунок 3.9 – Гешування

curve = registry.get_curve('secp256r1'): цей рядок отримує еліптичну криву 'secp256r1' з registry' і призначає її змінній curve`, як показано на рисунку 3.10. Вибираючи еліптичну криву для криптографічних операцій, дуже важливо враховувати різні фактори, такі як безпека, продуктивність і сумісність. Далі буде проаналізовано дві популярні криві, "brainpoolP256r1" і "secp256r1" за наступними критеріями:

Аналіз безпеки: Крива "brainpoolP256r1" пропонує 256-бітний розмір ключа та пройшла ретельне дослідження, демонструючи стійкість проти відомих атак [98]. Властивості безпеки кривої роблять її придатним вибором для криптографічних операцій у середовищах IoT [99]. 'secp256r1': Крива 'secp256r1', визначена Групою стандартів ефективної криптографії (SECG), широко поширена та вважається безпечною проти відомих атак. Він пройшов ретельний аналіз і використовується в численних криптографічних програмах, включаючи TLS і Bitcoin [100]. Безпека

кривої добре зарекомендувала себе в криптографічному співтоваристві. Обидві криві демонструють високий рівень безпеки та були ретельно перевірені дослідницьким співтовариством. Однак важливо зазначити, що поточні дослідження та прогрес у криптоаналізі можуть призвести до виявлення вразливостей у майбутньому. Регулярний моніторинг криптографічного ландшафту має вирішальне значення для підтримки безпеки систем IoT.

Аналіз продуктивності: криві 'brainpool', включаючи 'brainpoolP256r1', пропонують ефективні арифметичні операції, що забезпечує сприятливі характеристики продуктивності. Ці криві спеціально розроблені для забезпечення покращеної продуктивності при збереженні безпеки [98]. Оптимізації в «brainpoolP256r1» можуть дати переваги в певних реалізаціях або платформах IoT з обмеженими ресурсами.'secp256r1': 'secp256r1' широко підтримується та оптимізовано для продуктивності. Завдяки обширним дослідженням і зусиллям з оптимізації, результатом цього є ефективні арифметичні операції. Крім того, поширена апаратна підтримка 'secp256r1', що сприяє його популярності та широкому використанню. Різниця в продуктивності між «brainpoolP256r1» і «secp256r1» залежить від конкретної реалізації, платформи та доступних оптимізацій. Оцінюючи вимоги до продуктивності, рекомендується враховувати цільове середовище IoT і доступні ресурси.

Аналіз сумісності: хоча криві "brainpool" можуть мати дещо нижчу сумісність порівняно з широко стандартизованими кривими, такими як "secp256r1", вони все ще добре підтримуються в основних криптографічних бібліотеках і платформах. Занепокоєння щодо сумісності пом’якшуються тим фактом, що «brainpoolP256r1» визнано та прийнято криптографічним співтовариством. 'secp256r1' має широку сумісність між криптографічними бібліотеками, протоколами та обладнанням. Він широко підтримується та широко використовується, забезпечуючи взаємодію між різними платформами та системами. Його використання в численних криптографічних програмах робить його безпечним вибором для розгортання IoT [100, с. 916].

Беручи до уваги комплексний аналіз безпеки, широке впровадження, оптимізацію продуктивності та сумісність з існуючими системами, «secp256r1» зазвичай рекомендується як краща крива для більшості додатків IoT. Більш того, Криві Брейнпула використовують випадкові прості числа, на відміну від квазіпростих чисел Мерсенна, які використовують криві NIST. Як результат, швидке скорочення неможливе для кривих "brainpool", і це має серйозні наслідки для продуктивності різних кривих.

`message = b'Текст, який потрібно зашифрувати відкритим ключем ECC і розшифрувати його відповідним закритим ключем ECC'`: цей рядок визначає повідомлення, яке буде зашифровано як рядок байтів.

`private_key = secrets.randbelow(curve.field.n)`: цей рядок генерує випадковий закритий ключ за допомогою модуля `secrets`.Приватний ключ є випадковим числом, меншим за порядок кривої (`curve.field.n`).

`public_key = private_key * curve.g`: цей рядок обчислює відповідний відкритий ключ шляхом множення закритого ключа (`private_key`) на точку генератора (`curve.g`), як показано на рисунку 3.10.

```
if __name__ == "__main__":

    curve = registry.get_curve('secp256r1')

    message = b'Text to be encrypted by ECC public key and decrypted by its corresponding ECC private key'

    print("Original message:", message)

    private_key = secrets.randbelow(curve.field.n)
    public_key = private_key * curve.g
```

Рисунок 3.10 – Вивід оригінального повідомлення, генерування ключа та обчислення відкритого ключа

`encrypted_message = generate_encryption_keys(public_key)`: цей рядок, який наведений на рисунку 3.11, викликає функцію ` generate_encryption_keys` для шифрування повідомлення за допомогою відкритого ключа (`public_key`) і AES-GCM.

`encrypted_message_obj = { ... }`: цей блок коду, який наведений на рисунку 3.11, створює словник `encrypted_message_obj`, що містить зашифроване повідомлення (`ciphertext`), одноразовий номер (`nonce`), тег автентифікації (`authTag`) і стиснуту форму відкритого ключа (`public_key`).

```
encrypted_message = generate_encryption_keys(public_key)
encrypted_message_obj = {
    'ciphertext': binascii.hexlify(encrypted_message[0]),
    'nonce': binascii.hexlify(encrypted_message[1]),
    'auth_tag': binascii.hexlify(encrypted_message[2]),
    'public_key': hex(encrypted_message[3].x) + hex(encrypted_message[3].y % 2)[2:]
}
```

Рисунок 3.11 – Виклик функції шифрування та вміст зашифрованого повідомлення

`print("Encrypted message:", encrypted_message_obj)`: цей рядок друкує зашифроване повідомлення.

`decrypted_message= generate_decryption_key(private_key, encrypted_message[0], encrypted_message[1], encrypted_message[2], encrypted_message[3])`: цей рядок викликає функцію `ecc_calc_decryption_key` для розшифровки повідомлення за допомогою закритого ключа (`private_key`) і отриманий зашифрований текст, nonce, тег автентифікації та відкритий ключ.

`print("Decrypted message:", decrypted_message)`: цей рядок друкує розшифроване повідомлення.

Дані рядки, які проводять друк зашифрованого, друк розшифрованого повідомлення після виклику функції розшифрування, наведені на рисунку 3.12.

```
print("Encrypted message:", encrypted_message_obj)

decrypted_message = generate_decryption_key(private_key, encrypted_message[0], encrypted_message[1],
                                            encrypted_message[2], encrypted_message[3])
print("Decrypted message:", decrypted_message)
```

Рисунок 3.12 – Друк зашифрованого, друк розшифрованого повідомлення після виклику функції розшифрування

3.2 Опис розробленого механізму безпеки

У розробленому механізмі принцип шифрування виконується наступним чином:

1. Функція generate_encryption_keys, яка наведена на рисунку 3.6, приймає відкритий ключ одержувача (pub_key) як вхідні дані.

2. Він генерує випадковий приватний ключ (private_key) за допомогою secrets.randbelow(curve.field.n), де curve.field.n є порядком еліптичної кривої.

3. Функція обчислює відповідний відкритий ключ (public_key) для згенерованого закритого ключа шляхом множення закритого ключа на точку генератора еліптичної кривої (curve.g).

4. Спільний ключ ECC обчислюється шляхом множення відкритого ключа одержувача (pub_key) на згенерований закритий ключ (private_key).

5. Щоб отримати 256-бітний секретний ключ, викликається функція derive_256_bit_key, яка показана на рисунку 3.10, зі спільним ключем ECC як вхідні дані. Ця функція гешує координати x і y точки ECC за допомогою SHA-256 і повертає 256-бітний ключ, отриманий із гешу.

6. Повідомлення (повідомлення) зашифровано за допомогою AES-GCM із отриманим секретним ключем.

7. Викликається функція encrypt_AES_GCM, згідно рисунку 3.8, яка виконує процес шифрування.

8. Функція повертає зашифрований текст, nonce, тег автентифікації та відкритий ключ, який використовується для шифрування, як показано на рисунку 3.6.

В свою чергу, процес дешифрування:

1. Функція generate_decryption_key, наведена на рисунку 3.7, приймає приватний ключ одержувача (private_key), зашифрований текст, nonce, тег автентифікації та відкритий ключ, який використовується для шифрування (public_key), як вхідні дані.

2. Спільний ключ ECC обчислюється шляхом множення відкритого ключа, який використовується для шифрування (public_key), на закритий ключ одержувача

(private_key).Ця функція отримує 256-бітний секретний ключ за допомогою функції derive_256_bit_key, яка показана на рисунку 3.10, подібної до процесу шифрування.

3. Зашифрований текст розшифровується за допомогою AES-GCM із отриманим секретним ключем. Викликається функція decrypt_AES_GCM, яка наведена на рисунку 3.9, яка виконує процес дешифрування.

4. Функція повертає розшифрований відкритий текст.

Додаткове роз`яснення певних дій:

1. Функція encrypt_AES_GCM, наведена на рисунку 3.8, приймає повідомлення (message) і секретний ключ як вхідні дані.

2. Він витягує 128-бітний ключ AES із 256-бітного секретного ключа.

3. Генерується випадковий одноразовий номер довжиною 12 байт.

4. Об’єкт шифрування AES-GCM створюється за допомогою ключа AES і nonce.

5. Повідомлення зашифровано за допомогою шифратора AES-GCM.

6. Повертається зашифрований текст, nonce і тег автентифікації.

7. Функція decrypt_AES_GCM, показана на рисунку 3.9, приймає зашифрований текст, nonce, тег автентифікації та секретний ключ як вхідні дані.

8. Він витягує 128-бітний ключ AES із 256-бітного секретного ключа.

9. Об’єкт шифрування AES-GCM створюється за допомогою ключа AES, nonce та тегу автентифікації.

10. Зашифрований текст розшифровується за допомогою дешифратора AES-GCM.

11. Повертається розшифрований відкритий текст.

Стиснення та шістнадцяткове перетворення:

1. Відкритий ключ, який використовується для шифрування, стискається шляхом конкатенації шістнадцяткового представлення його координати x і молодшого біта його координати y hex(encrypted_message[3].x) + hex(encrypted_message[3].y % 2)[2:], як показано на рисунку 3.12.

2. Зашифрований текст, nonce та тег автентифікації перетворюються на шістнадцяткове за допомогою binascii.hexlify.

3. Нарешті друкуються оригінальне повідомлення, зашифроване повідомлення та розшифроване повідомлення, як наведено на рисунку 3.13.

Цей код використовує ECC для встановлення спільного секретного ключа між відправником і одержувачем, який потім використовується для шифрування та дешифрування повідомлення за допомогою AES-GCM (Додаток А). Результат наведено на рисунку 3.13.

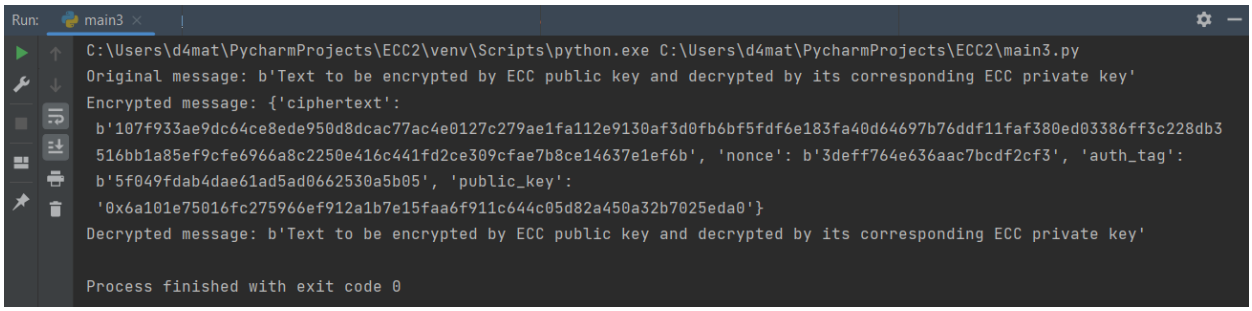

Рисунок 3.13 – Результат розробленого механізму

3.3 Вплив механізму безпеки на швидкість та ресурси пристроїв IoT

Впровадження AES-GCM в ECC може мати кілька наслідків для швидкості шифрування/дешифрування, споживання пам’яті та використання ресурсів пристроями IoT. Шифрування/дешифрування на основі ECC зазвичай швидше порівняно з RSA за того самого рівня безпеки. ECC працює з меншими розмірами ключів, що призводить до швидших криптографічних операцій. AES-GCM — це швидкий симетричний алгоритм шифрування, який підходить для пристроїв з обмеженими ресурсами. Він забезпечує ефективне шифрування та автентифікацію з паралелізованими операціями, що забезпечує швидшу обробку. Поєднання ECC і AES-GCM може призвести до високої швидкості шифрування/дешифрування завдяки ефективності обох алгоритмів. Також ECC вимагає менших розмірів ключів порівняно з RSA для еквівалентних рівнів безпеки. Цей атрибут зменшує обсяг пам’яті реалізацій на основі ECC, що робить його придатним для пристроїв з обмеженими ресурсами пам’яті. AES-GCM також ефективно працює з меншими вимогами до пам’яті, що додатково сприяє зменшенню споживання пам’яті в

загальній реалізації. Реалізації на основі ECC зазвичай вимагають менше обчислювальних ресурсів, таких як обчислювальна потужність і енергія, порівняно з RSA. AES-GCM розроблено для ефективного використання ресурсів. Він пропонує паралельні операції шифрування/дешифрування, що дозволяє краще використовувати ресурси в пристроях IoT, а також використовує можливості апаратного прискорення, наприклад AES-NI (Advanced Encryption Standard New Instructions) — це розширення архітектури набору інструкцій x86, яке забезпечує апаратне прискорення для операцій шифрування та дешифрування AES. Він представляє спеціальні інструкції, спеціально розроблені для підвищення продуктивності алгоритмів AES. AES-NI має на меті підвищити продуктивність і ефективність криптографічних операцій AES шляхом перенесення їх на спеціальне обладнання в процесорі. Він прискорює ключові операції AES, включно з шифруванням, дешифруванням, генерацією та розширенням ключів, реалізуючи їх безпосередньо в апаратному забезпеченні. Основною метою AES-NI є прискорення операцій AES, зменшення обчислювального навантаження на ЦП і підвищення загальної продуктивності системи. AES-NI забезпечує суттєві покращення продуктивності для шифрування та дешифрування AES порівняно з програмними реалізаціями. Він досягає більшої швидкості шифрування/дешифрування, виконуючи операції AES безпосередньо в апаратному забезпеченні, що ефективніше, ніж програмні реалізації. Інструкції AES-NI можуть обробляти кілька блоків даних паралельно, додатково підвищуючи пропускну здатність і зменшуючи затримку. AES-NI може значно підвищити продуктивність AES-GCM, оскільки використовує AES для шифрування та автентифікації. За допомогою AES-NI можна прискорити операції шифрування/дешифрування AES, необхідні для AES-GCM, що призведе до швидшої обробки GCM. AES-NI може значно підвищити продуктивність AES-GCM, зробивши його більш придатним для пристроїв з обмеженими ресурсами та програм з високою пропускною здатністю [101].

3.4 Переваги реалізованого механізму безпеки

Виконана програмна реалізація пропонує кілька переваг у контексті IoT:

1. Ефективність: ECC відомий своєю ефективністю з точки зору розміру ключа та складності обчислень. ECC пропонує еквівалентну безпеку з меншими розмірами ключів, що призводить до швидшого обчислення та менших вимог до пам’яті. Це вкрай важливо для пристроїв IoT з обмеженими ресурсами, які часто мають обмежену обчислювальну потужність, пам’ять та енергію.

2. Коротша довжина ключа: ECC забезпечує той самий рівень безпеки, що й RSA, зі значно меншою довжиною ключа. Це вигідно для пристроїв IoT, оскільки зменшує накладні витрати на керування ключами та операції шифрування/дешифрування. Більш коротка довжина ключа також мінімізує споживання пропускної здатності під час зв’язку, що важливо для пристроїв IoT з обмеженими можливостями мережі.

3. Швидка генерація ключів: код використовує бібліотеку `secrets` для генерації випадкових чисел, що є безпечним і ефективним підходом. Здатність швидко генерувати ключі є важливою в сценаріях IoT, де пристроям може знадобитися встановити безпечне з’єднання та швидко виконати операції шифрування/дешифрування.

4. Безпечне шифрування: Код поєднує ECC для обміну ключами та AES-GCM для шифрування повідомлень. AES-GCM — це широко поширений алгоритм симетричного шифрування, який забезпечує конфіденційність, цілісність і автентифікацію даних. Використання AES-GCM гарантує безпеку зашифрованих повідомлень під час передачі та захист від несанкціонованих змін.

5. Геш-функція для отримання ключа: код використовує геш-функцію (SHA-256) для отримання 256-бітного секретного ключа зі спільного ключа ECC. Цей підхід гарантує, що отриманий ключ підходить для використання з шифруванням AES-GCM. Завдяки використанню сильної геш-функції підвищується безпека отриманого ключа.

6. Компактний і стиснутий відкритий ключ: Код стискає представлення відкритого ключа за допомогою шістнадцяткового кодування. Це зменшує розмір переданого відкритого ключа, що робить його більш придатним для середовищ IoT з обмеженими ресурсами. Компактне представлення має вирішальне значення в сценаріях, де обмеження пропускної здатності та пам’яті є серйозними проблемами.

7. Наскрізне шифрування: Код демонструє повний процес наскрізного шифрування, включаючи шифрування відправником і дешифрування одержувачем. Цей рівень шифрування гарантує, що конфіденційні дані, якими обмінюються пристрої IoT, залишаються захищеними та конфіденційними протягом усього процесу зв’язку.

3.5 Висновки за розділом №3

У даному розділі було проведено ряд досліджень та отримано наступні результати:

1. Проаналізовано метод покращення захисту обраного протоколу розподілу ключів після порівняння за різними показниками та обґрунтовано доцільність імплементування AES-GCM в ECC.

2. Реалізовано та описано механізм шифрування та дешифрування за допомогою ECC у поєднанні з AES-GCM для захисту зв’язку IoT.

3. Оцінено вплив розробленого механізму безпеки на ресурси пристроїв IoT.

ВИСНОВКИ

У дипломній роботі розв’язано актуальне завдання забезпечення захисту інформації за рахунок використання протоколу розподілу ключів для IoT. В ході розв’язання поставлених задач були отримані наступні наукові та практичні результати:

1. Проведено аналіз концепцій типів архітектур, хмарні обчислення, хмару як платформу, проблем інтеграції в хмарні обчислення та викликів для IoT.
2. Проаналізовано ризики для IoT, а саме інцидентів та статистичних даних, тим самим обґрунтовано необхідності впровадження захищених ПРК для підтримки цілісності та конфіденційності даних, що передаються в мережах IoT.
3. Проведено аналіз стандартів та правил, які регулюють вимоги безпеки для ПРК у контексті IoT.
4. Досліджено застосування асиметричної криптографії для IoT.
5. Проаналізовано фундаментальні концепції, що лежать в основі асиметричної криптографії, включаючи використання відкритих і закритих ключів, а також дослідження різних алгоритмів шляхом розгляду математичних формул та схем, таких як RSA та ECC, які використовуються в IoT.
6. Досліджені різні аспекти ECC і RSA, де алгоритми розглядаються всебічно. Аналіз проводився шляхом порівняльних таблиць згідно існуючих досліджень з точки зору деяких показників.
7. Проаналізовано метод покращення захисту обраного протоколу розподілу ключів після порівняння за різними показниками та обґрунтовано доцільність імплементування AES-GCM в ECC.
8. Реалізовано та описано механізм шифрування та дешифрування за допомогою ECC у поєднанні з AES-GCM для захисту зв’язку IoT.
9. Оцінено вплив розробленого механізму безпеки на ресурси пристроїв IoT.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hojlo J. Future of Industry Ecosystems: Shared Data and Insights [Електронний ресурс] / J. Hojlo. – 2021. – Режим доступу: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/

2. Nokia OYJ. Threat Intelligence Report 2021 / Nokia OYJ // [Electronic resource] – 2021. – Access mode: https://vpnoverview.com/wp-content/uploads/nokia_threat_intelligence_report_2021_report_en.pdf.

3. Abubakar M., Ali H., Ghaleb B., Wadhaj I., Buchanan W. An Overview of Blockchain-Based IoT Architectures and Designs / M. Abubakar, H. Ali, B. Ghaleb, I. Wadhaj, W. Buchanan // [Electronic resource]. – 2023. – Access mode: https://napier-repository.worktribe.com/output/3020671/an-overview-of-blockchain-based-iot-architectures-and-designs.

4. Eustis A. The Mirai Botnet and the Importance of IoT Device Security / A. Eustis // [Electronic resource]. – 2019. – Access mode: https://www.researchgate.net/publication/333313393_The_Mirai_Botnet_and_the_Importance_of_IoT_Device_Security.

5. Singh D., Tripathi G., Jara A.J., editors. A survey of Internet-of-Things: Future vision, architecture, challenges and services / D. Singh, G. Tripathi, A.J. Jara // 2014 IEEE world forum on Internet of Things (WF-IoT). – 2014.

6. Malche T., Maheshwary P., editors. Internet of Things (IoT) for building smart home system / T. Malche, P. Maheshwary // 2017 International Conference on ISMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). – 2017.

7. Zhao Y., Wang W., Li Y., Meixner C.C., Tornatore M., Zhang J. Edge computing and networking: A survey on infrastructures and applications / Y. Zhao, W.Wang, Y.Li, C.C.Meixner, M.Tornatore, J.Zhang // IEEE Access. – 2019.

8. Chakraborty S., Aithal S. Let Us Create an Alexa Skill for Our IoT Device Inside the AWS Cloud / S. Chakraborty, S. Aithal // International Journal of Case Studies in Business, IT, and Education. – 2023. – P. 214-225.

9. Hameed Rashid M., Abed W. IoT sensor network data processing using the TWLGA Scheduling Algorithm and the Hadoop Cloud Platform / M. Hameed Rashid, W. Abed // Wasit Journal of Computer and Mathematics Science. – 2023. – P. 135-145.

10. Palma F., Olsson T., Wingkvist A., Gonzalez-Huerta J. Assessing the linguistic quality of REST APIs for IoT applications / F. Palma, T. Olsson, A. Wingkvist, J. Gonzalez-Huerta // Journal of Systems and Software. – 2022.

11. Singh V., Sharan H. Security Analysis and Improvements to IoT Communication Protocols -CoAP / V. Singh, H. Sharan // International Research Journal of Engineering and Technology (IRJET). – 2019.

12. Himmat M., Algazoli G., Hammam N., Abdalla A. Review on the Current State of the Internet of Things and its Extension and its Challenges / M. Himmat, G. Algazoli, N. Hammam, A. Abdalla // European Journal of Information Technologies and Computer Science. – 2022.

13. Cihan P. IoT Technology in Smart Agriculture / P. Cihan // International Conference on Recent Academic Studies.– 2023. – P. 185-192.

14. Tekinerdogan B., Koksal O., Çelik T. System Architecture Design of IoT-Based Smart Cities / B. Tekinerdogan, O. Koksal, T. Celik // Applied Sciences. – 2023.

15. Muhammed A., Ucuz D. Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives / A. Muhammed, D. Ucuz // 2020 8th International Symposium on Digital Forensics and Security (ISDFS). – 2020. – P. 1-4.

16. Laghari A., Wu K., Laghari RA., Ali M., Khan AA. A review and state of art of Internet of Things (IoT) / A. Laghari, K. Wu, R. A. Laghari, M. Ali, A. A. Khan // Archives of Computational Methods in Engineering. – 2021. – P. 1-19.

17. Bodduna R. A Review on the Different Types of Internet of Things (IoT) / R. Bodduna // Jour of Adv Research in Dynamical & Control Systems – 2019.

18. Islam SR., Kwak D., Kabir MH., Hossain M., Kwak K-S. The internet of things for health care: a comprehensive survey / SR. Islam, D. Kwak, MH. Kabir, M. Hossain, K-S. Kwak // IEEE Access. –2015. – P. 678-708.

19. Silva BN., Khan M., Han K. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities / BN. Silva, M. Khan, K. Han // Sustainable Cities and Society. –2018. – P. 697-713.

20. Yasmin R., Petajajarvi J., Mikhaylov K., Pouttu A. On the integration of LoRaWAN with the 5G test network / R. Yasmin, J. Petajajarvi, K. Mikhaylov, A. Pouttu // – 2017. – P.1-6.

21. Satka Z., Ashjaei M., Fotouhi H., Daneshtalab M., Sjödin M., Mubeen S. QoS-MAN: A Novel QoS Mapping Algorithm for TSN-5G Flows / Z. Satka, M. Ashjaei, H. Fotouhi, M. Daneshtalab, M. Sjödin, S. Mubeen // [Electronic resource]. – 2022. Access mode: https://ieeexplore.ieee.org/document/9904785.

22. Sultan Hajam S.A.S. Shahid. IoT-Fog architectures in smart city applications: A survey / S.A.S.Shahid Sultan Hajam // China Communications. –2021. – P. 117-140.

23. Lee J., Nazarenko A.A., Luis-Ferreira F., Gonçalves D., Sarraipa J. A Novel Hardware Security Architecture for IoT Device: PDCRP (PUF Database and Challenge–Response Pair) Bloom Filter on Memristor-Based PUF / J. Lee, A. A. Nazarenko, F. Luis-Ferreira, D. Gonçalves, J. Sarraipa // Applied Science. – 2020.

24. Ibrahim N, Cross-layer design in the Internet of Things (IoT) issues and possible solutions / N.Ibrahim. // ResearchGate. – 2023.

25. Bana A-S., de Carvalho B.S., Tavares A, Massive MIMO for Internet of Things (IoT) connectivity / A-S. Bana, B. S. de Carvalho, A. Tavares // Physical Comunication. – 2019.

26. Giao J., Nazarenko A.A., Luis-Ferreira F., Gonçalves D., Sarraipa J. A Framework for Service Oriented Architecture (SOA)-Based IoT Application Development / J. Giao, A. A. Nazarenko, F. Luis-Ferreira, D. Gonçalves, J. Sarraipa // Processes. –2022.

27. Ngu A.H.H., Gutierrez M., Metsis V., Nepal S., Sheng Q.Z. IoT Middleware: A Survey on Issues and / A. H. H . Ngu, M. Gutierrez, V. Metsis, S. Nepal, Q. Z. Sheng // IEEE INTERNET OF THINGS JOURNAL.–2016.

28. Sommer F.S.M.F.a.J.S.P. Message-oriented Middleware for Industrial Production Systems / F.S.M.F.a.J.S.P. Sommer // in IEEE 14th International Conference on Automation Science and Engineering (CASE). – Munich, 2018.

29. Belokosztolszki A., Moody D., Pease R.P. Role-based access control for publish/subscribe middleware architectures / A. Belokosztolszki, D. Moody, R. P. Pease // in Proceedings of the 2nd international workshop on Distributed event-based systems. – 2003.

30. Asato T., Sasaki Y., Okada T. A Reusability-based Hierarchical Fault-detection Architecture for Robot Middleware and its Implementation in an Autonomous Mobile Robot System / T. Asato, Y. Sasaki, T. Okada // in IEEE/SICE International Symposium on. – Sapporo, 2016.

31. Awan K.A., Ahmad I., Ullah D., Ali H.A., Khan A.K., Javed J.J., Raza P.C. EdgeTrust: A Lightweight Data-Centric Trust Management Approach for IoT-Based Healthcare 4.0 / K. A. Awan, I. Ahmad, D. Ullah, H. A. Ali, A. K. Khan, J. J. Javed, P. C. Raza // Electronics (Basel). – 2022.

32. Krčo S., Pokrić B., Carrez F., editors. Designing IoT architecture (s): A European perspective / S. Krčo, B. Pokrić, F. Carrez // 2014 IEEE world forum on internet of things (WF-IoT). – 2014.

33. Van Kranenburg R., Bassi A. IoT challenges / R. Van Kranenburg, A. Bassi // Communications in Mobile Computing. – 2012. – P. 1-5.

34. Sharma R., Khyati, Singh H., Joshi C. A Study On IoT – Botnet Detection Techniques / R. Sharma, Khyati, H. Singh, C. Joshi // Journal of Current Research in Engineering. – 2022.

35. Threat Hunter Team. Threat Landscape Trends – Q1 2020 / Threat Hunter Team // Symantec Enterprise Blogs. – 2020, June 9. – [Electronic resource]. – Access mode: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/threat-landscape-q1-2020.

36. Roberts P.F. Norsk Hydro Hit with ‘Severe’ LockerGoga Ransomware Attack / P. F. Roberts // The Security Ledger. –2019, March19. – [Electronic resource]. – Access mode: https://securityledger.com/2019/03/norsk-hydro-hit-with-severe-lockergoga-ransomware-attack/.

37. Malware Must Die! A re-emerged IoT threat [Electronic resource] // Blog Malware must die! – 2020. – Access mode: https://blog.malwaremustdie.org/2020/02/mmd-0065-2021-linuxmirai-fbot-re.html.

38. Goodin D. Attackers can force Amazon Echos to hack themselves with self-issued commands / D. Goodin // Ars Technica. –2022. – [Electronic resource]. –Access mode: https://arstechnica.com/information-technology/2022/03/attackers-can-force-amazon-echos-to-hack-themselves-with-self-issued-commands/.

39. Zhang Y., Zhang J., Li X., Liu Y., Zhang Y. A Novel Spoofing Generator Using Vector Tracking-Based Software-Defined Receiver for GPS Anti-Spoofing Research / Y. Zhang, J. Zhang, X. Li, Y. Liu, Y. Zhang // Sensors. –2019.

40. Fruhlinger J. Equifax data breach FAQ: What happened, who was affected, what was the impact? / J. Fruhlinger // CSO Online. – 2020. – [Electronic resource]. – Access mode: https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html.

41. Ringeval M., Wagner G., Denford J., Paré G., Kitsiou S. Fitbit-Based Interventions for Healthy Lifestyle Outcomes: Systematic Review and Meta-Analysis / M. Ringeval, G. Wagner, J. Denford, G. Paré, S. Kitsiou // J Med Internet Res. – 2020.

42. Meneghello F., Calore M., Zucchetto D., Polese M., Zanella A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices / F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella // in IEEE Internet of Things Journal. – 2019.

43. Evenepoel C., Clevers E., Deroover L., Van Loo W., Matthys C., Verbeke K. Accuracy of Nutrient Calculations Using the Consumer-Focused Online App MyFitnessPal: Validation Study / C. Evenepoel, E. Clevers, L. Deroover, W. Van Loo, C. Matthys, K. Verbeke // J Med Internet Res. – 2020.

44. FortiGuard SE Team. Reaper: The Next Evolution of IoT Botnets / FortiGuard SE Team // Fortinet. – 2017. – [Electronic resource]. – Access mode: https://www.fortinet.com/blog/threat-research/reaper-the-next-evolution-of-iot-botnets.

45. Ali K., Askar S. Security Issues and Vulnerability of IoT Devices / K. Ali, S. Askar // International Journal of Science and Business. – 2021. – P. 101-115.

46. FBI. Cyber Tip: Be Vigilant with Your Internet of Things (IoT) Devices / FBI // [Electronic resource]. – 2015. – Access mode: https://www.fbi.gov/news/stories/cyber-tip-be-vigilant-with-your-internet-of-things-iot-devices.

47. Ponemon Institute. The Internet of Things: A New Era of Third-Party Risk / Ponemon Institute // Shared Assessments Program. – 2020 – [Electronic resource]. – Access mode: https://www.ponemon.org/local/upload/file/IoT%20and%20Third%20Party%20Risk%20Final1.pdf.

48. Tidy J. Predatory Sparrow: Who are the hackers who say they started a fire in Iran? / J.Tidy // BBC News. – 2022. – [Electronic resource]. – Access mode: https://www.bbc.com/news/technology-62072480.

49. Storm D. Hackers demonstrated first ransomware for IoT thermostats at DEF CON / D. Storm // Computerworld. – 2016, August8. – [Electronic resource]. – Access mode: https://www.computerworld.com/article/3105001/hackers-demonstrated-first-ransomware-for-iot-thermostats-at-def-con.html.

50. Hogan M., Piccarreta B. Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT) / M. Hogan, B. Piccarreta // NIST Interagency/Internal Report (NISTIR). – National Institute of Standards and Technology, Gaithersburg, MD. – 2023. – [Electronic resource]. – Access mode: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=925023.

51. E. Barker. Recommendation for Key Management: Part 1 – General (NIST Special Publication 800-57) / National Institute of Standards and Technology // [Electronic resource]. – 2020. – Access mode: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final.

52. M. Hogan, B. Piccarreta. Interagency Report on Status of International Cybersecurity Standardization for the Internet of Things (IoT) (NISTIR 8200) / National Institute of Standards and Technology // [Electronic resource]. – 2018. – Access mode: https://csrc.nist.gov/publications/detail/nistir/8200/final.

53. Bellardo J., Savage S. 802.11i: The Advanced Encryption Standard (AES) and Security in the IEEE 802.11 Wireless LAN Standard / J. Bellardo, S. Savage // IEEE Communications Surveys & Tutorials. – 2003. – P. 2-17.

54. IEEE Standard 802.11i-2004: Amendment 6: Medium Access Control (MAC) Security Enhancements [Electronic resource] // IEEE Computer Society. – 2004. – Access mode: https://ieeexplore.ieee.org/document/1318903.

55. Garcia-Morchon O., Kumar S., Sethi M. Internet of Things (IoT) Security: State of the Art and Challenges (RFC 8576) / O. Garcia-Morchon, S. Kumar, M. Sethi // Internet Research Task Force (IRTF). – 2019. – [Electronic resource]. – Access mode: https://datatracker.ietf.org/doc/rfc8576/.

56. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) [Electronic resource] // Official Journal of the European Union. - L119/1,4.5.2016. – 2016. – Access mode: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

57. California Consumer Privacy Act of 2018 [CCPA], Sections 1798.81.5(a), 1798.81.5(b) [Electronic resource]. – 2018. – Access mode: https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=CIV&sectionNum=1798.81.5.

58. Schneier B. Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C / B.Schneier // John Wiley & Sons,Inc. – 1996.

59. Van Tilborg H.C.A., Jajodia S. Encyclopedia of Cryptography and Security / H.C.A. van Tilborg, S. Jajodia // Springer. – 2011.

60. Gao S., Chen H., Fan L. Padding Oracle Attack on PKCS#1 v1.5: Can Non-standard Implementation Act as a Shelter? / S. Gao, H. Chen, L. Fan // International Conference on Cryptology and Network Security. – 2013.

61. Stallings W. Cryptography and network security: principles and practice / W. Stallings // Pearson. – 2017. – P. 277.

62. Onatsky A.V., Yona L.G. Asymmetric encryption methods / A.V.Onatsky,L. G. Yona // Module 2: Cryptographic methods of information protection in telecommunication systems and networks: Textbook manual / N. V. Zakharchenko, Odesa: ONAS named after A. S. Popova. – 2010.

63. Царук Д., Фесенко А. Застосування асиметричної криптографії для безпечного документообігу / Д. Царук, А. Фесенко // Інформація, комунікація, суспільство 2023: Матеріали XII-ї Міжнародної наукової конференції ІКС-2023. – 2023. – с. 40-41.

64. Schneier B. Applied cryptography: protocols, algorithms, and source code in C (2nd ed.) / B. Schneier // John Wiley & Sons. – 2007. – P. 620-622.

65. RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard / RSA Laboratories // [Electronic resource]. – 2002. – Access mode: http://www.rsa.com

66. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems / R. Anderson // John Wiley & Sons. – 2001.

67. Simeonov D. Discrete Log, CDH, and DDH / D. Simeonov // [Electronic resource]. – 2011. – Access mode: http://theory.stanford.edu/~dfreeman/cs259c-f11/finalpapers/pollardrho.pdf.

68. Lopez J., Dahab R. Fast multiplication on elliptic curves over GF(2^m) without precomputation / J. Lopez, R. Dahab // – Springer. – 1999.

69. Xiang S., Lai S., Meng Y. Performance Comparison of Elliptic Curve and RSA Digital Signatures / S. Xiang, S. Lai, Y. Meng // Zhongguo Zhong Xi Yi Jie He Za Zhi [Electronic resource]. – 2009. – Access mode: http://www.ncbi.nlm.nih.gov/pubmed/20329605.

70. Bafandehkar M., Yasin S., Mahmod R., Hanapi Z.M. Comparison of ECC and RSA Algorithm in Resource Constrained Devices / M. Bafandehkar, S. Yasin, R. Mahmod, Z. M. Hanapi // 2013 International Conference on IT Convergence and Security (ICITCS). – 2013.

71. Alam M. A Comparative Study of RSA and ECC and Implementation of ECC on Embedded Systems / M. Alam // Researchgate. – 2016. – P. 86–93.

72. P. K. Dhillon and S. Kalra. Elliptic curve cryptography for real time embedded systems in IoT networks // 5th International Conference on Wireless Networks and Embedded Systems (WECON). – 2016. P. 1-6.

73. Chatzigiannakis I., Vitaletti A., Pyrgelis A. A privacy-preserving smart parking system using an IoT elliptic curve based security platform [Electronic resource] // Computer Communications. – 2016. – Access mode: http://dx.doi.org/10.1016/j.com.

74. Al Hamid H.A., Rahman S.M.M., Shamim Hossain M., Almogren A., Alamri A. Secure Lightweight ECC-Based Protocol for Multi-Agent IoT Systems / H.A. Al Hamid, S.M.M. Rahman, M. Shamim Hossain, A. Almogren, A. Alamri // 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). – 2017.

75. Yang H., Oleshchuk V., Prinz A. Verifying group authentication protocols by Scyther / H. Yang, V. Oleshchuk, A. Prinz // – 2016. – P. 3-19.

76. Mahto D. RSA and ECC : A Comparative Analysis / D.Mahto // International Journal of Applied Engineering Research 12(19):9053-9061. – 2017.

77. Chhabra A., Arora S. An Elliptic Curve Cryptography Based Encryption Scheme for Securing the Cloud against Eavesdropping Attacks / A. Chhabra, S. Arora // IEEE 3rd International Conference Collaboration Internet Computing CIC 2017. – 2017.

78. Al Hamid H.A., Rahman S.M.M., Shamim Hossain M., Almogren A., Alamri A. EdgeTrust: A Lightweight Data-Centric Trust Management Approach for IoT-Based Healthcare 4.0 / H.A.Al Hamid, S.M.M. Rahman, M. Shamim Hossain, A. Almogren, A. Alamri // IEEE Access. – 2017.

79. Suárez-Albela M., Fraga-Lamas P., Castedo L., Fernández-Caramés T. Clock Frequency Impact on the Performance of High-Security Cryptographic Cipher Suites for Energy-Efficient Resource-Constrained IoT Devices / M. Suárez-Albela, P. Fraga-Lamas, L. Castedo, T. Fernández-Caramés // Sensors [Electronic resource]. – 2018. – Access mode: http://www.mdpi.com/1424-8220/19/1/15.

80. Su M., Fern T.M. A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices / M.Su,T.M.Fern // 2018 Global Internet of Things Summit (GIoTS). – 2018.

81. Suárez-Albela M., Fraga-Lamas P., Fernández-Caramés T. A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices / M.Suárez-Albela,P.Fraga-Lamas,T.Fernández-Caramés // Sensors [Electronic resource]. – 2018. Access mode: http://www.mdpi.com/1424-8220/18/11/3868.

82. Cook J.D. A tale of two elliptic curves [Electronic resource] / John D. Cook Consulting. – 2018. – Access mode: https://www.johndcook.com/blog/2018/08/21/a-tale-of-two-elliptic-curves/.

83. Albalas F., Al-Soud M., Almomani O., Almomani A. Security-aware CoAP application layer protocol for the internet of things using elliptic-curve cryptography / F. Albalas, M. Al-Soud, O. Almomani, A. Almomani // Int Arab J Inf Technol. – 2018.

84. Kaedi S., Doostari M.A., Ghaznavi-Ghoushchi M.B. Low-complexity and differential power analysis (DPA)-resistant two-folded power-aware Rivest–Shamir–Adleman (RSA) security schema implementation for IoT-connected devices / S. Kaedi, M. A. Doostari, M. B. Ghaznavi-Ghoushchi // IET Comput Digit Tech [Electronic resource].– 2018. – Access mode: http://digital-library.theiet.org/content/journals/10.1049/iet-cdt.2018.5098.

85. Vahdati Z., Ghasempour A., Salehi M., Md Yasin S. Comparison of ECC and RSA algorithms in IoT devices / Z. Vahdati, A. Ghasempour, M. Salehi, S. Md Yasin // Journal of Theoretical and Applied Information Technology. – 2019.

86. Johnson L. Statutory and Regulatory GRC / L. Johnson // Security Control Evaluation Testing, Assess Handb.– 2015 . – P. 11–33.

87. Signature Verification in Real Time // [Electronic resource]. – Access mode: https://www.xyzmo.com/e-signature-products/signature-verification.

88. Maqsood F, Ahmed M, Mumtaz M, Ali M. Cryptography: A Comparative Analysis for Modern Techniques / Maqsood F, Ahmed M, Mumtaz M, Ali M. // Int J Adv Comput Sci Appl. – 2017.

89. Bharathi B., Manivasagam G., MAK. Metrics for Performance Evaluation of Encryption Algorithms / B.Bharathi,G.Manivasagam // International Journal of Advance Research in Science and Engineering. – 2017. – P. 106.

90. RSA vs ECC – Which is Better Algorithm for Security? // [Electronic resource]. – Access mode: https://www.ssl2buy.com/wiki/rsa-vs-ecc-which-is-better-algorithm-for-security.

91. Harsha A., Patil B. A Review: Security of Data in Cloud Storage using ECC Algorithm / A. Harsha, B. Patil // Bonfring Int J Softw Eng Soft Comput. – 2017. – P. 143–146.

92. Olenski J. ECC 101: What is ECC and why would I want to use it? [Electronic resource] // – 2015. – Access mode: https://www.globalsign.com/en/blog/elliptic-curve-cryptography/.

93. Xiang S., Lai S., Meng Y. Performance Comparison of Elliptic Curve and RSA Digital Signatures / S. Xiang, S. Lai, Y. Meng // Zhongguo Zhong Xi Yi Jie He Za Zhi [Electronic resource]. – 2004. – Access mode: http://www.ncbi.nlm.nih.gov/pubmed/20329605.

94. Viega J., McGrew D. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP) (RFC 4106) / J. Viega, D. McGrew // Internet Engineering Task Force (IETF) [Electronic resource]. – 2005. – Access mode: https://www.rfc-editor.org/rfc/rfc4106.

95. Ahmad N., Wei L.M., Jabbar M.H. Advanced Encryption Standard with Galois Counter Mode using Field Programmable Gate Array / N. Ahmad, L. M. Wei, M. H. Jabbar // In Journal of Physics: Conference Series. – 2018.

96. McGrew D., Viega J. The Galois/Counter Mode of Operation (GCM) / D. McGrew, J. Viega // Submission to NIST [Electronic resource]. – 2004. – Access mode: http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf.– January2004.

97. Harkins D., Carrel D. The Internet Key Exchange (IKE) / D. Harkins, D. Carrel // RFC2409. – 1998.

98. Lochter M., Merkle J. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation (RFC 5639) / M. Lochter, J. Merkle // RFC Editor [Electronic resource]. – 2010. – Access mode: https://www.rfc-editor.org/rfc/rfc5639.

99. SEC 2: Recommended Elliptic Curve Domain Parameters [Electronic resource] // Standards for Efficient Cryptography Group (SECG). – 2010. – Access mode: http://www.secg.org/sec2-v2.pdf.

100. Houria A., Azine B., Abdelkader G. A comparison between the secp256r1 and the koblitz secp256k1 bitcoin curves / A.Houria, B.Azine, G.Abdelkader // Indonesian Journal of Electrical Engineering and Computer Science. – 2019.

101. Fei X., Li K., Yang W. A fast parallel cryptography algorithm based on AES-NI / X. Fei, K. Li, W. Yang // Journal of Intelligent & Fuzzy Systems. – 2016. – P. 1-9.

ДОДАТОК А

Програмна реалізація покращення механізму шифрування та дешифрування ECC симетричним алгоритмом AES-GCM

```
from tinyec import registry
import hashlib
import secrets
import binascii
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend

def generate_encryption_keys(pub_key):
    private_key = secrets.randbelow(curve.field.n)
    public_key = private_key * curve.g

    shared_key = pub_key * private_key
    secret_key = derive_256_bit_key(shared_key)

    ciphertext, nonce, auth_tag = encrypt_AES_GCM(message, secret_key)

    return (ciphertext, nonce, auth_tag, public_key)

def generate_decryption_key(private_key, ciphertext, nonce, auth_tag, public_key):
    shared_key = public_key * private_key
    secret_key = derive_256_bit_key(shared_key)

    plaintext = decrypt_AES_GCM(ciphertext, nonce, auth_tag, secret_key)
    return plaintext


def encrypt_AES_GCM(message, secret_key):
    aes_key = secret_key[:16]
    nonce = secrets.token_bytes(12)
    cipher = Cipher(algorithms.AES(aes_key), modes.GCM(nonce), backend=default_backend())
```

```
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(message) + encryptor.finalize()
    auth_tag = encryptor.tag

    return ciphertext, nonce, auth_tag


def decrypt_AES_GCM(ciphertext, nonce, auth_tag, secret_key):
    aes_key = secret_key[:16]
    cipher = Cipher(algorithms.AES(aes_key), modes.GCM(nonce, auth_tag), backend=default_backend())

    decryptor = cipher.decryptor()
    plaintext = decryptor.update(ciphertext) + decryptor.finalize()
    return plaintext


def derive_256_bit_key(point):
    sha = hashlib.sha256(int.to_bytes(point.x, 32, 'big') + int.to_bytes(point.y, 32, 'big'))
    return sha.digest()

if __name__ == "__main__":

    curve = registry.get_curve('secp256r1')

    message = b'Text to be encrypted by ECC public key and decrypted by its corresponding ECC private key'

    print("Original message:", message)

    private_key = secrets.randbelow(curve.field.n)
    public_key = private_key * curve.g

    encrypted_message = generate_encryption_keys(public_key)
    encrypted_message_obj = {
        'ciphertext': binascii.hexlify(encrypted_message[0]),
        'nonce': binascii.hexlify(encrypted_message[1]),
        'auth_tag': binascii.hexlify(encrypted_message[2]),
        'public_key': hex(encrypted_message[3].x) + hex(encrypted_message[3].y % 2)[2:]
    }
```

```
print("Encrypted message:", encrypted_message_obj)

decrypted_message = generate_decryption_key(private_key, encrypted_message[0], encrypted_message[1],
                                            encrypted_message[2], encrypted_message[3])
print("Decrypted message:", decrypted_message)
```

ДОДАТОК Б

Таблиця Б.1

Споживання енергії згідно дослідження «Практичне порівняння продуктивності ECC і RSA для пристроїв IoT з обмеженими ресурсами (2018)»

Дослідження	Рівень безпеки	Розмір ключа (біт)		Споживання енергії (мВт*год)	
		RSA	ECC	RSA	ECC
Практичне порівняння продуктивності ECC і RSA для пристроїв IoT з обмеженими ресурсами	80	1024	192	17.86	9.05
	112	2048	224	21.55	17.38
	128	3072	256	56,78	15.43
	192	7680	384	-	22.26

ДОДАТОК В

Таблиця В.1

Споживання енергії за різними показниками частот

Дослідження	Рівень безпеки	Розмір ключа (біт)		Частота МГц	Споживання енергії (мВт*год)	
		RSA	ECC		RSA	ECC
Вплив тактової частоти на продуктивність високозахищених криптографічних шифрів для пристроїв IoT з обмеженим енергоспоживанням	80	1024	192	80	~20	~11.58
				160	~13.41	~9,75
				240	~13,63	~8,26
	112	2048	224	80	~31	~17.27
				160	~20,69	~13.06
				240	~18.18	~13.80
	128	3072	256	80	~67,5	~20,83
				160	~42,5	~14.21
				240	~38,41	~12,97
	192	7680	384	80	-	~28,84
				160	-	~19.26
				240	-	~17.35

ДОДАТОК Д

Таблиця Д.1

Час генерації та виконання ключа

№	Стаття	Рівень безпеки	Розмір ключа		Час генерації ключа(мс)		Час виконання (мс)		
			RSA	ECC	RSA	ECC	Примітки	RSA	ECC
1.	Модель безпеки для збереження конфіденційності медичних великих даних у хмарі охорони здоров'я з використанням обчислювальної системи Fog із криптографією на основі пар	80	512	106	383	57	Операція з відкритим ключем	430	810
							Операція закритого ключа	10990	810
		112	768	132	889	98	Операція з відкритим ключем	1940	2190
							Операція закритого ключа	83260	2190
		128	1024	160	2609	108	-	-	-
		160	2048	210	18399	121	-	-	-
2.	Захищений легкий протокол на основі ECC для мультиагентних систем IoT	80	2048	224	-	-	Запит	78,77	32,66
					-	-	Розшифровка	34,89	33.40
					-	-	Результат	16.49	0,81
		112	3072	256	-	-	Запит	181,8	32,66
					-	-	Розшифровка	34,93	33.40
					-	-	Результат	48.51	0,81

продовження табл. Д.1

3.	Криптографія еліптичної кривої для вбудованих систем реального часу в мережах IoT	80	1024	160-233	~2000	~476	-	-	-
		112	2048	224-255	~15333	~857	-	-	-
4.	Порівняльне дослідження RSA та ECC і впровадження ECC у вбудовані системи (2016) Безпека даних у хмарному сховищі за допомогою алгоритму ECC	80	1024	163	160	80	-	-	-
		112	2240	233	7470	180	-	-	-
5.		128	3072	283	9800	270	-	-	-
		192	7680	409	133900	640	-	-	-
		256	15360	571	679060	1440	-	-	-

ДОДАТОК Е

Таблиця Е.1

Час шифрування та дешифрування

Стаття	Рівень безпеки	Розмір ключа		К-сть біт	Час шифрування/дешифрування (мс)				
		RSA	ECC		Загальний час		Примітки	RSA	ECC
					RSA	ECC			
RSA та ECC: Порівняльний Аналіз	80	1024	160	8	785	1815,2	Ш	30.7	488,5
							Р	754.3	1326,7
				64	5673,8	8078,4	Ш	136.6	2168,5
							Р	5537,2	5909,9
				256	19877,2	30809,1	Ш	559,6	7924
							Р	19317,7	22885,1
	112	2048	24	8	2737,5	3789,3	Ш	29.9	2203
							Р	2707,5	1586,3
				64	20574,3	16918,8	Ш	163.5	9985,5
							Р	20410,8	6933,3
				256	102615,3	66033,9	Ш	581,5	39700,8
							Р	102033,7	26333,1
	128	3072	256	8	6971,4	5645,3	Ш	30.5	3876,3
							Р	6940,9	1769
				64	46645,4	22446,6	Ш	167.2	15088,2
							Р	46478,2	7358,4
				256	210169,7	85844,6	Ш	561.1	58438,6
							Р	209608,6	27406
	144	-	-	8	13696,2	6728,8	Ш	48.9	4726,6
							Р	13647,2	2002.2
				64	77902,7	28709,3	Ш	138.5	20230,8
							Р	77764,2	8478,5
				256	311636,8	109655,6	Ш	571,8	77503,4
							Р	311064,9	32152,2

Згідно таблиці Е.1, стовпець «Примітки», шифрування позначено, як «Ш», а розшифрування, як «Р».

ДОДАТОК Ж

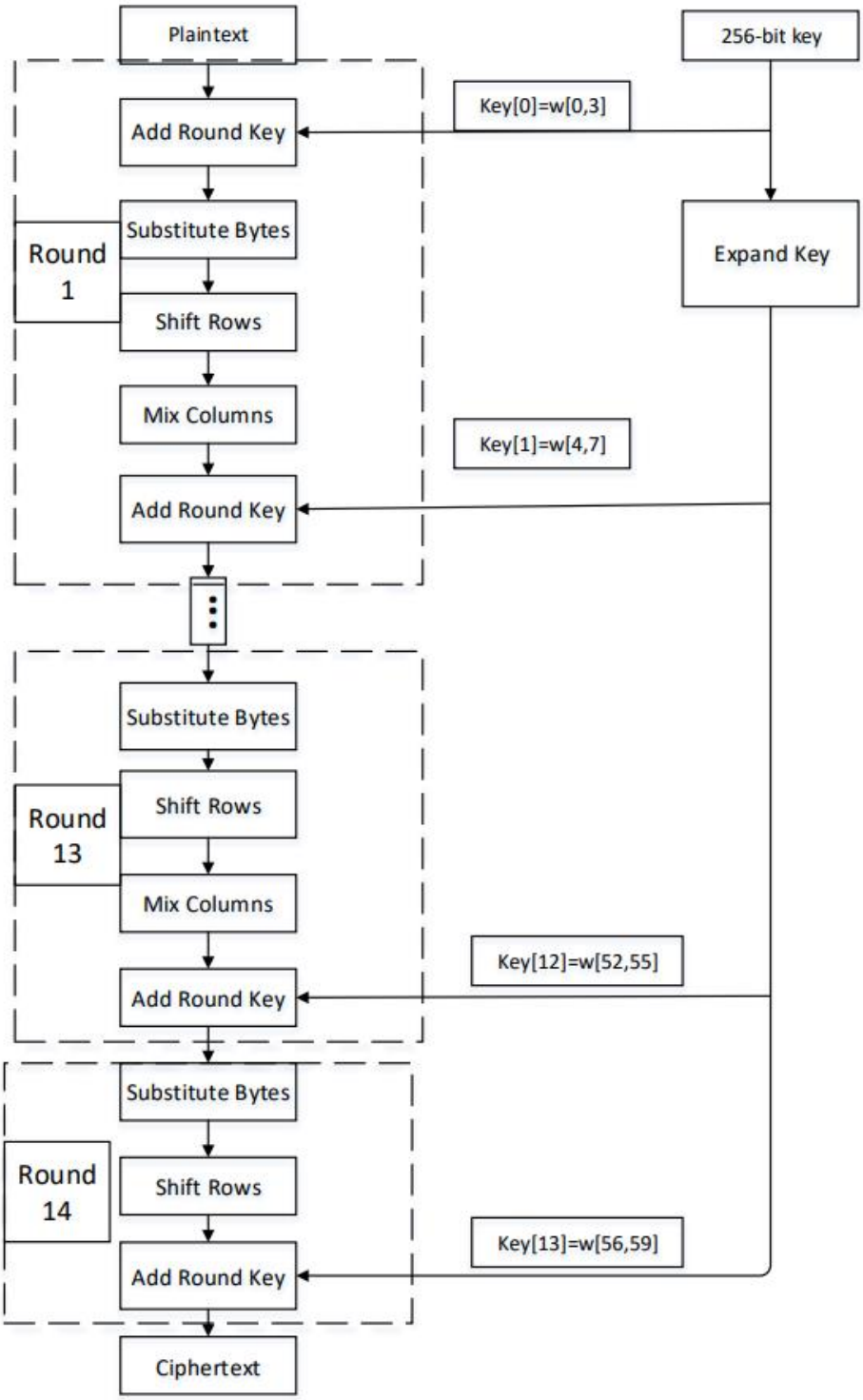

Рисунок Ж.1 – Архітектура шифрування AES-GCM із довжиною ключа 256 біт

ДОДАТОК З

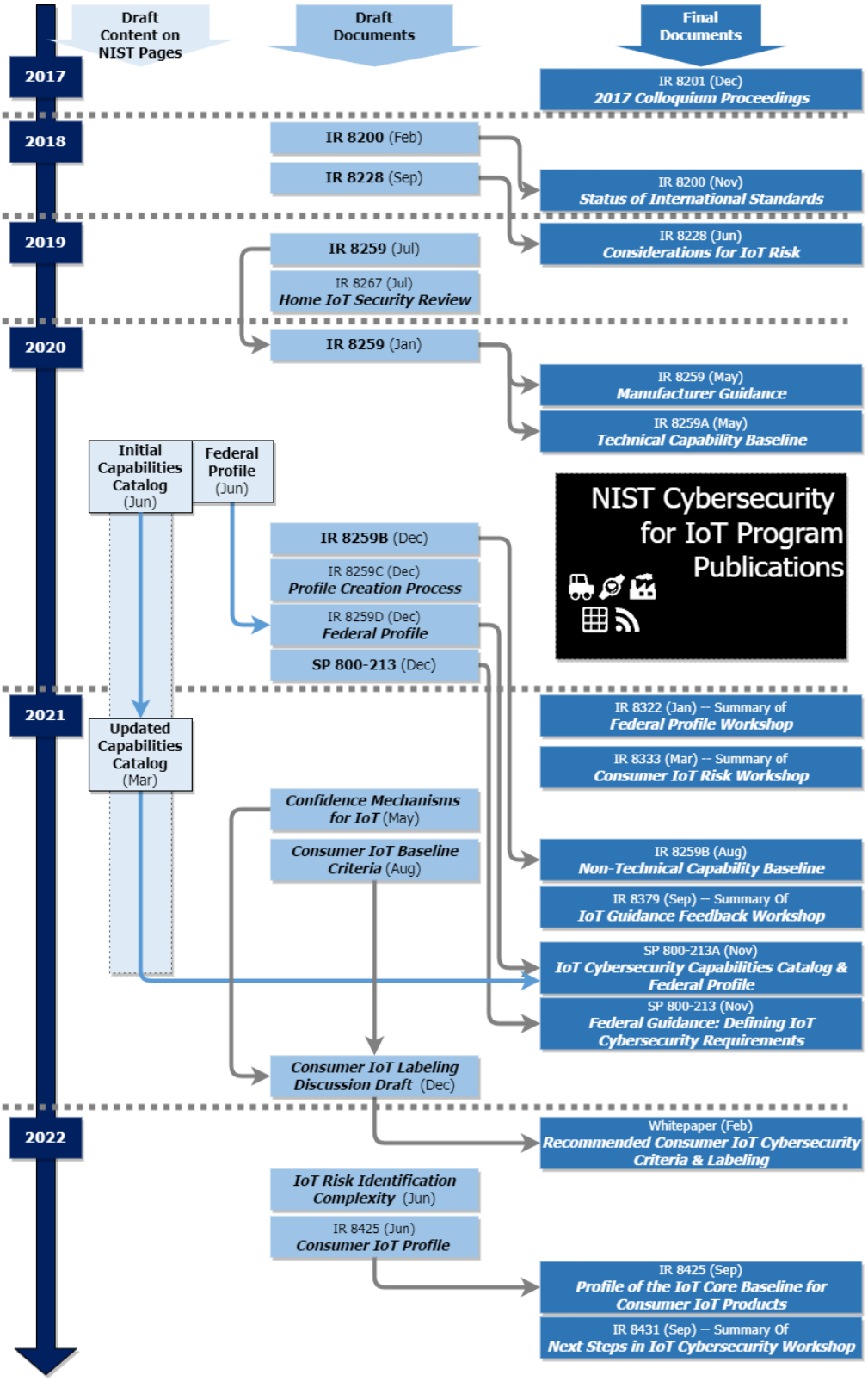

Рисунок 3.1 – Стадії розробок публікацій, досліджень, рекомендацій NIST для IoT