ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ТЕХНОЛОГІЯ СТВОРЕННЯ ШИФРОВАНОЇ КОМУНІКАЦІЇ PEER-TO-PEER ПРИСТРОЇВ НА БАЗІ ТЕХНОЛОГІЇ LORA»

на здобуття освітнього ступеня магістра
зі спеціальності 125 Кібербезпека
(код, найменування спеціальності)
освітньо-професійної програми Інформаційна та кібернетична безпека
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
Микита ЄРЬОМЕНКО

Виконав: здобувач(ка) вищої освіти групи БСДМ-61
ЄРЬОМЕНКО Микита
(ПРІЗВИЩЕ, Ім'я)

Керівник: БОРСУКОВСЬКИЙ Юрій
к.т.н., доцент (ПРІЗВИЩЕ, Ім'я)

Рецензент:
(ПРІЗВИЩЕ, Ім'я)

Київ 2024

ЗМІСТ

Стор.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LoRa - Long Range

LoRaWAN - Long Range Wide Area Network

IoT - Internet of Things

LPWAN - Low-power wide-area network

CSS - Chirp spread spectrum

SF - Spreading factor

SFD - Start frequency delimiter

MAC - Media Access Control

MIC - Message Integrity Code

OTAA - Over The Air Activation

ABP - Activation By Personalization

MITM – Man In The Middle attack

ECC - Elliptic Curve Cryptography

NTRU - N-th Degree Truncated Polynomial Ring Units

P2P - Peer-To-Peer

ACK - Acknowledgment

ВСТУП

Актуальність дослідження. У сфері сучасних технологій бездротового зв'язку, технологія LoRa займає значне місце. Завдяки її великому потенціал у створенні енергоефективних і довготривалих мереж, які можуть працювати на великій дистанції, ця технологія здобула свою популярність не тільки серед поодиноких ентузіастів зі сфери телекомунікацій, а й серед виробників обладнання, особливо для застосувань в сфері Інтернету речей. Однак для її ефективного застосування, особливо у критично важливих сферах зростає необхідність в забезпеченні надійного рівня безпеки і конфіденційності переданої інформації.

Зі зростанням популярності технологія LoRa та використанням IoT пристроїв, побудованих на її основі, у багатьох сферах, від персонального використання та охоронних систем до смарт-міст, систем моніторингу здоров’я та промислових застосувань, стає все більш очевидною вразливість цієї систем до кіберзагроз. Питання забезпечення надійної та конфіденційної передачі даних є особливо критичним, оскільки розкриття або модифікація інформації може призвести до серйозних наслідків. Використання LoRa для безпечних peer-to-peer з'єднань відкриває перспективи вирішення цієї проблеми.

Не менш важливим фактом є те, що технологія LoRa знайшла своє застосування у військовій сфері та сфері оборони. Багато як військових організацій, так і звичайних волонтерів виготовляють свої прилади, використовуючи технологію LoRa через її великий радіус дії так низьке енергоспоживання, що є дуже важливими факторами в бойових умовах. Тому технологія LoRa особливо потребує досліджень у напрямку створення захищених peer-to-peer комунікацій у децентралізованій мережі, оскільки уразливість таких систем може призвести до серйозних наслідків, адже компрометація інформації або її неправильне передавання можуть порушити роботу критичних об’єктів оборони. Це вимагає надійних і адаптивних методів шифрування, що враховують

обмежені ресурси пристроїв, з мінімальним енергоспоживанням, але при цьому забезпечують високий рівень захисту системи.

Об'єкт дослідження – встановлення захищеного з'єднання між пристроями в децентралізованій peer-to-peer мережі.

Предмет дослідження – алгоритм створення захищеного з'єднання в децентралізованій peer-to-peer мережі на базі технології LoRa.

Мета роботи – розробити ефективний алгоритм встановлення захищеного з'єднання в децентралізованій LoRa-мережі, враховуючи обмеження пропускної здатності та енергетичні ресурси пристроїв.

Наукові завдання:

- Оцінити особливості та обмеження технології LoRa у контексті безпечної передачі даних;
- Виявити недоліки існуючих рішень шифрованої комунікації на основі технології LoRa для децентралізованої мережі.
- Дослідити існуючі методи шифрування даних у мережах;
- Розробити ефективний алгоритм встановлення шифрованого peer-to-peer з'єднання з урахуванням обмежень пристроїв.

Методи дослідження – опрацювання літератури за даною темою, аналіз документації LoRa та LoRaWAN, міжнародних стандартів та їх порівняння.

Практичне значення одержаних результатів: розроблено ефективний алгоритм для створення захищеного каналу зв'язку в децентралізованій peer-to-peer мережі LoRa-пристроїв, з реалізацією механізмів автентифікації та захисту від атак, що не пов'язані з шифруванням даних.

Результати кваліфікаційної роботи апробовані на Всеукраїнській науковій конференції «Актуальні проблеми кібербезпеки», яка відбулася 25 жовтня 2024 року в Державному університеті інформаційно-комунікаційних технологій, м. Київ.

1 АНАЛІЗ ПРОБЛЕМИ ЗАБЕЗПЕЧЕННЯ ЗАХИЩЕНОЇ КОМУНІКАЦІЇ ПРИСТРОЇВ НА БАЗІ ТЕХНОЛОГІЇ LORA

1.1. Призначення, принцип роботи та особливості використання технології LoRa

Технологія LoRa - це сучасна технологія бездротового зв’язку, що спрощує процеси встановлення бездротового зв'язку та дозволяє передавати дані на значні відстані, споживаючи при цьому мінімальну кількість енергії.

Розроблена для задоволення потреб Інтернету речей (IoT), ця технологія йде далі, забезпечуючи надійний та економічно ефективний обмін інформацією, який може бути ускладнений при використанні традиційних систем зв'язку у конкретних умовах. Технологія LoRa з'явилася як відповідь на потребу у створенні систем зв'язку для великої кількості пристроїв, які розміщені на значних територіях та великих відстанях один від одного, без необхідності постійного доступу до традиційних комунікаційних інфраструктур, таких як мобільний зв'язок або Wi-Fi.

На відміну від інших, звичних для нас, технологій безпровідного зв’язку, таких як Wi-Fi, Bluetooth та стільниковий зв'язок, які створені для високошвидкісної передачі великих обсягів даних, LoRa орієнтована на надсилання невеликих пакетів інформації, наприклад, показників температури, положення у просторі або стану пристроїв. Це робить її ідеальною для рішень, які потребують передачі лише основної інформації без зайвих додаткових деталей. Також можна зазначити, що завдяки низькій швидкості передачі та малій пропускній здатності, LoRa суттєво знижує енергоспоживання пристроїв, які використовують її для зв'язку. Це дозволяє сенсорам та іншим IoT-пристроям працювати від батарейок протягом багатьох років без необхідності частого

обслуговування чи заміни цих батарейок. Крім того, використання LoRa вільне від ліцензійних зборів, оскільки вона працює на загальнодоступних частотах [1], що значно знижує витрати на організацію мережі порівняно з ліцензованими технологіями зв'язку, такими як LTE-M чи NB-IoT. На порівняльному графіку (рис. 1.1) видно, що в порівнянні з іншими технологіями безпровідної комунікації, головними особливостями технології LoRa є її низька вартість використання та велика дальність дії, але разом з тим є і недолік у вигляді низької пропускної здатності [2].

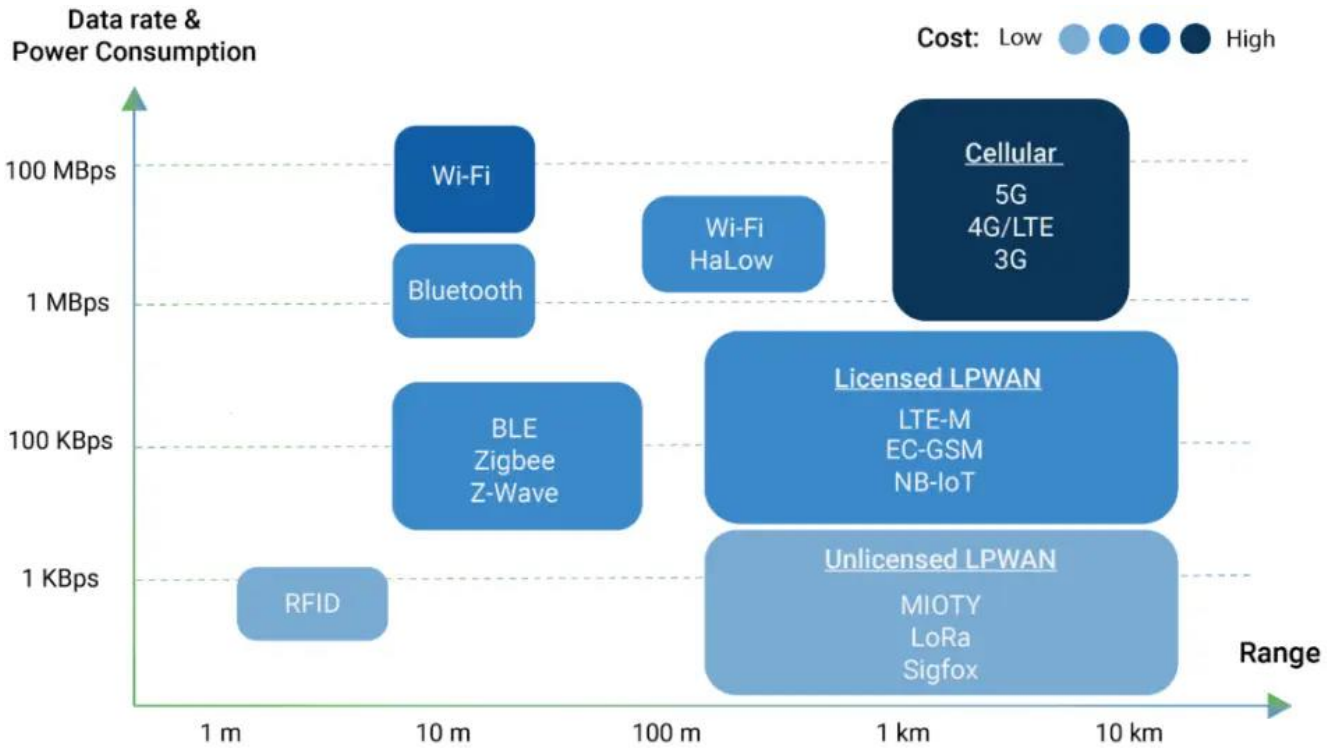

Рис. 1.1. Порівняння LoRa з іншими бездротовими технологіями [2]

Як типовий приклад використання технології LoRa у системах комунікацій можна навести систему моніторингу зрошення на фермі, яка надсилає дані про вологість ґрунту, температура повітря та рівень опадів, що дозволяє оптимізувати використання води і зменшити витрати на обслуговування полів.

Незалежно від того, чи потрібно збирати дані з інтелектуальних лічильників на великих сільськогосподарських територіях або контролювати активи у віддалених місцях, всебічні переваги використання технології LoRa численні, і

саме це стало рушійною силою збільшення кількості сфер її використання та здобуття популярності в останні роки.

Основна мета використання LoRa полягає у забезпеченні з'єднання між пристроями в середовищах, де необхідне довготривале функціонування від автономного, малопотужного та обмеженого енергоносія, такого як батарейка чи акумулятор, де доступ до енергії та інфраструктури є обмеженим, а передача даних на великі відстані є ключовим елементом системи.

Пристрої з технологією LoRa можуть функціонувати в умовах віддаленого розташування та відсутності будь-якої інфраструктури для комунікації, таких як віддалені степи, гірські регіони або густі ліси. Це робить LoRa надзвичайно цінною для різних сфер застосувань, де головним пріоритетом є велика відстань передачі даних та тривале автономне живлення пристроїв.

Щоб задовольнити ці потреби комунікації на далекі відстані та енергоефективності, була розроблена технологія малопотужних глобальних мереж LPWAN яка стала варіантом підключення для мереж IoT. LPWAN включає в себе такі протоколи модуляції, як диференціальний зсув фази, який використовується у SigFox; $\frac{\pi}{2}$ – бінарний зсув фази та $\frac{\pi}{2}$ – квадратурний зсув фази, використовувані у вузько смуговій модуляції IoT, також відомій як NB-IoT та Cat-M2; 16- та 64-квадратурна амплітудна модуляція у довгостроковій еволюції для машин (LTE-M), а також модуляції дальньої дії, відома як LoRa [3].

Основою роботи LoRa є використання *chirp spread spectrum* (CSS) модуляції [4], яка є однією з ключових особливостей технології LoRa. CSS використовує розширений спектр сигналу, що надає сигналу LoRa високу проникну здатність та зменшує втрати сигналу під час розповсюдження.

У традиційних системах бездротового зв'язку дані передаються за допомогою постійної частоти, в той час як в LoRa передача здійснюється за допомогою частотних послідовностей, які змінюються (рис. 1.2). Цей метод заснований на передачі даних через зміни частоти, створюючи унікальні «chirp»

сигнали, що є абревіатурою від «Compressed High Intensity Radar Pulse». Ще такі сигнали називають "співочі" частотні послідовності, і саме вони дозволяють передавати інформацію на великі відстані, аж до 15 км у сільській місцевості та до 5 км у міських умовах [5], а також забезпечують високу стійкість до перешкод і дозволяють сигналу проникати крізь фізичні бар'єри, такі як стіни, дерева, будинки та інше.

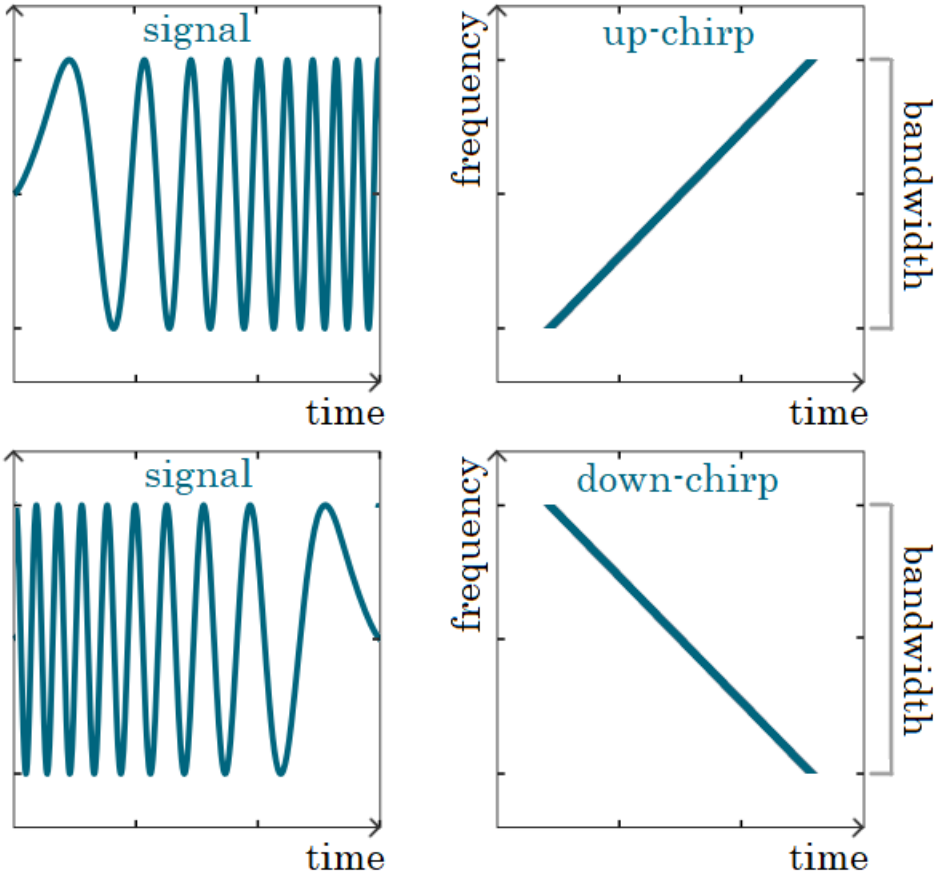

Рис. 1.2. Приклад “chirp” сигналів

Сигнал chirp відноситься до синусоїдальних сигналів, частота якого змінюється з часом. У випадку, коли ця зміна є лінійною, його називають лінійною частотною модуляцією. Якщо частота змінюється від найнижчої до найвищої, це називається up-chirp, а якщо частота змінюється від найвищої до найнижчої, ми називаємо це down-chirp. Такі сигнали широко використовуються у військових радіолокаційних системах. Переваги таких систем порівняно зі звичайним імпульсним радаром включають покращення дальності, роздільної

здатності діапазону, можливість досягти аналогічної продуктивності з нижчою піковою вхідною потужністю, підвищену стійкість до шуму, нижчі вимоги до напруги живлення, а також архітектуру радара, яка пропонує дещо кращу стійкість до перешкод.

Як було вже згадано раніше, LoRa спеціалізується на передачі невеликих пакетів даних із відносно низькою швидкістю — від 0,25 до 50 кбіт/с [6]. Така швидкість передачі є досить малою порівняно з іншими сучасними технологіями безпровідної комунікації, де передача може досягати сотень мегабіт на секунду.

Це пов’язано з тим, що LoRa використовує вузько-смуговий канал для передачі, коли сигнал передається на вузькій ділянці частотного спектра. У зв’язку з цим дані передаються повільніше, але перевагою є можливість надсилати інформацію на значно більші відстані з низькою ймовірністю втрати сигналу.

Вузько-смуговий характер LoRa також забезпечує високу чутливість приймача, яка досягає -137 дБм, що є набагато більш чутливим, ніж у більшості інших технологій бездротового зв’язку. Іншими словами, LoRa пристрої можуть вловлювати слабкі сигнали, що є ключовою перевагою для роботи на великих відстанях та у важких умовах. Висока чутливість приймача також означає, що навіть якщо сигнал надходить із певними перешкодами або зниженням інтенсивності, що трапляється при подоланні великих відстаней чи фізичних бар’єрів, LoRa приймач здатен його вловити і розшифрувати, що значно поліпшує надійність та стабільність такого з’єднання.

Слід зазначити, що налаштування швидкості передачі даних від 0,25 до 50 кбіт/с відбувається за рахунок того, що LoRa використовує такий параметр передачі даних як spreading factor (SF), тобто фактор поширення. Цей параметр є налаштовуваний, і може приймати значення від SF7 до SF12, де при кожному наступному значенні цього параметру, тривалість ефірного часу для передачі кожного біту збільшується рівно вдвічі, що збільшує час на передачу одного й

того самого повідомлення, але й підвищує чутливість приймача [7]. Взаємовідношення ефірного часу до значення параметру SF можна побачити на рисунку 1.3.

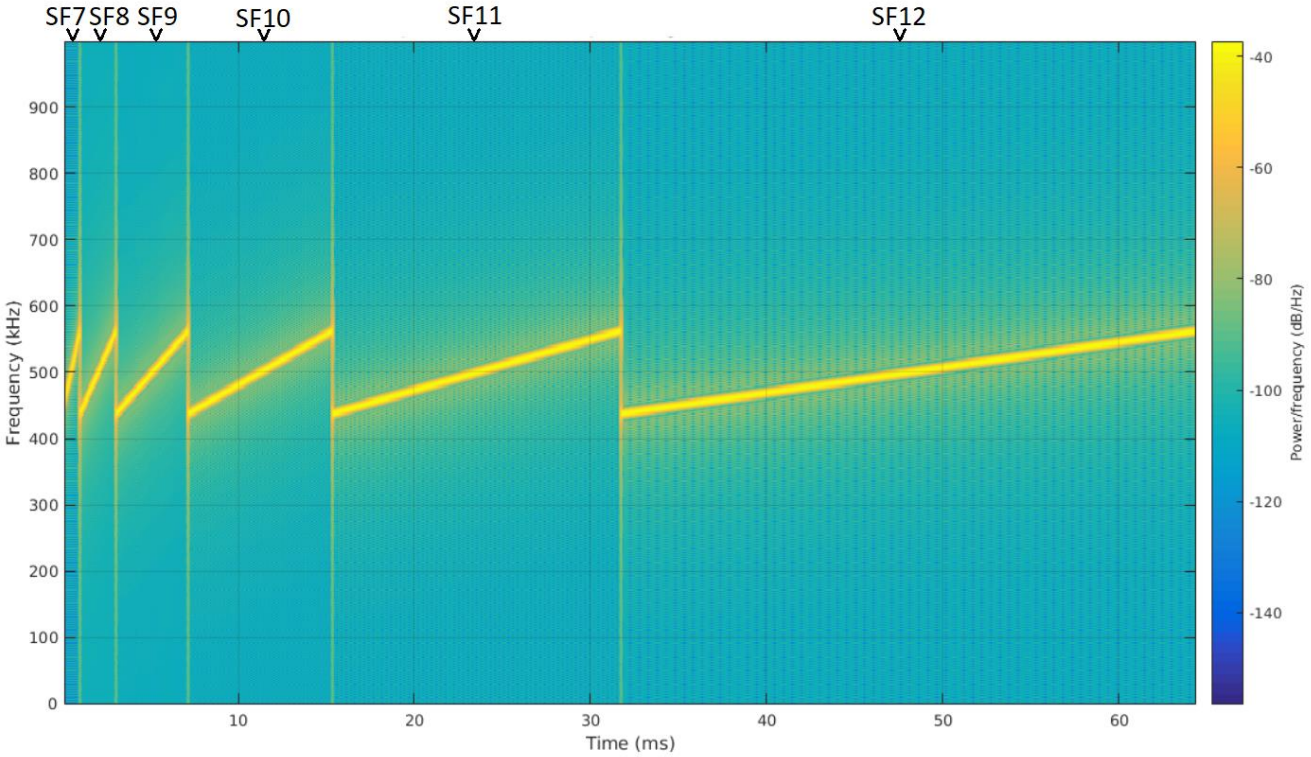

Рис. 1.3. Спектрограма порівняння параметру SF при різних значеннях

Різні фактори поширення дозволяють користувачу LoRa підлаштовувати передачу сигналу під конкретні потреби та умови мережі. Вибір значення SF є компромісом між дальністю зв'язку, швидкістю передачі, енергоспоживанням і стійкістю сигналів до шуму.

Так, наприклад, значення параметру SF7 підходить для тих ситуацій коли пристрій знаходяться ближче до приймача, оскільки це забезпечує більшу швидкість передачі даних, але натомість дальність є обмеженою. В той час як SF12, навпаки, дозволяє передавати на значно більші відстані, однак швидкість передачі при цьому суттєво падає.

У сучасних мережах, де використовується технологія LoRa, є можливість налаштувати автоматичний вибір оптимального значення SF на основі оцінки

поточної потужності сигналу, заряду батареї, та інших факторів, що допомагає підтримувати ефективну роботу такої мережі.

Низька швидкість передачі даних у технології LoRa також частково обумовлена її спеціальною структурою повідомлень [8], яка забезпечує надійність зв'язку і точну синхронізацію пристроїв до початку передачі даних. Ця структура (рис. 1.4) включає в себе такі важливих компонентів, як преамбула та символи синхронізації, які дозволяють пристроям LoRa чітко налаштовуватися один на одного, гарантуючи, що передане повідомлення буде отримано і оброблено коректно, навіть за умов низької потужності сигналу, великої відстані, і наявності шуму в ефірі.

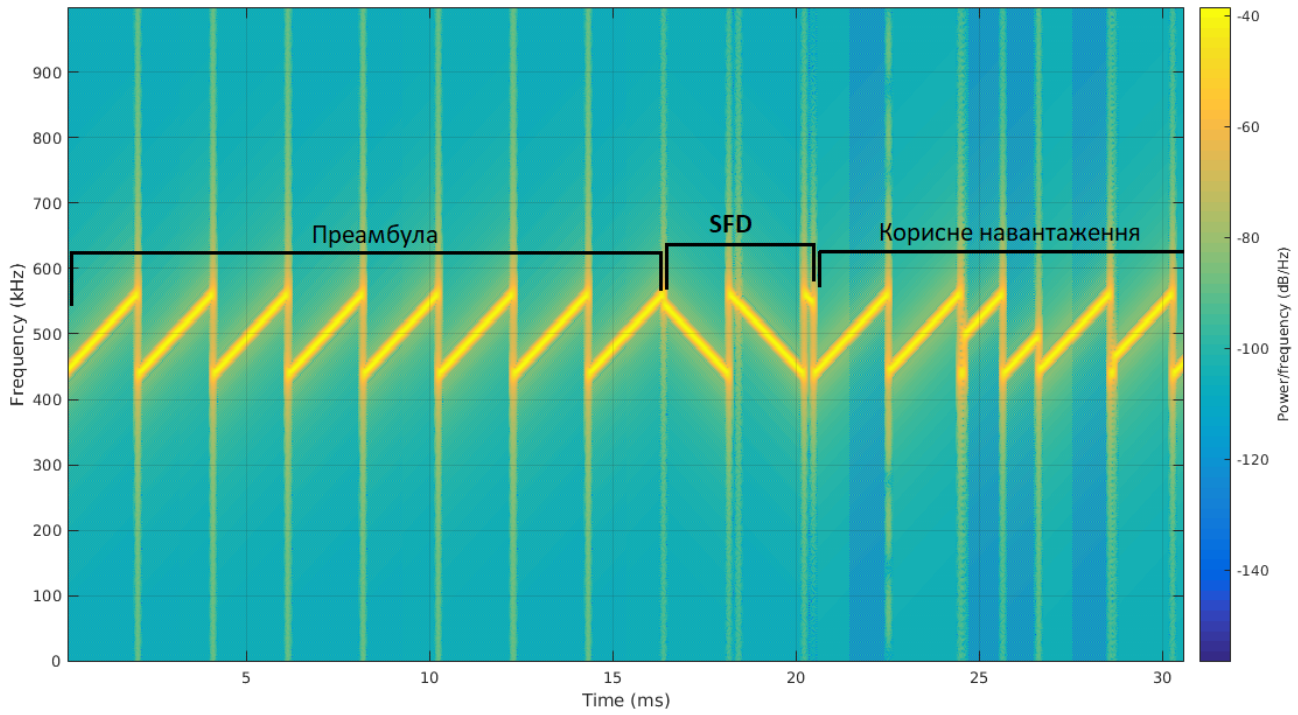

Рис. 1.4. Спектрограма структури LoRa повідомлення

Перша частина повідомлення складається з так званих "up-chirp" символів, які виконують роль преамбули. Кількість таких символів на початку повідомлення може варіюватись. Ця преамбула є початковим сигналом, який пристрій приймача використовує для визначення того, що повідомлення наближається. Символи преамбули дозволяють приймачу "прокинутися" і

приготуватися до прийому подальших частин повідомлення. Ці символb також допомагають відрізняти сигнал LoRa від інших завад і забезпечують початкове вирівнювання за частотою.

Після преамбули слідують два символи "down-chirp", які виконують функцію роздільника початкової частоти, тобто start frequency delimiter (SFD). Ці символи дають приймачу точну інформацію про момент початку передачі корисного навантаження і дозволяють точно налаштувати темп передачі повідомлення. Це є своєрідним етапом синхронізації і є критичним для встановлення з'єднання, бо саме це гарантує що приймач і передавач будуть працювати в одному часовому ритмі, що знижує ризик помилок і втрати даних при передачі.

І тільки після етапів преамбули та синхронізації в повідомленні йде основна частина, де міститься власне інформація, яку пристрій хоче передати. Ця частина складається з модульованих символів, кожен з яких несе в собі дані корисного навантаження. Корисне навантаження може містити показники сенсорів, інформацію про місце розташування, стан пристрою або інші важливі дані. Саме ця частина є основним «змістом» повідомлення, але її обсяг є досить малим через обмеження самої технології LoRa, яка призначена для передачі невеликих порцій даних на великі відстані.

Використання символів преамбули та синхронізації збільшує загальну тривалість передачі повідомлення. Хоча самі символи не містять корисної інформації і їхня присутність уповільнює процес передачі повідомлення, оскільки передача кожного символу займає певний час, вони є дуже важливими для забезпечення надійності зв'язку у жорстких умовах. У результаті, більша частина повідомлення LoRa складається не з корисних даних, а з контрольних символів, що і зумовлює зниження загальної швидкості передачі даних.

В результаті, така структура повідомлень виправдовує себе в жорстких умовах застосування, вона підвищує надійність зв'язку і забезпечує стійку передачу на великі відстані з мінімальним використанням енергії.

1.2. Аналіз використовуваних технологій для створення та забезпечення захищеної комунікації між LoRa пристроями

Кожен LoRa-пристрій може бути налаштований з унікальними параметрами, як-от частота, фактор поширення SF та потужність сигналу. Комбінація цих унікальних параметрів створює можливість персоналізації зв'язку для кожного конкретного випадку застосування. Така індивідуалізація конфігурацій дозволяє значно знизити ймовірність, що інші пристрої зможуть підключитися або перехопити дані без доступу до правильних параметрів зв'язку.

Так, наприклад, описаний в попередньому розділі фактор поширення може виступати в ролі своєрідного «коду», який дає змогу не тільки розпізнати сигнал LoRa, а й правильно його декодувати, зберігаючи цілісність повідомлень. Завдяки таким «кодам поширення» зловмисникам стає значно складніше ідентифікувати та зрозуміти сигнал, особливо без знання точного параметра SF, що створює певний рівень захисту на рівні радіочастот.

Через обмежену кількість можливих варіантів значень параметра фактору поширення і загальну доступність цих значень, зловмисник із відповідним набором обладнання може відносно легко здійснити атаку на таке з'єднання, використовуючи метод грубої сили, також відомий як bruteforce. Ця вразливість виникає через те, що параметр SF має фіксований набір значень, зазвичай від SF7 до SF12, що значно зменшує кількість значень, які потрібно перебрати для успішної атаки. Таким чином, зловмисник, який має доступ до приймально-передавального обладнання, може послідовно випробовувати кожне значення параметру SF, доки не знайде потрібне, що дозволить йому дешифрувати передані дані або втрутитися в обмін інформацією.

Наступним методом запобігання атак зміни та відтворення повідомлень є контроль доступу до радіоефіру. В технології LoRa це реалізовано набором обмежень і правил, які регулюють, як і коли пристрої можуть передавати дані, що

дозволяє уникнути перевантаження каналу. Зокрема, це стосується обмежень на частоту й тривалість передачі, завдяки чому декілька пристроїв можуть ділити один і той самий ефір без створення значних завад один для одного. LoRa використовує механізми обмеження, який називається duty cycle [8]. Цей механізм обмежує максимальну частку часу, коли передавач може працювати протягом певного періоду часу.

Так наприклад, пристрою може бути дозволено передавати сигнал лише 1% від загального часу, щоб не створювати перешкод для передачі іншим пристроям в мережі. Це дозволяє скоротити ризик перевантаження ефіру та забезпечити базову синхронізацію між пристроями, які одночасно користуються одним частотним діапазоном. Такі обмеження корисні в мережах з багатьма пристроями, які передають невеликі обсяги даних, адже вони запобігають одночасній передачі з кількох пристроїв. Крім того, дотримання duty cycle зменшує енергоспоживання пристроїв, що важливо для автономних сенсорів і модулів, які можуть працювати на малопотужних елементах живлення упродовж тривалого часу.

Хоча обмеження доступу до ефіру дають певні переваги для стабільності зв’язку, вони не є надійним засобом захисту від перехоплення повідомлень. Цей підхід має кілька серйозних вразливостей, які обмежують його ефективність у захисті даних.

Щоб обійти контроль доступу до ефіру, зловмисник може застосовувати приймач, налаштований на потрібну частоту, щоб постійно слухати ефір і записувати передані пакети, коли б вони не були відправлені. Встановлення обмеження duty cycle або інших параметрів не заважає таким діям, оскільки приймач пасивно реєструє всі сигнали в межах свого діапазону, не передаючи нічого у відповідь. Окрім того, зловмисник може використовувати пристрої, які не відповідають вимогам duty cycle, тобто які працюють без обмежень на передачу. Це може дозволити постійно перехоплювати та передавати пакети в мережі, порушуючи роботу законних пристроїв, які дотримуватися правил використання ефіру.

Ще одним вбудованим механізмом індивідуалізації даних що передаються є синхронізаційне слово (Sync Word) [10]. В технології LoRa це спеціальний набір бітів, які додаються до структури повідомлення для розпізнавання й синхронізації пристроїв LoRa. Цей параметр особливо корисний для відокремлення сигналів у LoRa-мережах, а також для уникнення можливих конфліктів із сусідніми мережами або пристроями, що працюють у тому самому частотному діапазоні.

Цей механізм працює по наступному принципу: на початку передачі повідомлення, в раніше вже згадану преамбулу, додається два байти інформації, яка показує, до якої мережі належить даний сигнал (рис. 1.5). Ці два байти виступають в ролі ідентифікатора мережі і можуть приймати будь-яке значення від 0x00 і до 0xFF, хоча по замовчуванню значення 0x12 вважається ідентифікатором для приватної мережі, а значення 0x34 для публічної, і є стандартним початковим значенням для будь-якого нового і неналаштованого LoRa модулю.

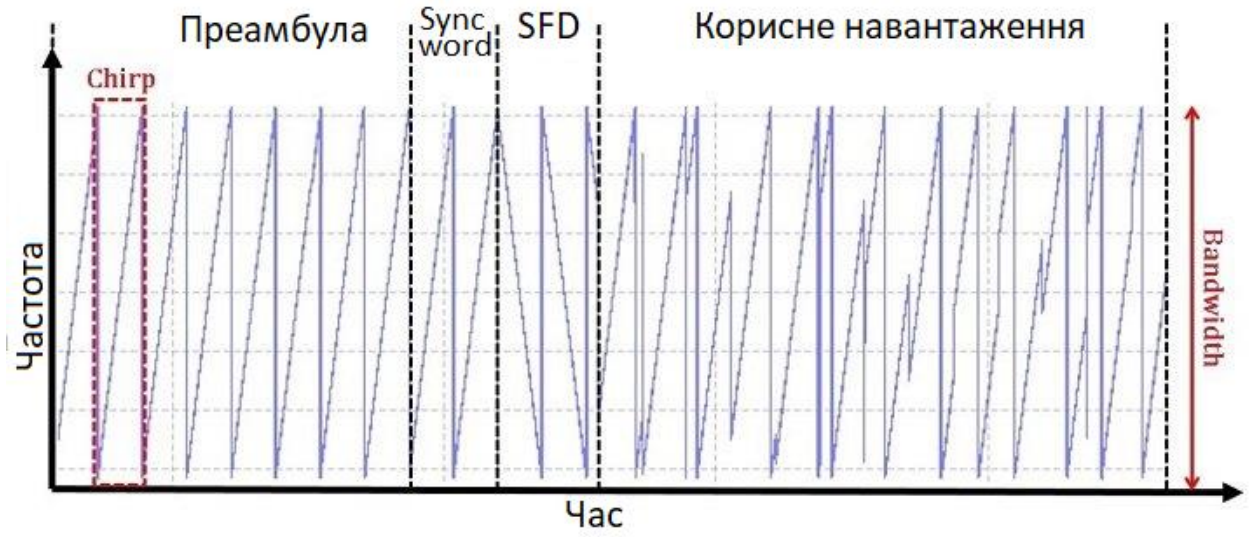

Рис. 1.5. Приклад структури LoRa повідомлення

Слід зауважити, що для успішної комунікації і надійного з’єднання, не лише тільки відправник має бути налаштованим для передачі сигналу із заздалегідь визначеним синхронізаційним словом, а й приймач теж має бути налаштований на очікування сигналу з таким самим Sync Word. Тому що, як вже згадувалось

раніше, преамбула виступає в ролі етапу підготовки для приймання сигналу, що йде слідом, і тепер, коли в неї додали ще два байти синхронізації, процес не може бути гарантовано успішним, якщо приймач не буде очікувати на такі саме байти. Таким чином, якщо приймач, наприклад, налаштований по замовчуванню на прийом публічної мережі із встановленим значенням Sync Word в 0x34, а до нього надходить сигнал з приватної мережі із значенням 0x12, то такий приймач просто не зможе синхронізуватись для подальшого правильного прийому повідомлення і просто отримає шум.

Sync Word забезпечує синхронізацію між передавачем і приймачем у межах мережі з однаковим значенням параметра Sync Word, а також допомагає уникнути прийому небажаних повідомлень, що надходять від інших LoRa-мереж, або пристроїв, які працюють у близьких частотних діапазонах. Завдяки цьому механізму повідомлення ігноруються, якщо вони мають неправильне значення параметру Sync Word, що мінімізує ймовірність обробки небажаних або помилкових даних.

Варто зазначити, що такий механізм хоча і є корисним інструментом для розмежування трафіку і уникнення помилкових спроб підключення до чужих мереж, але він не може бути використаний як надійний засіб захисту для передавання конфіденційних даних.

Основна проблема та недолік цього методу, як методу захисту, полягає в тому, що Sync Word - це простий ідентифікатор, а не засіб шифрування чи автентифікації. Саме повідомлення залишається незашифрованим, і будь-який приймач у зоні дії може бачити і приймати його. Це означає, що зловмисник, який має доступ до частотного діапазону LoRa, може проаналізувати отриманий сигнал, визначити значення Sync Word і використовувати його для синхронізації при отриманні наступного пакета даних, до якого він вже отримає доступ повністю.

Крім того, механізм Sync Word не забезпечує захист від підміни або модифікації даних, оскільки не можу гарантувати автентичності джерела. Тож без

застосування додаткового шифрування або іншого методу захисту та перевірки автентичності будь-який пакет із правильним Sync Word може бути прийнятий як легітимний, навіть якщо його походження невідоме чи зловмисне. Таким чином, Sync Word залишається корисним лише для базової синхронізації пристроїв, які працюють в одному частотному діапазоні.

Як вже було зазначено, LoRa є технологією, яка працює виключно на фізичному рівні зв'язку в рамках моделі OSI (рис. 1.6), і її основне завдання полягає в забезпеченні стабільної передачі та прийомі даних на певній частоті в умовах обмеженої пропускної здатності.

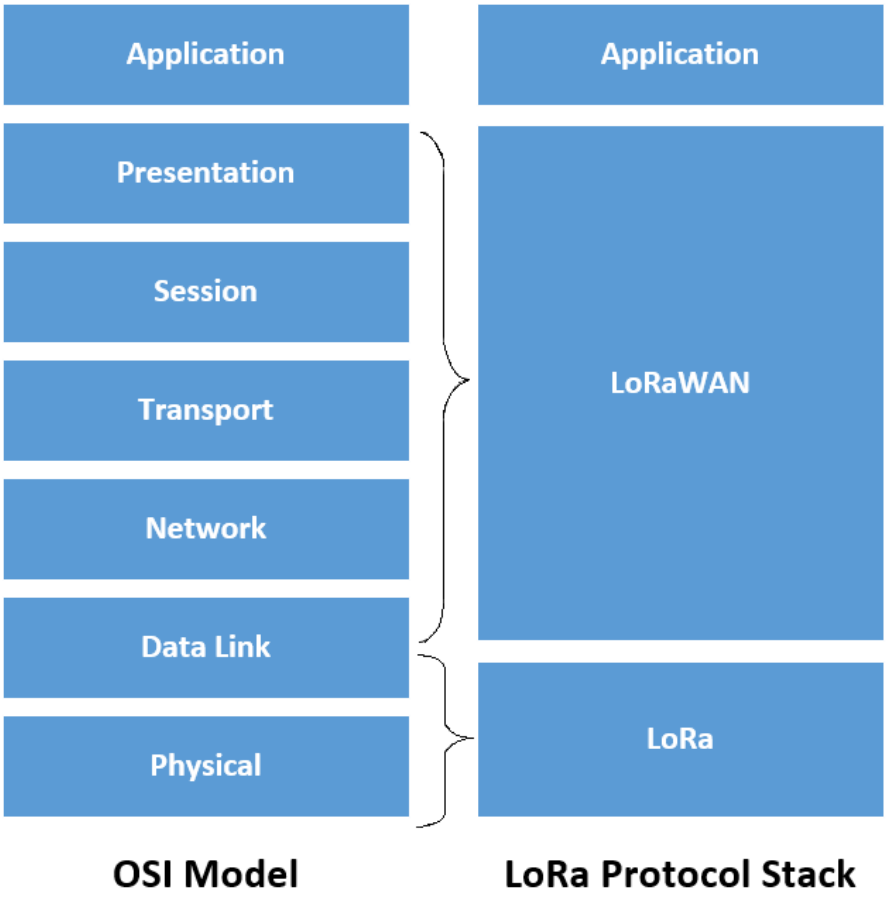

Рис. 1.6. LoRa та LoRaWAN в моделі OSI

LoRa визначає те як передаються сигнали, контролює їхню модуляцію, методи розширення спектру, а також обробляє сигнали на рівні, який дає змогу з'єднати два пристрої на основі радіочастотного обміну даними. Важливо

розуміти, що ця технологія створена для надійної роботи в умовах низької потужності, великого радіуса дії і роботи з мінімальним енергоспоживанням, тож через таку вузьку спеціалізацію LoRa обмежується лише фізичним рівнем і сама по собі не включає функції шифрування комунікації між пристроями чи їх автентифікації. Це означає, що технологія не може самостійно забезпечити захист даних або гарантувати, що до мережі підключаються лише легітимні пристрої. Для LoRa пріоритетом є способи передачі та обробки сигналів, те як вони створюються, модулюються та передаються у радіоефір.

Відсутність механізмів шифрування трафіку і автентифікації пристроїв в LoRa обумовлена тим, що фізичний рівень, на якому вона працює, не відповідає за безпеку передачі даних. Іншими словами, LoRa розроблена більше для створення та підтримки зв’язку на рівні сигналів, аніж для забезпечення їх конфіденційності та захищеності.

Для побудови більш складних мережевих структур та для досягнення більш високого рівня безпеки під час передавання даних на базі технології LoRa було створено протокол LoRa Wide Area Network (LoRaWAN) [11]. На відміну від базового LoRa, який забезпечує тільки фізичний рівень передачі, відповідаючи за модуляцію, обробку та передачу сигналу, LoRaWAN працює на рівні управління доступом до середовища, тобто на рівні Medium Access Control (MAC), і є надбудовою для технології LoRa (рис. 1.7). Це означає, що LoRaWAN виконує додаткові функції, такі як контроль доступу до мережі, управління маршрутизацією пакетів даних і забезпечення автентифікації пристроїв.

LoRaWAN створена спеціально для середовищ, де потрібна не тільки дальність і енергоефективність LoRa, а й більш комплексна обробка даних. Таким чином, LoRaWAN є мережевою технологією, орієнтованою на Інтернет речей (IoT), що дозволяє підключати численні пристрої до однієї мережі, забезпечуючи безпеку для їх комунікації. Це досягається шляхом об’єднання декількох важливих елементів.

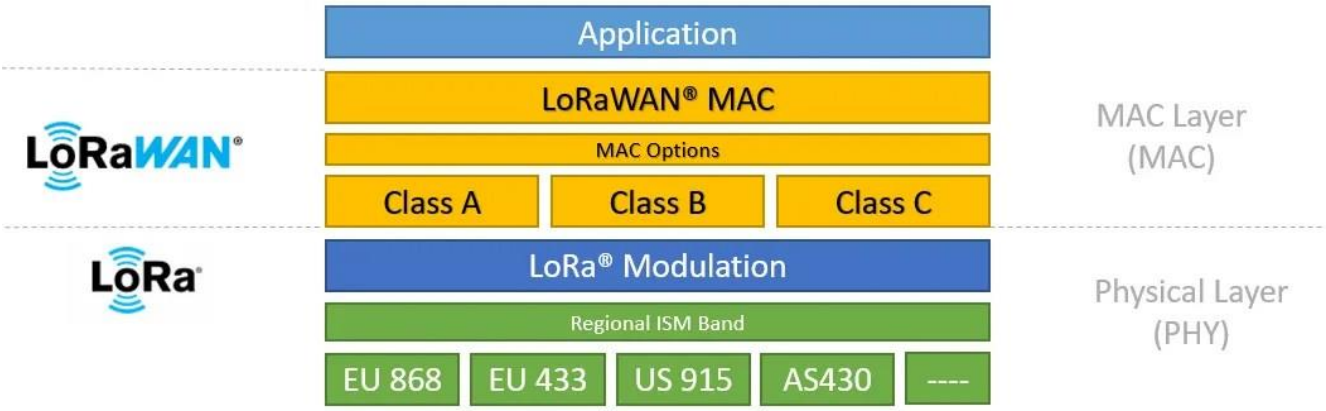

Рис. 1.7. LoRaWAN - протокол MAC рівня

Першим ключовим елементом системи LoRaWAN є її комплексна та багатошарова архітектура (рис. 1.8), яка дозволяє забезпечити надійний та поетапний зв'язок між численними пристроями та користувачами. LoRaWAN складається не лише з кінцевих пристроїв, таких як датчики або сенсори, та користувачів, які отримують дані з цих пристроїв, але включає також кілька важливих проміжних компонентів, які забезпечують стабільне функціонування всієї мережі. До таких компонентів належать шлюзи (gateways), мережеві сервери (Network servers) і сервера додатків (Application servers).

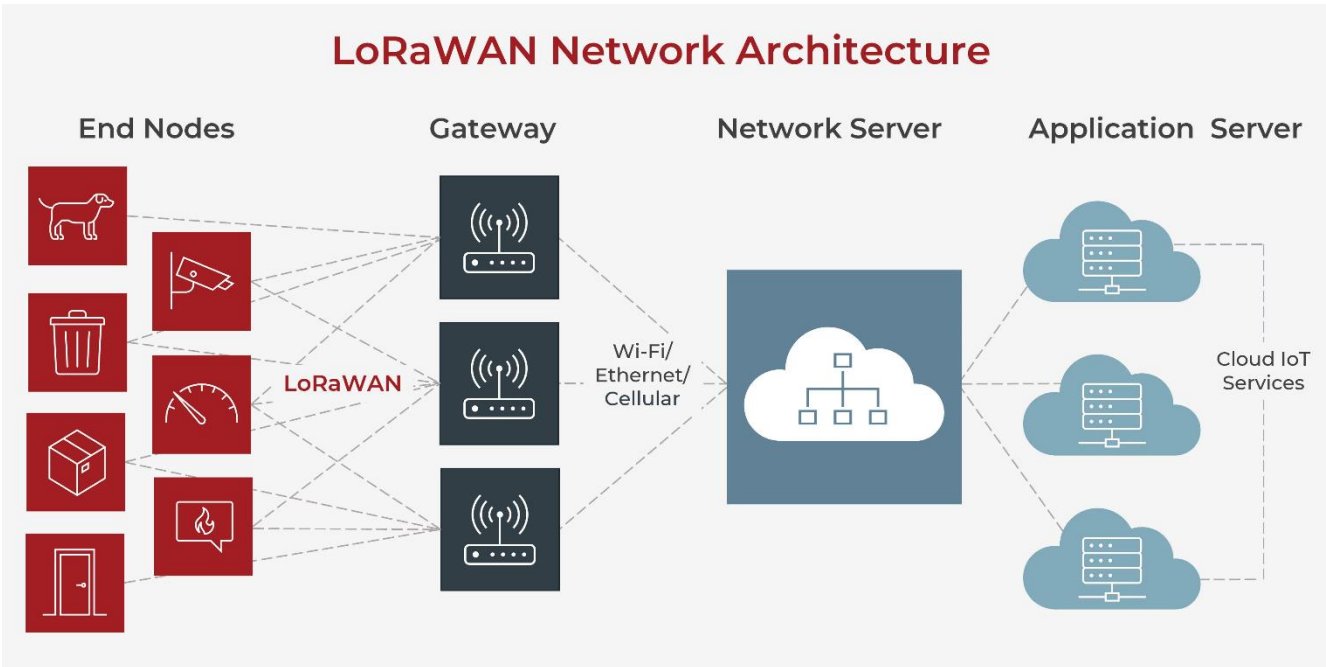

Рис. 1.8. Приклад архітектуру мережі LoRaWAN

В такій складній та багатошаровій архітектурі з великою кількістю кінцевих пристроїв, кожен проміжний компонент виконує функцію фільтрації і перетворення отриманих даних в більш зрозумілі для людини значення, при цьому зберігаючи цілісність так конфіденційність цих даних. Так до прикладу шлюзи виконують роль мостів між кінцевими пристроями LoRa та серверною інфраструктурою LoRaWAN. Їхнє основне завдання полягає в тому, щоб приймати сигнали від численних пристроїв, які знаходяться у межах їхнього радіусу дії, і передавати ці дані далі на мережевий сервер через звичайне підключення до інтернету, наприклад, за допомогою Ethernet, Wi-Fi або мобільної мережі. Шлюзи не здійснюють обробку або дешифрування даних, натомість вони просто приймають та передають інформацію, що забезпечує відносно просту структуру пристроїв і високу пропускну здатність для обробки великої кількості з’єднань. Завдяки цьому шлюзи можуть обслуговувати одночасно тисячі пристроїв, що робить їх надзвичайно ефективними для масштабних IoT-мереж.

Мережевий сервер, який виступає в ролі основного центру управління, є ще одним важливим компонентом мережі LoRaWAN. Він виконує кілька важливих функцій, включаючи обробку та маршрутизацію даних, контроль доступу до мережі, управління безпекою, автентифікацію пристроїв, а також оптимізацію використання радіочастот. Мережевий сервер також відповідає за обробку колізій між сигналами від різних пристроїв і зменшення завад у мережі, що особливо важливо у середовищах з великою кількістю активних пристроїв. Це дозволяє підтримувати стабільність і продуктивність LoRaWAN навіть у складних умовах з високою інтенсивністю трафіку.

Сервер додатків є заключним компонентом мережі LoRaWAN. Його завдання полягає у прийомі вже оброблених та відфільтрованих даних від мережевого сервера, які потім розшифровуються і передаються на інтерфейси для взаємодії з системою, або на кінцеві застосунки, які в свою чергу вже використовуються користувачами. Саме на рівні сервера додатків здійснюється інтерпретація даних з кінцевих пристроїв, наприклад, значень температури,

показників вологості, даних про розташування тощо. Цей компонент надає інтерфейси та API для взаємодії з іншими системами, а також може зберігати, аналізувати й візуалізувати зібрані дані.

Завдяки такій багатокомпонентній структурі LoRaWAN може підтримувати надійне підключення для великої кількості пристроїв у масштабних IoT-мережах. Але окрім простого надання підключень для великої кількості пристроїв, LoRaWAN також реалізує безпеку цим підключенням, що є другим головним елементом цієї системи.

Ключовою перевагою LoRaWAN над LoRa є забезпечення захищеності переданих даних в умовах використання загальнодоступного радіоефіру. Оскільки LoRa працює на ліцензійно вільних частотах, дані, що передаються, потенційно можуть бути перехоплені зловмисниками або сторонніми пристроями, що знаходяться в радіусі дії сигналу. LoRaWAN вирішує цю проблему завдяки системі з декількома шарами шифрування, яка розподілена поміж всіма описаними вище елементами мережі. Це досягається за рахунок використання двох унікальних ключі: один для захисту користувацьких даних (Application Session Key), і другий для безпечного обміну мережевими командами (Network Session Key). Тож навіть якщо неавторизований користувач зможе зчитати передачу, без необхідних ключів він не зможе розшифрувати її та отримати доступ до конфіденційної інформації, що робить LoRaWAN значно надійнішим рішенням для захищеної комунікації, ніж звичайна LoRa, яка не має засобів захисту даних.

Кожен кінцевий пристрій отримує ці два унікальних ключі по завершенню процесу активації [12]. В протоколі LoRaWAN існує два можливих варіанти активації: Over-The-Air-Activation (OTAA), який рахується більш надійним і динамічним; та Activation By Personalization (ABP), який є простішим, але натомість менш захищеним. Розглянемо ці два методи по порядку.

При описанні методу активації «через повітря» (Over-The-Air-Activation), слід зауважити, що на момент написання цієї роботи існує дві версії цього

протоколу: «Over The Air Activation in LoRaWAN 1.0.x», який був розроблений першим, та «Over The Air Activation in LoRaWAN 1.1», який є доповненням першої версії протоколу і має функціонал роботи з новим компонентом мережі під назвою сервер приєднання (Join Server), доданим для забезпечення більшої надійності процесу активації пристроїв в мережі, та легшої централізованої імплементації даного протоколу.

У протоколі LoRaWAN 1.0.x процедура приєднання вимагає обміну двома повідомленнями між кінцевим пристроєм і мережевим сервером. Цими повідомленнями є запит на приєднання від кінцевого пристрою до мережевого серверу (Join-request), та повідомлення прийняття приєднання (Join-accept) від мережевого серверу до кінцевої точки.

Перед початком процесу активації слід переконатись в наявності та унікальності таких ключів як AppEUI, DevEUI та AppKey на кінцевому пристрої. AppKey - це секретний ключ AES-128, відомий як кореневий ключ (root key). Для можливості проведення процесу активації такий самий AppKey має бути збережений в тій мережі, до якої кінцевий пристрій намагається приєднатись. Ключі AppEUI і DevEUI не є секретними і доступні для всіх. Слід зауважити, що ключ AppKey ніколи не надсилається по мережі і має зберігатись на кінцевому пристрою та мережевому сервері.

Першим кроком процесу приєднання пристрою до мереж, який завжди ініціюється пристроєм, є надсилання повідомлення із запитом на приєднання. Поля цього повідомлення та розміри кожного з полів наведені в таблиці 1.1.

Таблиця 1.1.

Структура повідомлення «Join-request»

AppEUI	DevEUI	DevNonce	MIC
8 байт	8 байт	2 байти	4 байти

Першим поле в повідомленні Join-request йде поле AppEUI, яке є 64-бітним глобальним унікальним ідентифікатором програми в адресному просторі IEEE EUI64, який ідентифікує об’єкт, здатний обробити Join-request. Наступним йде

DevEUI – 64-розрядний глобальний унікальний ідентифікатор пристрою в адресному просторі IEEE EUI64, який унікально ідентифікує кінцевий пристрій. І останнє поле це в Join-request – DevNonce. Це поле є унікальним, випадковим, 2-байтовим значенням, згенерованим кінцевим пристроєм. Мережевий сервер використовує DevNonce кожного кінцевого пристрою, щоб відстежувати їхні запити на приєднання. Якщо кінцевий пристрій надсилає запит на приєднання з попередньо використаним значенням поля DevNonce, ця ситуація відома як атака повтору, мережевий сервер відхиляє запит на приєднання та не дозволяє кінцевому пристрою зареєструватися в мережі. Як підтвердження того, що повідомлення було надіслано без змін зі сторони, для всіх полів у повідомленні запиту на приєднання за допомогою секретного ключа AppKey, який зберігається на пристрої, обчислюється код цілісності повідомлення (MIC). Розрахований MIC також додається до повідомлення Join-request.

Тут слід зазначити, що значення AppKey не надсилається разом із повідомленням в запиті на приєднання, а саме повідомлення надсилається в незашифрованому вигляді. Повідомлення запиту на приєднання може бути передане з використанням будь-якої швидкості передачі даних і з використанням одного з регіональних каналів приєднання, тож на мережевий сервер воно проходить через один або кілька шлюзів.

Після того як мережевий сервер прийняв повідомлення Join-request від пристрою, відбувається другий крок протоколу приєднання. Сервер робить процес перевірки повідомлення, і уразі у разі його легітимності, генерує два сесійних ключі AppSKey та NwkSKey. Після цього сервер відправляє пристрою повідомлення Join-accept, щоб сповістити пристрій про успішне приєднання до мережі. Поля повідомлення Join-accept та їх розміри наведені в таблиці 1.2.

Таблиця 1.2.

Структура повідомлення «Join-accept»

AppNonce	NetID	DevAddr	DLSettings	RXDelay	CFList	MIC
3 байти	3 байти	4 байти	1 байт	1 байт	16 байт	4 байти

Поле AppNonce виступає в ролі випадкового числа, або унікального ідентифікаційного коду, назначеного мережевим сервером пристрою. AppNonce використовується кінцевим пристроєм, щоб розрахувати такі самі сесійні ключі AppSKey та NwkSKey, які були розраховані на сервері.

Наступні два поля NetID та DevAddr виступають в ролі ідентифікаторів самого пристрою. Для легшої організації керування пристроями, кожному з них призначається 24-бітний номер мережі (NetID), до якої вони відносяться, а вже для безпосереднього керування самим пристроєм в середині конкретної мережі, кожному пристрою надається 32-бітне значення адреси девайсу (DevAddr), яке розраховується та назначається мережевим сервером.

Після цього йде поле розміром в один байт під назвою DLSettings. Це поле містить налаштування для приймального вікна, що пристрій має використовувати для прийому даних. Тут зберігаються параметри, які визначають, зокрема, вибір каналу для другого приймального вікна (RX2) та фактор розширення (SF) для цього каналу. Завдяки DLSettings пристрій дізнається, як налаштувати свої приймальні параметри для оптимальної роботи в умовах конкретної мережі. Використання цього поля забезпечує гнучкість у конфігурації приймальних вікон та дозволяє мережі враховувати фактори, такі як навантаження на канали та географічні особливості. Це особливо важливо для зниження ризику колізій сигналів і покращення якості з’єднання у розгалужених мережах з багатьма пристроями.

Наступне поле під назвою RxDelay, яке визначає затримку між моментом передачі даних пристроєм і моментом відкриття приймального вікна, відповідальне за встановлення часового інтервалу, через який пристрій відкриє своє приймальне вікно після завершення передачі даних. Це налаштування є критично важливим для координації зв’язку, оскільки з неправильним вибором цього інтервалу можуть виникнути збої в обміні даними. Наприклад, надто короткий інтервал може призвести до пропуску сигналу від шлюзу, тоді як надто довгий може призвести до затримки у сприйнятті інформації. RxDelay дозволяє

оптимально налаштувати ці інтервали залежно від таких особливостей мережі, як фізична віддаленість між пристроєм і шлюзом або наявності інтерференцій. Це налаштування покращує стабільність роботи системи, дозволяючи пристроям синхронізувати приймальні вікна з очікуваними відповідями від мережі.

Передостаннім є додаткове поле CFList. Це поле, що містить список частотних каналів, які підтримуються мережею, до якої приєднується кінцевий пристрій. Це поле є необов'язкове і зазвичай застосовується у випадках, коли мережа хоче запропонувати додаткові канали для передачі даних, окрім стандартних, які за замовчуванням підтримуються пристроєм. CFList дає мережевому серверу можливість надати кінцевому пристрою інформацію про додаткові частоти, що підходять для передачі даних у конкретному регіоні. Оскільки частотні плани можуть відрізнятися залежно від регіону розташування, CFList дозволяє пристрою динамічно адаптувати свої налаштування до частот, специфічних для його географічного місцезнаходження. Це підвищує ефективність використання радіочастотного ресурсу та забезпечує відповідність пристрою до місцевих вимог і стандартів частотного діапазону.

І останнім полем в повідомленні про успішність приєднання до мережі йде MIC. Як і у випадку з Join-request, за допомогою секретного ключа AppKey, що зберігається на сервері, для кожного поля в повідомленні Join-accept обчислюється код цілісності, який підтверджує правдивість отриманого повідомлення. Таким чином, після того як пристрій отримає це повідомлення, за допомогою свого AppKey він зможе перевірити його цілісність.

На четвертому кроці, після того як кінцевий пристрій успішно отримує повідомлення Join-accept, мережевий сервер виконує розподіл згенерованих ключів. На цьому етапі ключі сесії розділяються таким чином, щоб гарантувати безпеку даних на різних рівнях взаємодії. Мережевий сервер зберігає Network Session Key (NwkSKey), тоді як Application Session Key (AppSKey) передається на сервер додатків. Це розділення має критичне значення, оскільки NwkSKey та AppSKey мають різні функції.

Ключ NwkSKey, залишаючись на мережевому сервері, відповідає за підтримку цілісності службових повідомлень. За допомогою цього ключа мережевий сервер може перевіряти автентичність кожного повідомлення, що надходить від пристрою, завдяки розрахунку коду цілісності повідомлення (MIC). MIC генерується на основі NwkSKey і дозволяє виявляти будь-які несанкціоновані зміни у повідомленнях, що забезпечує захист від підробок або змін даних в ході їх передачі. AppSKey, в свою чергу, надсилається на сервер додатків, де він використовується для захисту корисного навантаження даних. Це означає, що будь-які дані, які передає кінцевий пристрій, шифруються на основі AppSKey, і лише сервер додатків, який має цей ключ, може розшифрувати інформацію. Завдяки цьому підходу, дані користувача та показники датчиків залишаються конфіденційними, і навіть мережевий сервер не може переглядати чи змінювати їх зміст.

На останньому, п’ятому етапі, кінцевий пристрій здійснює розшифрування отриманого повідомлення Join-accept за допомогою операції шифрування AES-128. Цей етап є ключовим для забезпечення безпеки з'єднання, оскільки на ньому здійснюється генерація двох основних сесійних ключів, які забезпечують як автентифікацію, так і конфіденційність переданих даних. Під час розшифрування Join-accept повідомлення кінцевий пристрій використовує AppKey — ключ, який був заздалегідь згенерований і закодований під час виробництва або попередньої конфігурації пристрою. У поєднанні з унікальним ідентифікатором сеансу AppNonce, який генерується мережею, цей ключ використовується для створення двох таких самих сесійних ключів NwkSKey і AppSKey, які зберігаються на мережевому сервері і сервері додатків відповідно.

Як було вже описано в четвертому кроці, NwkSKey використовується для обміну контрольними даними між кінцевим пристроєм та мережевим сервером, забезпечуючи цілісність усіх повідомлень, що надходять від кінцевого пристрою в мережу. В той час як AppSKey має інше призначення і використовується для забезпечення конфіденційності переданих даних. Завдяки AppSKey здійснюється

шифрування та дешифрування корисного навантаження у всіх пакетах, що надсилаються від пристрою на сервер додатків або навпаки.

Після створення цих сесійних ключів кінцевий пристрій офіційно активується в мережі, отримуючи повний доступ до неї, і тепер може передавати та отримувати дані в захищеному режимі. Це дозволяє уникнути несанкціонованого доступу до переданих даних і захищає мережу від потенційних загроз, забезпечуючи стабільне, безпечне і надійне функціонування пристрою у складі IoT-інфраструктури.

На діаграмі 1.9 зображені кроки, що наглядно демонструють описаний процес активації по повітрю (OTAA).

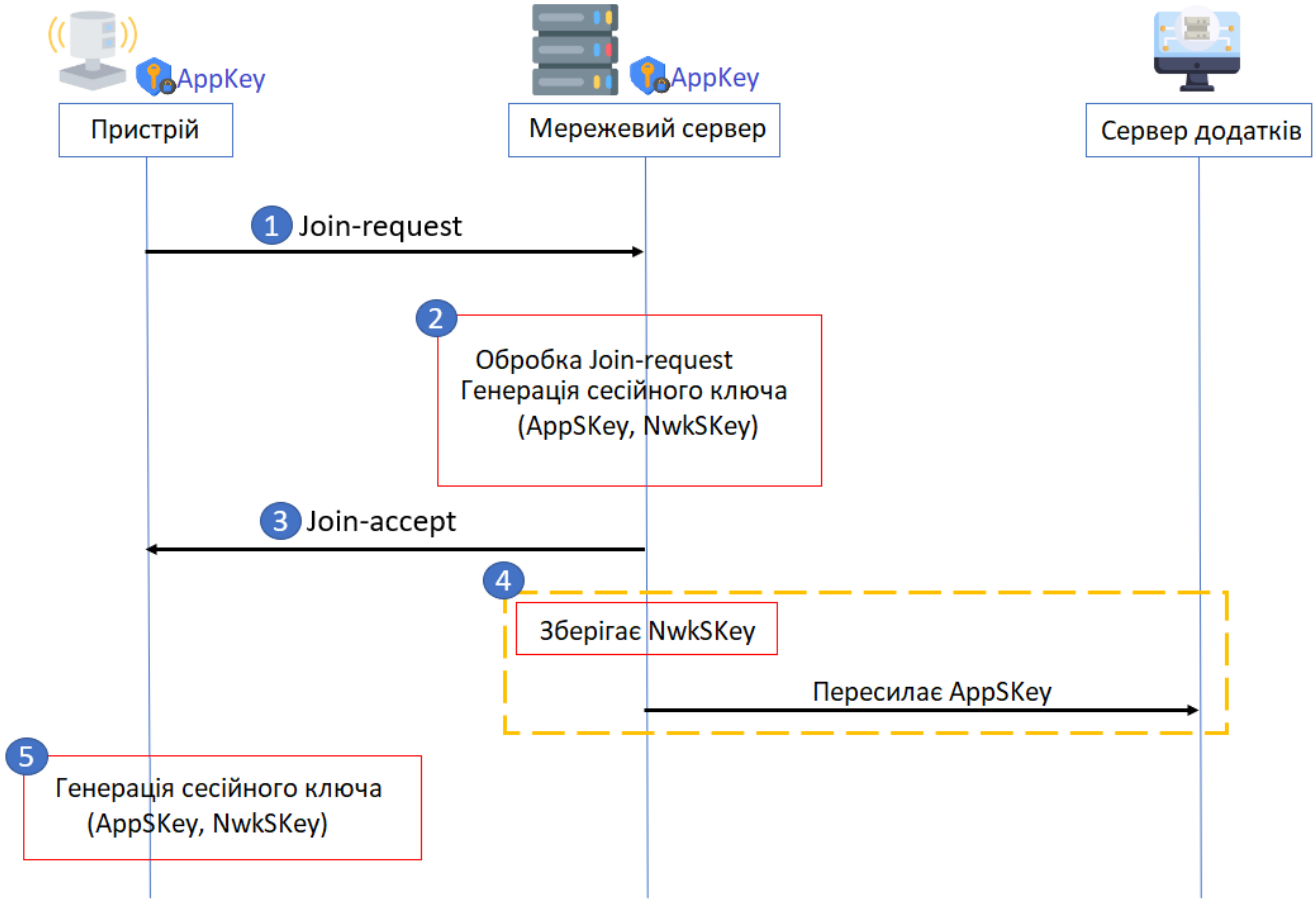

Рис. 1.9. Кроки протоколу «Over The Air Activation» в LoRaWAN 1.0.x

Варто зазначити, що в протоколі «Over The Air Activation» (OTAA) версії LoRaWAN 1.1 є кілька важливих відмінностей порівняно з LoRaWAN 1.0.x, які націлені на підвищення безпеки та стабільності зв’язку. Ці зміни були

впроваджені через зростаючі вимоги до безпеки IoT-мереж і для розширення можливостей роботи з різними сценаріями використання.

По-перше, на відміну від LoRaWAN 1.0.x, де використовувався лише один попередньо встановлений ключ AppKey, в LoRaWAN 1.1 було вирішено використовувати два передвстановлених ключі AppKey і NwkKey. Ці ключі є 128-бітними AES ключами і мають зберігатись як на кінцевому девайсі, так і в тій мережі, до якої приєднується девайс. Це розділення дозволяє краще захистити процес автентифікації та обміну ключами, оскільки кожен з цих ключів виконує окремі функції в механізмі безпеки мережі. Також варто зазначити, що на першому кроці, коли пристрій надсилає Join-request мережевому серверу, останнє поле запиту під назвою MIC, яке відповідає за збереження цілісності всього повідомлення, генерується за допомогою ключа NwkKey, на відміну від попередньої версії протоколу, де це відбувалось за допомогою AppKey ключа. Окрім того, при відправлені запиту на приєднання, в протоколі версії LoRaWAN 1.1, назва поля AppEUI була замінена на JoinEUI, але при цьому розмір та ціль використання поля залишились ті самі.

Однією з найбільших змін у LoRaWAN 1.1 є введення нового компонента мережі під назвою сервер приєднання (Join Server), який тепер виконує основну роль у генерації та обміні ключами безпеки при приєднанні нового пристрою до мережі. У LoRaWAN 1.0.x функції серверу приєднання були інтегровані в мережевий сервер, проте в новій версії протоколу він функціонує незалежно, що дозволяє мережам із високими вимогами до безпеки більш гнучко управляти ключами без втручання в роботу мережевого сервера. Тож тепер, на першому кроці, коли на мережевий сервер приходить запит на приєднанні до мережі, то він просто перенаправляє цей запит від пристрою на сервер приєднання, який вже обробляє цей запит і на другому кроці генерує вже не два сесійних ключі, а чотири: AppSKey, FNwkSIntKey, SNwkSIntKey, та NwkSEncKey.

Якщо ця генерація сесійних ключів пройшла успішно, то сервер приєднання відправляє повідомлення «Success OK» на мережевий сервер, який в свою чергу

вже відправляє повідомлення Join-accept на пристрій, який очікує дозволу на приєднання. Тут слід зауважити, що в порівнянні з протоколом в LoRaWAN 1.0.x, в новій версії протоколу зміст повідомлення Join-accept відрізняється. В LoRaWAN 1.1 поле AppNonce було замінене на JoinNonce, та його розмір зменшений до одного байта. Кінцева структура повідомлення про підтвердження приєднання до мережі наведена в таблиці 1.3.

Таблиця 1.3

Структура повідомлення «Join-accept» в LoRaWAN 1.1

JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList	MIC
1 байт	3 байти	4 байти	1 байт	1 байт	16 байт	4 байти

По завершенню відправки повідомлення «Success OK», на четвертому кроці, сервер приєднання розподіляє згенеровані сесійні ключі між мережевим сервером та сервером додатків, відправляючи NwkSKey на перший та AppSKey на другий відповідно, що забезпечує кращий контроль за доступом і підвищує рівень безпеки.

На останньому, п’ятому етапі, коли пристрій отримує Join-accept повідомлення від мережевого серверу, за допомогою значення з поля JoinNonce він генерую такі самі чотири сесійних ключі (FNwkSIntKey, SNwkSIntKey, NwkSEncKey та AppSKey), що зберігаються в мережі, після чого він стає активованим і готовим до захищеної та шифрованої комунікації з мережею.

Кожен із цих ключів має своє призначення, що дозволяє більш чітко розмежувати функції кожного з них та зменшити ризики компрометації всього з’єднання. Так, наприклад, ключ FNwkSIntKey використовується для обчислення частини коду цілісності повідомлення (MIC) тільки для повідомлень від кінцевого пристрою до мережі. Це робить його корисним для сервісів або компонентів мережі, які можуть обробляти uplink-повідомлення, але не мають повного доступу до інших аспектів комунікації. Така сегментація допомагає розділити функції безпеки та забезпечує, що деякі сервери мають доступ лише до окремих частин

інформації. Наприклад, перенаправляючий мережевий сервер може бути обмежений у доступі і не матиме повних прав на розшифровку або обробку даних.

В той час як SNwkSIntKey, навпаки, є більш універсальним ключем. Він використовується для обчислення MIC як uplink-повідомлень разом ключем з FNwkSIntKey, так і для всіх downlink-повідомлень від мережі до кінцевого пристрою. Це означає, що SNwkSIntKey використовується для перевірки цілісності повідомлень в обох напрямах, що забезпечує надійність та безпечність передачі на рівні основного мережевого серверу.

Так само як і у випадку з SNwkSIntKey, ключ NwkSEncKey використовується для шифрування та дешифрування у всіх uplink і downlink повідомленнях, але вже корисного навантаження, а не коде цілісності, як це робить SNwkSIntKey. Цей ключ забезпечує конфіденційність MAC-команд, які можуть містити важливу інформацію про конфігурацію пристрою, параметри зв’язку або інші налаштування, що потребують захисту. Використання NwkSEncKey дозволяє захистити ці команди від несанкціонованого доступу, що запобігає спробам маніпуляції або спотворення критично важливих налаштувань пристрою.

Останній з чотирьох ключів – AppSKey. Як і в протоколі версії LoRaWAN 1.0.x, він забезпечує конфіденційність даних, що передаються між кінцевим пристроєм і сервером застосунків. На відміну від інших ключів, що забезпечують цілісність та захист від підробок на рівні мереж, ключ AppSKey застосовується виключно для шифрування й дешифрування даних, які передаються на рівні застосунку. Таким чином, ці ключі, забезпечують високий рівень безпеки у мережах, що використовують протокол «Over the Air Activation» версії LoRaWAN 1.1, виконуючи різні функції з захисту цілісності та конфіденційності даних. Завдяки їх розподіленому використанню, LoRaWAN 1.1 підвищує загальну стійкість мережі до атак, забезпечуючи захист не тільки даних, що передаються, а й керуючих команд.

Покроковий процес активації по повітрю у версії протоколу LoRaWAN 1.1 показані на рисунку 1.10.

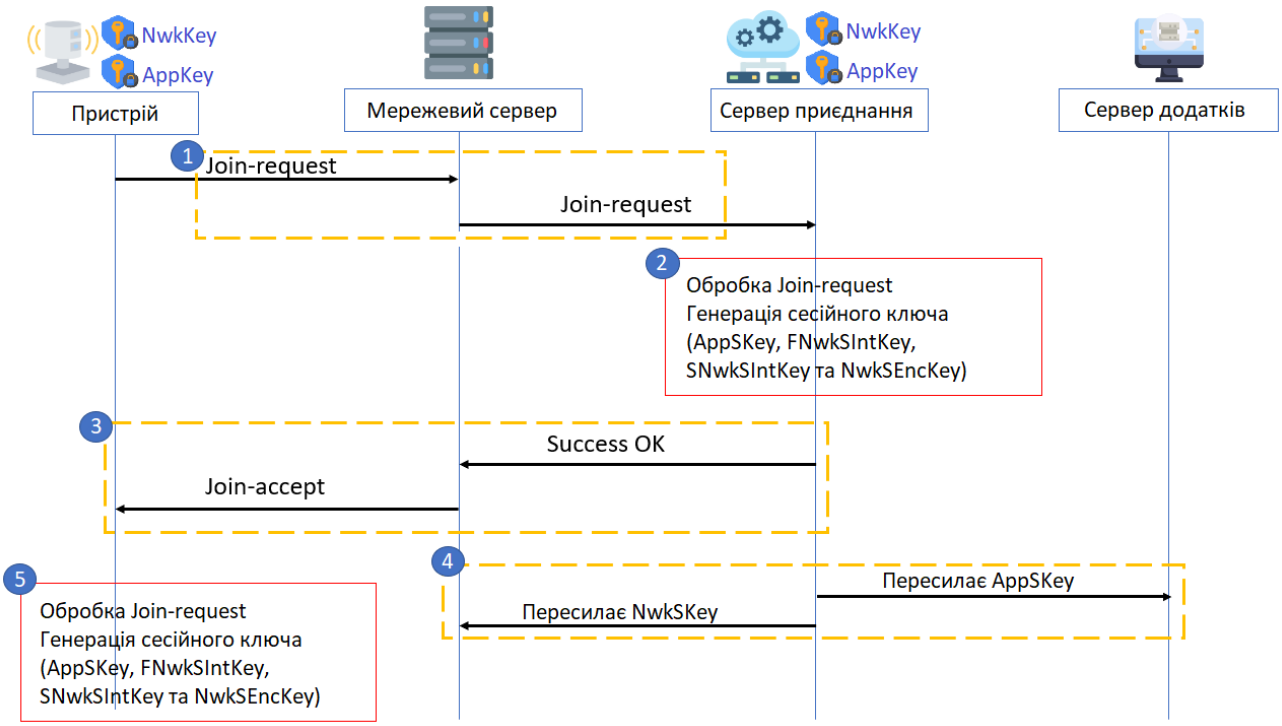

Рис. 1.10. Кроки протоколу «Over The Air Activation» в LoRaWAN 1.1

Одним із ключових досягнень LoRaWAN 1.1 є підвищення захисту від атак. Розподіл ключів між різними рівнями мережі унеможливлює атаки типу грубої сили, коли використовується простий підбір одного ключа для отримання доступу до всієї інформації. Крім того, кожен ключ відповідає лише за конкретний аспект зв'язку, що ускладнює атаки на цілісність повідомлень або втручання у внутрішній потік даних. Усе це робить LoRaWAN 1.1 більш безпечним протоколом, ніж його попередні версії, і дозволяє використовувати його в більш широкому спектрі додатків, де критично важливими є питання безпеки та захисту даних.

Також, як було згадано раніше, існує другий протокол активації пристрою в мережі під назвою «Активація за допомогою персоналізації» (Activation By Personalization), що є простішим у використанні, але й менш захищеним і менш динамічним, порівняно з протоколом «Over The Air Activation». У межах

протоколу активації за допомогою ABP всі необхідні параметри та ключі для шифрування даних і забезпечення цілісності повідомлень мають бути встановлені вручну ще до запуску пристрою. Ці параметри є ідентичними по функціоналу до вже розглянутих параметрів в протоколі «Over The Air Activation», таких як адреса девайсу та сесійні ключі, які зберігаються в пам’яті пристрою та є постійними протягом усього часу його роботи в мережі.

Оскільки активація в протоколі ABP не вимагає обміну повідомленнями з мережевим сервером для встановлення сеансових ключів, цей метод підходить для сценаріїв, де необхідно уникнути додаткових операцій приєднання, які можуть споживати енергію або викликати затримки в обміні даними. Це є важливою перевагою для пристроїв, що працюють у важкодоступних або обмежених щодо енергії середовищах, де пристрій може перебувати в режимі сну і прокидатися лише для періодичної передачі даних, оскільки такий підхід мінімізує обсяги обміну даними і знижує загальні енергетичні витрати.

Водночас ABP має і певні недоліки, пов’язані з безпекою та гнучкістю використання. Оскільки сесійні ключі, що відповідають за шифрування та перевірку повідомлень, мають бути попередньо встановленими та не змінюються в процесі роботи пристрою, це створює вразливість для можливих атак, оскільки скомпрометовані ключі дозволять зловмиснику перехоплювати та змінювати дані без необхідності повторної активації або оновлення ключів. На відміну від OTAA, де ключі генеруються динамічно під час кожної активації і мають обмежений термін дії, ABP не підтримує оновлення ключів під час експлуатації пристрою. Це означає, що для зміни ключів необхідно вручну перепрограмувати пристрій, що може бути незручно або навіть неможливо в певних умовах.

Крім того, ABP не підтримує механізм динамічного призначення адрес, як це відбувається в OTAA. Усі пристрої, активовані за допомогою ABP, мають постійні адреси DevAddr, які задаються вручну. Це обмежує гнучкість мережі та може призвести до конфліктів адрес, особливо в великих мережах із значною кількістю пристроїв, де існує ймовірність випадкового дублювання адрес. Для

уникнення таких конфліктів необхідно ретельно планувати розподіл адрес та використовувати лише унікальні DevAddr.

Таким чином, ABP є простим і енергоефективним методом активації, але він має певні обмеження, що робить його менш безпечним і гнучким у порівнянні з OTAA. ABP залишається популярним серед рішень, де важлива стабільність і тривалий час автономної роботи пристрою, а питання безпеки або необхідність зміни ключів є менш критичними.

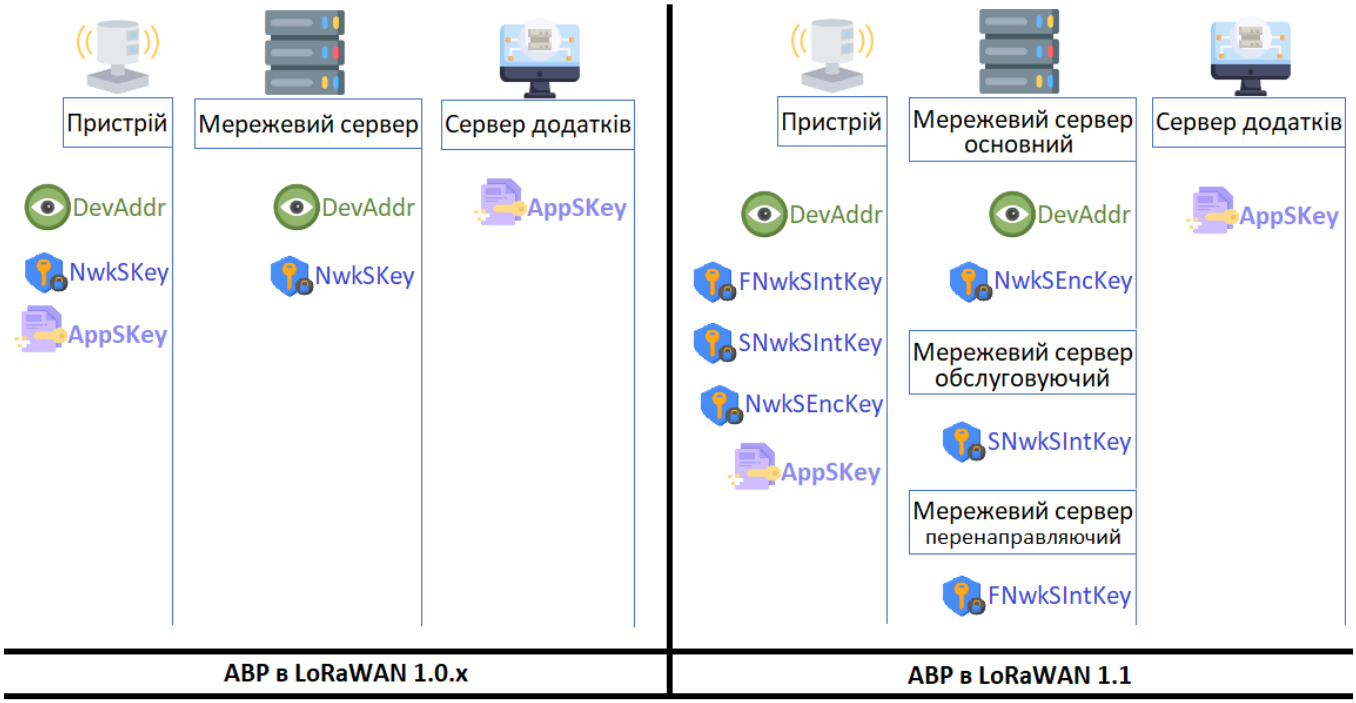

Рис. 1.11. Розподіл сесійних ключів в різних версіях протоколу ABP

Варто зазначити, що для протоколу ABP також є певні відмінності між версіями LoRaWAN 1.0.x та LoRaWAN 1.1 (рис. 1.11.), які стосуються безпеки та управління сесійними ключами. У LoRaWAN 1.0.x протокол ABP використовує лише два ключі — NwkSKey для забезпечення цілісності даних на рівні мережі та AppSKey для шифрування даних застосунку. В той час як в LoRaWAN 1.1 структура ключів для ABP стала складнішою, що підвищило безпеку протоколу. У новій версії протоколу було додано три ключі на рівні мережі: FNwkSIntKey, SNwkSIntKey та NwkSEncKey. Кожен з них виконує специфічні функції. Так,

наприклад, ключі FNwkSIntKey і SNwkSIntKey використовуються у парі та забезпечують цілісність даних для повідомлень, що відправляються та приймаються, а NwkSEncKey відповідає за шифрування команд управління MAC. Це дозволяє LoRaWAN 1.1 ефективніше контролювати безпеку обміну даними на різних рівнях і захищати повідомлення від можливих атак та перехоплень.

Як можна побачити, LoRaWAN використовує сучасні алгоритми і процедур безпеки, які дозволяють захищати та перевіряти цілісність даних від моменту їх передачі і до отримання. Завдяки розділенню сесійних ключів, протокол LoRaWAN забезпечує захищений канал зв'язку для даних застосунку та управління мережею на залежно один від одного. Додатково, можливість динамічної активації пристроїв за допомогою OTAA гарантує, що ключі безпеки можуть оновлюватися при кожному новому підключенні пристрою, що значно підвищує загальний рівень захищеності мережі. Окрім шифрування, LoRaWAN також забезпечує автентифікацію кожного пристрою під час підключення до мережі, додаючи ще один рівень захисту. Кожен пристрій має проходити процедуру автентифікації, яка перевіряє справжність джерела сигналу, а також його відповідність мережевим стандартам безпеки. Такий підхід гарантує легітимність кожного нового з'єднання в мережі, що значно підвищує загальну безпеку системи та знижує ризик атаки з боку сторонніх пристроїв або зловмисників.

Таким чином, LoRaWAN є основним доповненням до технології LoRa, оскільки забезпечує необхідну мережеву інфраструктуру, управління доступом, шифрування даних і функції автентифікації. Протокол LoRaWAN забезпечує саме ті функції мережевого рівня, які необхідні для створення масштабованих та безпечних IoT-систем. Без LoRaWAN звичайна технологія LoRa не має необхідних засобів шифрування і автентифікації, тому вразлива до несанкціонованого доступу та перехоплення даних.

1.3. Визначення та аналіз недоліків існуючих рішень забезпечення захищеної комунікації peer-to-peer пристроїв на основі технології LoRa та LoRaWAN

Розглянувши принципи роботи та технічні особливості технології LoRa, можна зрозуміти, що сама по собі вона не забезпечує надійного захисту даних. LoRa відповідає лише за фізичний рівень передачі сигналу і модуляцію, тому питання безпеки передачі інформації та автентифікації пристроїв у цій технології відсутні. Це означає, що дані, передані між пристроями, можуть бути вразливими для перехоплення або маніпуляції, особливо в сценаріях, де потрібна пряма комунікація пристроїв без залучення додаткових мережевих компонентів.

В розгалужених IoT-мережах, побудованих на основі технології LoRa питання безпечної комунікації вирішується за допомогою протоколу LoRaWAN, який доповнює LoRa механізмами шифрування та захисту на рівні мережі. Однак, попри всі переваги, LoRaWAN має низку суттєвих обмежень, особливо в контексті створення захищених peer-to-peer з'єднань.

Одним із головних недоліків LoRaWAN у забезпеченні безпечної peer-to-peer комунікації є її архітектурна залежність від централізованих компонентів, таких як шлюзи і різні сервери, які відповідають за конкретну роль у системі. Усі дані користувачів та пристроїв в мережі LoRaWAN проходять через ці сервери, де виконуються шифрування, автентифікація та розподіл ключів. Це створює центральну точку контролю, яка може стати вразливою для атак. У випадку, якщо центральні сервери піддаються кібератакам або виходять з ладу, функціонування всієї мережі може бути порушене, що є критичним недоліком для надійних і безперервних peer-to-peer з’єднань.

Також важливо зрозуміти, що централізована архітектура, характерна для LoRaWAN, з усіма її компонентами, ускладнює створення дійсно незалежних з’єднань «пристрій-пристрій». Така багатошарова інфраструктура була спеціально

розроблена для використання у сценаріях розгалужених IoT мереж, у яких дані надходять від великої кількості кінцевих пристроїв до одної центральної точки. У таких централізованих мережах, шлюзи і сервери виконують більшість функцій управління трафіком, шифрування, а також автентифікації повідомлень. Відповідно, кожен кінцевий пристрій в такій системі не має повного контролю над шифруванням і обробкою даних, оскільки ці процеси делеговані на сервери. Це створює значну архітектурну обмеженість та залежність, оскільки, щоб забезпечити захищену комунікацію, кінцевий пристрій має покладатися на зовнішні компоненти. Втрата зв’язку з сервером або його вихід із ладу повністю порушує функціонування такої схеми, оскільки з'єднання не може бути встановлено напряму між пристроями. Окрім цього, пересилання інформації через сервери і шлюзи веде до затримок, що створює незручності та перешкоди для динамічного та швидкого обміну даними між пристроями та їх керуванням в мережі. Таким чином, використання архітектури LoRaWAN для сценаріїв, де пристрої повинні напряму обмінюватися інформацією в реальному часі є непрактичним та надмірним.

Ще одним суттєвим недоліком LoRaWAN для динамічних peer-to-peer з'єднань є необхідність попереднього зберігання та розподілу ключів на обох кінцях потенційного з’єднання для забезпечення безпеки комунікації. У стандартних LoRaWAN-мережах секретний ключ, такий як AppKey, або його нова версія з двома ключами AppKey і NwkKey в LoRaWAN 1.1, мають бути заздалегідь збережені в пристроях, до початку процесу безпечної комунікації. Це створює додаткові складнощі для децентралізованих peer-to-peer з’єднань.

По-перше, для peer-to-peer мереж, де один пристрій зміг би напряму обмінюватись даними з іншим пристроєм, головною проблемою попереднього зберігання всіх ключів шифрування, окрім їх великої кількості, стає ризик компрометації пристрою, а разом з ним і всіх збережених ключів. Якщо один із пристроїв втрачає ключ або стає скомпрометованим, зловмисники потенційно можуть отримати доступ до ключів шифрування всіх пристроїв, з якими

скомпрометований пристрій мав з'єднання, що дасть змогу провести атаки на ці пристрої і порушити стабільне функціонування мережі. Це обмеження особливо значне для великих мереж, бо кожен пристрій потребує унікального ключа шифрування і дублювання ключа може привести до некоректної роботи мережі.

По-друге, необхідність попереднього розподілу AppKey знижує гнучкість та ефективність процесу розгортання нових пристроїв. Як видно на офіційній діаграмі [13] роботи LoRaWAN мережі (рис. 1.12), така система дуже покладається на централізоване управління, так як вона потребує чіткої та загальної координації роботи мережі на вищих рівнях, для забезпечення розподілу ключів шифрування та налаштування захищеної комунікації в мережі. Іншими словами, така система не підходить для децентралізованих мереж, так як потребує керуючої сутності, яка контролює попередній розподіл ключів шифрування. У децентралізованих peer-to-peer мережах не може бути централізованого сервера, що зберігав би ключі, а значить, така мережа має бути побудована за рахунок алгоритмів з динамічною природою шифрування.

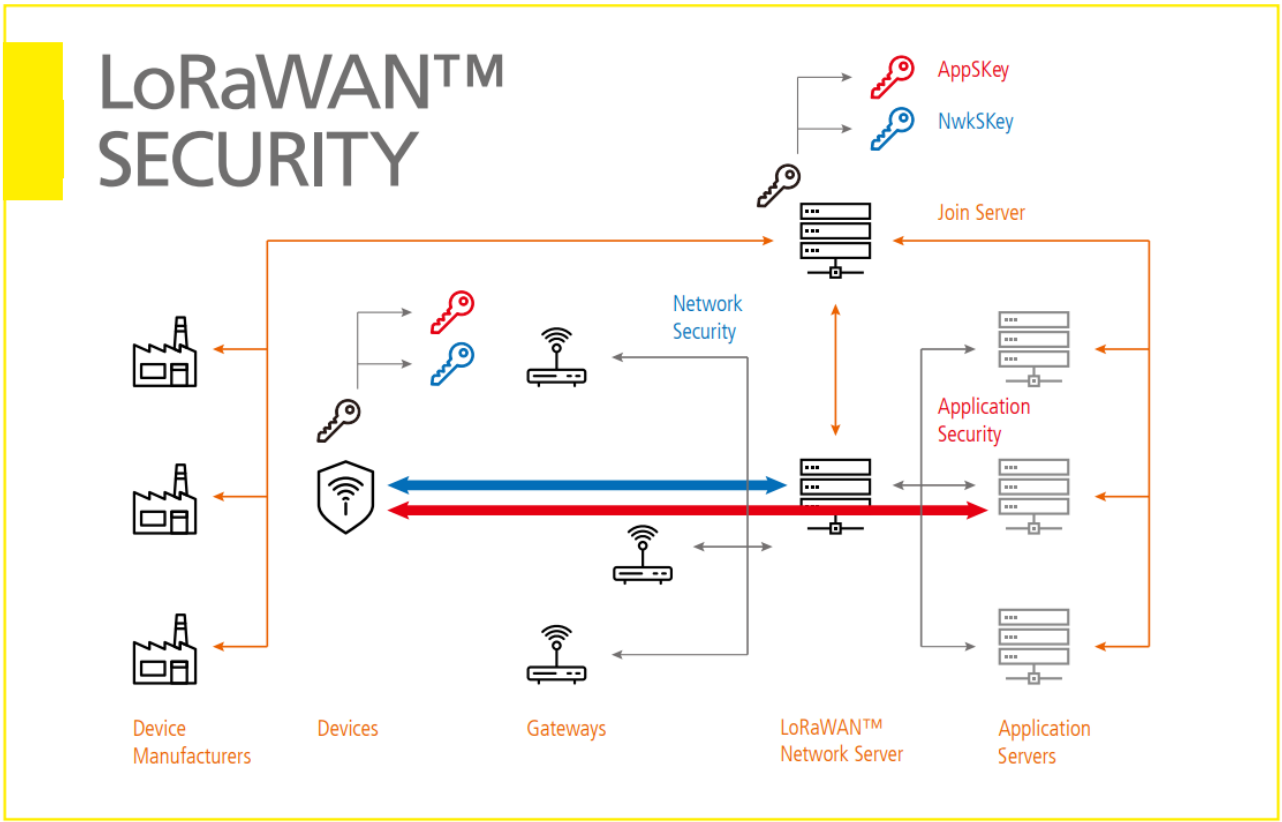

Рис.1.12. Діаграма розподілу ключів шифрування в LoRaWAN [13]

2 АНАЛІЗ МЕТОДІВ РЕАЛІЗАЦІЇ АЛГОРИТМУ ШИФРОВАНОЇ ТА НАДІЙНОЇ КОМУНІКАЦІЇ З ВИКОРИСТАННЯМ ВІДКРИТОГО КЛЮЧА НА ОСНОВІ ТЕХНОЛОГІЇ LORA

2.1. Аналіз існуючих асиметричних алгоритмів шифрування та їх недоліків

Після того як було розглянуто протокол LoRaWAN, його особливості і переваги, які він надає для технології LoRa, було виявлено кілька обмежень у його архітектурі для використання в децентралізованих peer-to-peer мережах Найважливішим з цих обмежень є залежність такого підходу від симетричного шифрування, що базується на попередньо розподілених секретних ключах. Такий підхід передбачав би зберігання однакових ключів на обох пристроях, які мають встановити захищене з’єднання. Це, у свою чергу, значно зменшує гнучкість мережі, позбавляє пристрої можливості динамічного налаштування захищеного з’єднання та створює проблеми з безпекою, адже компрометація одного пристрою або витік ключа ставить під загрозу всю мережу.

Вирішення цієї проблеми лежить в алгоритмах асиметричного шифрування, які пропонують можливість налаштування динамічних з’єднань без необхідності зберігання спільного секретного ключа на кожному пристрої, що надає гнучкості і динамічності в процес встановлення шифрованої комунікації. Асиметричне шифрування передбачає наявність пари ключів: публічного та приватного. Публічний ключ може бути поширений у мережі, тоді як приватний залишається на пристрої і має зберігатись в таємниці. Це дозволяє пристроям динамічно встановлювати з’єднання та захищено обмінюватися даними, навіть якщо вони раніше не мали зв'язку між собою.

Серед добре відомих та поширених алгоритмів шифрування з відкритим

ключем можна виділити наступні: Алгоритм RSA [14]; Алгоритм Diffie-Hellman [15]; Криптографія на еліптичних кривих (ECC) [16]; Алгоритм усічених поліноміальних кільцевих одиниць N-го ступеня (NTRU) [17]. Всі ці алгоритми мають власні особливості, сфери застосування і недоліки. Таким чином, для того щоб вибрати який з них підходить для задачі забезпечення надійного шифрування на пристроях з обмеженими ресурсами, варто розглянути принцип роботи кожного з них.

Алгоритм RSA, розроблений Рональдом Рівестом, Аді Шаміром та Леонардом Адлеманом у 1977 році, є одним із найвідоміших та найпоширеніших асиметричних алгоритмів у криптографії. Його популярність обумовлена високою надійністю і можливістю забезпечити безпечний обмін інформацією через Інтернет. Алгоритм RSA базується на задачі факторизації великих чисел, що є математично складною проблемою. Щоб зрозуміти, як працює цей алгоритм, розглянемо кожен з його етапів детальніше.

Перший етап — це генерація ключів. Для цього користувач вибирає два великих простих числа, позначені як p і q. Просте число — це таке число, яке ділиться лише на себе і на одиницю. Простота обраних чисел має велике значення, оскільки це впливає на стійкість алгоритму. Вибрані числа множать одне на одного, отримуючи значення n - модуль, що входить до складу як відкритого, так і закритого ключа.

Далі обчислюється функція Ейлера для n (2.1), яка визначається як кількість чисел, менших за n, які є взаємно простими з n.

$$\phi(n) = (p - 1) \times (q - 1) \tag{2.1}$$

Ця функція дозволяє далі обрати експоненту для шифрування — число e, яке є взаємно простим з $\phi(n)$, тобто не має з ним спільних дільників, крім одиниці. Число e зазвичай обирається невеликим для зручності, наприклад, 65537, яке є поширеним вибором через його простоту і ефективність. Після цього

обчислюється експонента дешифрування d (2.2), яка є оберненою до e за модулем $\phi(n)$.

$$d \times e \equiv 1(\mathrm{mod}\, \phi(n)) \tag{2.2}$$

Таким чином ми отримуємо згенеровані ключі: відкритий ключ складається з пари (e, n), а закритий ключ — з пари (d, n).

Тепер розглянемо, як відбувається шифрування за допомогою RSA. Припустимо, що Аліса хоче відправити зашифроване повідомлення Бобу, який раніше надав їй свій відкритий ключ (e, n). Аліса бере своє повідомлення M, яке перед шифруванням має бути перетворене у числовий формат, наприклад, використовуючи ASCII-коди символів. Вона обчислює шифротекст C за формулою 2.3, який після обчислення можна передавати Бобу по відкритому каналу зв'язку. Навіть якщо шифротекст C перехопить стороння особа, вона не зможе прочитати повідомлення без знання закритого ключа d.

$$C = M^e \bmod n \tag{2.3}$$

Коли Боб отримує шифротекст C, він використовує свій закритий ключ, що складається з пари (d, n), для розшифрування. Для цього він обчислює оригінальне повідомлення M за формулою 2.4.

$$M = C^d \bmod n \tag{2.4}$$

Таким чином, отримане значення M є початковим повідомленням, яке Аліса хотіла передати.

RSA також може використовуватися для створення цифрових підписів. Наприклад, якщо Боб хоче підписати документ, щоб підтвердити його автентичність, він використовує свій закритий ключ для створення підпису. Для цього він обчислює підпис S для повідомлення M за формулою 2.5.

$$S = M^d \bmod n \tag{2.5}$$

Після цього, підпис S разом із повідомленням M передається Алісі. Коли Аліса отримує це повідомлення, вона може перевірити підпис, використовуючи відкритий ключ Боба обчислити M' (2.6) і порівняти отримане значення з оригінальним повідомленням M. Якщо вони збігаються, це підтверджує, що підпис створено саме Бобом, і документ є автентичним.

$$M' = S^e \bmod n \tag{2.6}$$

RSA є розповсюдженим алгоритмом, проте він має певні недоліки. Найбільший з них полягає в обчисленнях з великими числами, що робить RSA доволі повільним порівняно з іншими алгоритмами. Крім того, для гарантування безпеки ключі повинні бути великими, зазвичай понад 2048 біт, що створює додаткові вимоги до обчислювальних ресурсів. Основна причина стійкості RSA - це складність факторизації великих чисел. Сучасні комп’ютери неспроможні розкласти на множники надто великі числа за доцільний час, проте розробка квантових комп'ютерів може створити загрозу для алгоритму, оскільки квантовий алгоритм Шора здатен розкладати великі числа значно швидше. Таким чином можна дійти висновку, що RSA не є ідеальним алгоритмом, особливо в контексті IoT та малопотужних пристроїв, які мають значні обмеження для розміру пам’яті та обчислювальних ресурсів.

Наступний за популярністю алгоритм - це алгоритм Диффі-Хеллмана (Diffie-Hellman), який був розроблений у 1976 році Вітфілдом Диффі та Мартіном Хеллманом і є основою для обміну ключами в криптографії. Також, він є одним з найперших асиметричних протоколів, який дозволяє двом сторонам, які не мають спільного секрету, створити спільний секретний ключ, який можна використовувати для подальшого шифрування даних. Алгоритм Диффі-Хеллмана

є основою для безпечного обміну ключами через незахищені канали зв'язку, наприклад, Інтернет чи радіоефір. Важливо зазначити, що сам алгоритм не використовує для шифрування або підпису дані, а тільки служить для створення спільного секретного ключа, який потім використовується для шифрування за допомогою симетричних алгоритмів.

Процес роботи алгоритму можна розділити на кілька етапів. Перший етап включає вибір публічних параметрів. Спочатку дві сторони, припустимо, Аліса та Боб, погоджуються на використання двох значень: велике просте число p і первісний елемент g, що є елементом групи за модулем p. Просте число p та число g є публічними і доступні обом сторонам. Для забезпечення достатнього рівня безпеки числа p та g повинні бути великими, щоб ускладнити їх розклад на множники або пошук циклічних елементів.

Далі кожна сторона генерує свої власні приватні ключі. Аліса вибирає випадкове число a, яке буде її приватним ключем, а Боб вибирає випадкове число b, яке стає його приватним ключем. Ці числа повинні бути конфіденційними і не мають передаватися іншим сторонам.

Після цього кожна сторона обчислює відкриті ключі. Аліса обчислює свій відкритий ключ A, використовуючи формулу 2.7, а Боб аналогічно обчислює свій відкритий ключ B, підставляючи свій приватний ключ у відповідне місце (2.8).

$$A = g^a \bmod p \tag{2.7}$$

$$B = g^b \bmod p \tag{2.8}$$

Тепер, коли Аліса і Боб мають свої відкриті ключі A і B, вони обмінюються ними через незахищений канал зв’язку. Варто зазначити, що, навіть якщо зловмисник перехопить ці відкриті ключі, він не зможе дізнатися приватні ключі або спільний секрет, оскільки задача дискретного логарифмування, тобто знаходження a або b, знаючи $g^a \bmod p$ або $g^b \bmod p$, є складною та не має швидкого розв'язку для великих чисел.

На наступному етапі обидві сторони використовують свої приватні ключі і отримані відкриті ключі для обчислення спільного секрету. Аліса використовує приватний ключ a і відкритий ключ Боба B для обчислення спільного секрету S_A (2.9), а Боб, в свою чергу, використовує приватний свій ключ b і отриманий відкритий ключ Аліси A для обчислення свого спільного секрету S_B (2.10).

$$S_A = B^a \bmod p \tag{2.9}$$

$$S_B = A^b \bmod p \tag{2.10}$$

Оскільки обидва ці вирази розраховуються однаково, спільний секрет Аліси S_A та спільний секрет Боба S_B будуть однаковими (2.11).

$$S_A = S_B = g^{ab} \bmod p \tag{2.11}$$

Цей спільний секрет S є однаковим для обох сторін, але при цьому жодна зі сторін не дізнається приватний ключ іншої сторони. Спільний секрет може бути використаний для подальшого симетричного шифрування повідомлень.

Алгоритм Диффі-Хеллмана є основним інструментом для безпечного обміну ключами через незахищені канали зв'язку. Зловмисник, перехопивши відкриті ключі та спробувавши застосувати алгоритми для зворотного обчислення приватних ключів або спільного секрету, зіткнеться з дуже складними математичними задачами, зокрема з дискретним логарифмуванням, яке є важко розв'язуваним для великих чисел. Однак алгоритм має свої обмеження - він не забезпечує автентифікацію, і тому може бути вразливим до атак типу «man-in-the-middle», коли зловмисник перехоплює і підміняє відкриті ключі. Окрім того, як і у випадку з RSA, надійність алгоритму Диффі-Хеллмана залежить від довжини використовуваного ключа, і базується на складності розкладання великих чисел на множники, а значить також є вразливим до атак квантових комп’ютерів за алгоритмом Шора.

Наступним зі списку йде алгоритм ECC. Криптографія на еліптичних кривих є потужним інструментом для забезпечення безпеки в сучасних криптографічних протоколах. Цей алгоритм є асиметричним, однак на відміну від традиційних методів, таких як RSA або Diffie-Hellman, ECC забезпечує більшу криптографічну безпеку при меншій довжині ключів, що робить його більш підходящим вибором для систем з обмеженими ресурсами.

Принцип роботи ECC базується на математичних властивостях еліптичних кривих, що використовуються для створення груп, на яких можна проводити операції додавання та множення. Еліптична крива визначається рівнянням вигляду 2.12, де a і b - це сталі числа, які визначають конкретну еліптичну криву.

$$y^2 = x^3 + ax + b \tag{2.12}$$

Важливо, щоб крива не мала особливих точок, таких як сингулярності, що може призвести до втрати її особливих властивостей. Параметри a і b обираються так, щоб забезпечити безпеку операцій на кривій.

Основна ідея ECC полягає в тому, що операції над точками на еліптичній кривій можуть бути використані для побудови криптографічних алгоритмів. У ECC найбільш важливою операцією є множення точки на певне число, тобто операція скалярного множення. Якщо ми маємо точку P на кривій, то множення цієї точки на скаляр k дає нову точку Q, яка також знаходиться на тій самій кривій.

Процес генерації ключів у ECC включає кілька етапів. Розглянемо його знов на прикладі Аліси і Боба. Ми будемо користуватися схемою, яка часто називається алгоритмом Ель-Гамаля на еліптичних кривих [18]. Припустимо Боб хоче надіслати зашифроване повідомлення M Алісі, використовуючи її відкритий ключ. Спочатку Аліса має згенерувати свій відкритий ключ. Для цього вона обирає еліптичну криву з певними параметрами a і b, що забезпечують потрібну

безпеку. Після цього вона обираю базову точку G та просте число n, що є порядком точки еліптичної кривої. Точка G - це така точка на кривій, що при багаторазовому додаванні до себе вона охоплює всі точки, що входять у цю групу, а n - це число точок, яке потрібно додати до G, щоб знову отримати цю ж точку на нескінченності, тобто нейтральний елемент групи. Значення n є дуже важливим параметром, тому що воно визначає розмір циклічної групи точок, які можуть бути згенеровані з G.

Потім Аліса обирає випадкове число d_A, яке стане її приватним ключем, при цьому d_A має бути в межах від нуля до *n-1* включно. Приватний ключ d_A використовується для множення початкової точки G на це число, отримуючи відкритий ключ Q_A.

Перевагою такого підходу є те, що зворотне обчислення приватного ключа d_A по відкритому ключу Q_A є дуже складною задачею, яка базується на проблемі дискретного логарифмування на еліптичних кривих. Це означає, що навіть при знанні точки Q_A і генератора G не існує швидкого способу обчислити d_A, що забезпечує високий рівень безпеки навіть при менших довжинах ключів.

Після цього, Боб, знаючи відкритий ключ Аліси Q_A, хоче надіслати їй зашифроване повідомлення M, яке перетворене в точку P_M на кривій. Якщо M - числове повідомлення, його можна просто закодувати як координати точки на еліптичній кривій або іншим методом перевести в точку P_M на кривій E.

Спочатку Боб вибирає випадкове число k, де $1 < k < n$. Це число використовується лише один раз і має бути секретним. Після цього він обчислює дві точки на еліптичній кривій. Перша точка C_1 (2.13) є частиною зашифрованого повідомлення і служить для введення випадковості в шифротекст. А друга точка - C_2 (2.14) є результатом додавання повідомлення P_M до точки $k \cdot Q_A$, що залежить від випадкового числа k і відкритого ключа Q_A.

$$C_1 = k \cdot G \tag{2.13}$$

$$C_2 = P_M + k \cdot Q_A \tag{2.14}$$

Боб надсилає Алісі пару точок (C_1, C_2), які утворюють зашифроване повідомлення. Ці дві точки C_1 та C_2 разом утворюють еліптичні точки, що кодують повідомлення і випадковий компонент k, необхідний для безпеки.

Отримавши зашифроване повідомлення, Аліса може розшифрувати його за допомогою свого приватного ключа d_A. Аліса спочатку обчислює точку $d_A \cdot C_1$ за формулою 2.15.

$$d_A \cdot C_1 = d_A \cdot (k \cdot G) = k \cdot (d_A \cdot G) = k \cdot Q_A \tag{2.15}$$

Таким чином, $k \cdot Q_A$ є точкою, яку Аліса може обчислити, використовуючи лише свій приватний ключ d_A. Після цього Аліса віднімає точку $k \cdot Q_A$ від C_2, щоб отримати початкове повідомлення (2.16).

$$P_M = C_2 - d_A \cdot C_1 \tag{2.16}$$

Після обчислення цього рівняння вона отримує точку P_M, яка кодує початкове повідомлення M.

Алгоритм шифрування та підпису в ECC має багато спільного з іншими асиметричними алгоритмами. Наприклад, для шифрування повідомлення використовується публічний ключ, а для розшифрування — приватний. У випадку підпису повідомлення користувач створює підпис за допомогою свого приватного ключа, який може бути перевірений за допомогою відкритого ключа. Однак, операції з точками на еліптичних кривих дозволяють досягати тієї ж безпеки, при значно менших довжинах ключів. Наприклад, ключ розміром 256 біт у ECC забезпечує рівень безпеки, еквівалентний ключу RSA розміром 3072 біти. Це важливо для пристроїв з обмеженими ресурсами, де кожен біт пам’яті та кожен цикл процесора мають значення.

Але при всіх його перевагах, дуже важлива його правильна реалізація. Якщо алгоритм ECC не реалізовано правильно, це може призвести до вразливостей, оскільки він дуже чутливий до помилок при генерації чисел або виборі параметрів кривої. Тому важливо використовувати стандартні та добре протестовані бібліотеки для реалізації ECC. Окрім того, як RSA і Diffie-Hellman, ECC потенційно може бути вразливим до квантових атак, особливо через розвиток алгоритмів Шора, які можуть розв’язувати проблеми дискретного логарифмування на еліптичних кривих. Це обмежує довгострокову надійність ECC у майбутньому, коли квантові комп'ютери стануть доступними.

Останнім у списку є алгоритм NTRU. Він є одним із відомих постквантових криптографічних алгоритмів, розроблених для забезпечення стійкості до атак квантових комп'ютерів. Цей алгоритм базується на складності розв'язання задач з багатовимірними поліномами у кільці цілих чисел і може використовуватися як для шифрування, так і для цифрового підпису. Алгоритм NTRU використовує набір поліномів з цілими коефіцієнтами і спеціальні операції над ними, щоб створювати приватні та публічні ключі, шифрувати і дешифрувати повідомлення. Основою його стійкості є складність розв'язання задачі знаходження найближчої вектора у багатовимірному просторі.

Процес шифрування та розшифрування у NTRU заснований на трьох поліномах: відкритому ключі, секретному ключі і випадковому повідомленні, яке шифрується. Спочатку визначаються параметри, які використовуються в алгоритмі. Існують три основні параметри N, p і q. Значення N представляє ступінь поліномів, які використовуються в обчисленнях, тобто кількість коефіцієнтів у поліномі. Параметр p зазвичай є малим простим числом, яке допомагає зменшити складність обчислень, тоді як параметр q є більшим числом, яке використовується для обмеження результатів операцій в модульному обчисленні. У більшості реалізацій p обирається як число 3, а q має бути значно більшим, щоб забезпечити достатню криптографічну стійкість алгоритму.

Для генерації ключів у NTRU спочатку обираються два поліноми f і g випадковим чином із певного множини поліномів. Поліном f повинен бути таким, щоб його зворотній поліном існував у кільцях R_q(2.17) та R_p(2.18), де $\mathbb{Z}_q$ вказує на множину цілих чисел за модулем q, тобто на залишкові класи чисел від 0 до $q-1$, а у випадку $\mathbb{Z}_p$ йдеться про ціли числа за модулем p, де p і q є цілими числами, причому зазвичай q значно більше за p. Це забезпечує можливість коректного розшифрування повідомлень у майбутньому.

$$R_q = \mathbb{Z}_q[x]/(x^N - 1) \tag{2.17}$$

$$R_p = \mathbb{Z}_p[x]/(x^N - 1) \tag{2.18}$$

Поліном f вибирається так, щоб його коефіцієнти задовольняли певні умови, наприклад, значення близькі до нуля, що допомагає контролювати розмір коефіцієнтів після операцій модулярного множення. Потім обчислюється зворотний поліном f_q^{-1} від f у кільці R_q та зворотний поліном f_p^{-1} від f у кільці R_p. Важливо зазначити, що знаходження зворотного полінома є складною задачею і потребує виконання спеціальних алгоритмів, але цей етап виконується лише при створенні ключів, тому не впливає на швидкість при шифруванні та розшифруванні.

Публічний ключ h обчислюється як добуток полінома g на зворотний поліном f_q^{-1} за модулем q (2.19). Цей публічний ключ h, а також значення N, p та q є загальнодоступними і можуть використовуватися ким-завгодно для шифрування повідомлень.

$$h = f_q^{-1} \cdot g \bmod q \tag{2.19}$$

Приватний ключ складається із полінома f, а також обчислених зворотних поліномів f_q^{-1} та f_p^{-1}.

Процес шифрування в NTRU починається з того, що відправник генерує

випадковий поліном r, коефіцієнти якого є малими цілими числами, наприклад в межах від -1 до 1. Це допомагає забезпечити додатковий рівень випадковості для шифротексту. Після цього, повідомлення m представляється у вигляді полінома, коефіцієнти якого належать множині значень за модулем p. Далі обчислюється шифротекст e (2.20). Такі маніпуляції додають "шуму" до оригінального повідомлення m, ускладнюючи розшифрування без знання приватного ключа. По закінченню розрахунків, шифротекст e може бути переданий по відкритому каналу зв’язку.

$$e = r \cdot h + m \tag{2.20}$$

Для розшифрування отриманого шифротексту e власник приватного ключа спочатку обчислює поліном a за формулою 2.21, та оскільки e містить добуток $r \cdot h + m$, а публічний ключ h визначено як $f_q^{-1} \cdot g \bmod q$, підставлення цих значень дозволяє отримати вираз, який після скорочення включає поліном m з деяким малим шумом. Але цей результат легко зводиться до значень за модулем p, що дозволяє позбутися шуму і відновити початкове повідомлення m.

$$a = f \cdot e \bmod q \tag{2.21}$$

Алгоритм NTRU базується на алгебраїчних обчисленнях у просторі поліномів та має особливі властивості, які роблять його дуже швидким і ефективним порівняно з іншими методами шифрування, такими як RSA або ECC. Основою NTRU є обчислення в кільцях поліномів зі спеціальними властивостями, що дозволяє йому виконувати операції швидше, зокрема завдяки меншим обсягам ключів, що забезпечує більшу продуктивність на тому ж рівні захисту.

Проаналізувавши найпопулярніші алгоритми асиметричного шифрування, можна дійти висновку, що у випадку налаштування безпечного peer-to-peer

з’єднання для пристроїв з малим об’ємом обчислювальних ресурсів, але для забезпечення надійного та захищеного каналу зв’язку між пристроями, можна використовувати два алгоритми: алгоритмом Ель-Гамаля на еліптичних кривих, та алгоритм NTRU. В порівняння з тим же RSA, ці два алгоритми мають достатньо гарний захист при невеликих розмірах ключа (рис 2.1), що критично важливо при обмеженому розмірі пам’яті на мікроконтролерах та пристроях.

Security Level (bits)	RSA key sizes (bits)	ECC Key sizes (bits)	NTRU Key sizes (bits)
80	1024	163	2008
112	2048	224	3033
128	3072	256	3501
192	7680	384	5193
256	15360	512	7690

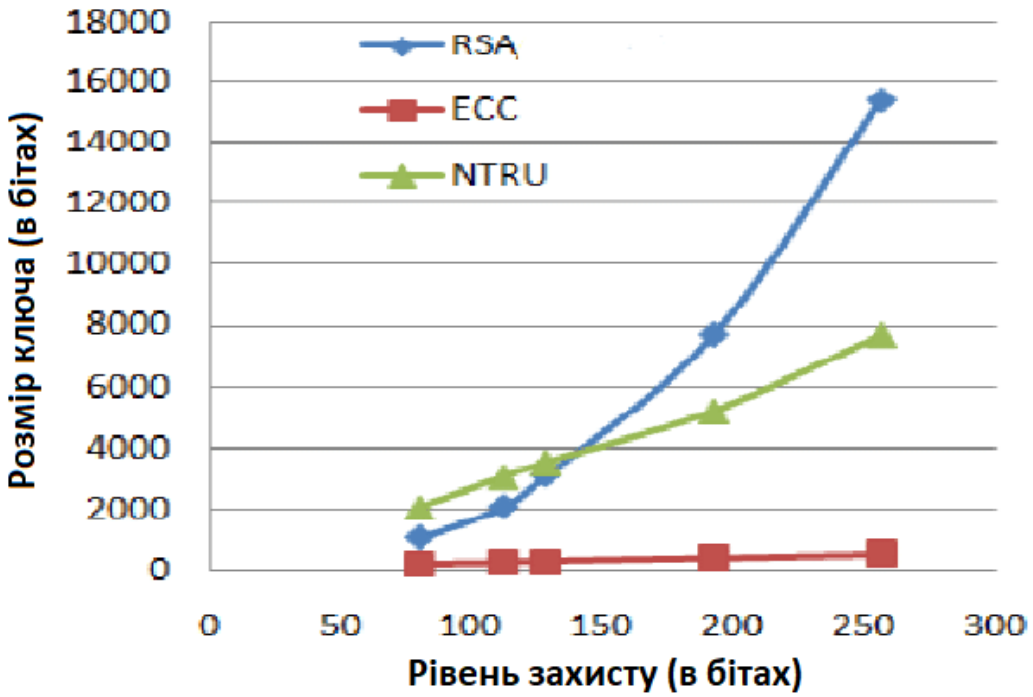

Рис 2.1. Порівняння співвідношень розміру ключа до рівня захисту алгоритму

Також слід зазначити, що алгоритмом Ель-Гамаля на еліптичних є простішим в імплементації, порівняно з алгоритмом NTRU, але при цьому він вразливий до атак квантових комп’ютерів та алгоритму Шора. Тому, хоч і загроза

квантової вразливості поки є тільки теоретичною, через непоширеність квантових комп’ютерів, NTRU є більш надійним і безпечним на сьогоднішній день, але натомість потребує складнішої реалізації та більших розрахунків при генерації ключів, що може стати проблемою для пристроїв з обмеженими обчислювальними ресурсами.

2.2. Аналіз існуючих технологій та алгоритмів автентифікації

Слід пам’ятати, що окрім шифрування даних, що передаються піж пристроями, не менш важливим компонентом безпечної комунікації є автентифікація цих самих пристроїв. Особливо в peer-to-peer мережах, автентифікація є критично важливим компонентом для забезпечення безпеки, оскільки вона спрямована на підтвердження ідентичності, що взаємодіють у мережі. На відміну від централізованих систем, де за процес ідентифікацію відповідає сервер, децентралізована природа P2P мереж ускладнює цей процес через відсутність єдиного пункту перевірки. Це створює додаткові виклики, які потребують підходів і технологій, що використовуються для реалізації автентифікації в таких децентралізованих мережах.

Автентифікація в peer-to-peer мережах виконує низку важливих функцій. По-перше, вона забезпечує ідентифікацію пристроїв чи користувачів, дозволяючи мережі підтвердити, що конкретний учасник є тим, за кого себе видає. Це необхідно для запобігання доступу неавторизованих учасників, які можуть загрожувати безпеці. Крім того, автентифікація сприяє захисту від різноманітних атак, таких як атака MITM, підробка ідентичностей або інші дії шкідливих вузлів, які можуть порушити функціонування мережі.

Іншою ключовою функцією автентифікації є забезпечення довіри між учасниками. У децентралізованому середовищі важливо створити механізми, які

дозволять вузлам мати впевненість один в одному. Завдяки надійній автентифікації встановлюються довірливі зв'язки між вузлами, що гарантує безпечну передачу даних лише між підтвердженими учасниками мережі.

Одним з головних компонентів в алгоритмі автентифікації є цифровий сертифікат [19]. Цифровий сертифікат — це електронний документ, який використовується для підтвердження автентичності пристрою, сервера або іншого суб’єкта електронної мережі. Його основне завдання полягає в забезпеченні довіри до пристрою, з яким встановлюється зв'язок, що є особливо важливим для захисту від шахрайства та підробок. Основою цифрових сертифікатів є система публічних і приватних ключів, яка гарантує, що власник сертифіката є справжнім володарем певного відкритого ключа.

Загалом процес створення цифрового сертифікату може варіюватись в залежності від цілі створення сертифікату і використання сторонніх сертифікаційних центрів, але загальний порядок дій, який може бути запропонований для використаний при децентралізованому підході автентифікації виглядає наступним чином.

Спершу потрібно створити криптографічно залежну пару ключів: приватний і публічний. Приватний ключ зберігається в таємниці і нікому не передається, так як він необхідний для підписання сертифіката. Публічний ключ є частиною самого сертифіката і буде використовуватися іншими пристроями або системами для перевірки підпису та автентичності.

Далі створюється сам документ, який має бути підписаний. Зазвичай цей містить важливу для перевіряючого пристрою інформацію про власника сертифіката, яку він хоче вберегти від зміни сторонніми суб’єктами. До цього можуть відноситись такі дані, як ID девайсу, його виробник або власник, коли і ким цей сертифікат був затверджений, а також публічний ключ пристрою. Сертифікат також може також містити строк дії, щоб обмежити його використання до певного часу. На цьому етапі сертифікат — це лише набір даних, які ми хочемо використовувати для автентифікації пристрою в мережі.

Третім кроком, оскільки ми хочемо, щоб сертифікат підтверджував саме автентичність нашого пристрою, необхідно його підписати. Для цього ми використовуємо приватний ключ. Тут слід зазначити, що у випадку підписання сертифікату для пристрою самим же пристроєм, використовуються приватний ключ цього ж пристрою. Але якщо підпис робиться стороннім суб'єктом, який виступає в ролі гарантодавця для цього пристрою, як наприклад сертифікаційний сервіс або виробник, то використовується приватний ключ саме цього суб'єкту, який гарантує автентичність даного пристрою.

Коли сертифікат підписується, спочатку обчислюється його хеш. Це робиться для того, щоб створити унікальний «відбиток» сертифіката - невеликий блок даних фіксованого розміру, який представляє весь сертифікат. Хеш-функції, як-от наприклад SHA-256, створюють цей відбиток, який дуже важко підробити або змінити, бо якщо змінити навіть невелику частину сертифіката, то та ж саме хеш функція дасть вже зовсім інший результат. Далі цей хеш, шифрується приватним ключем власника, і результат цього шифрування називається цифровим підписом. Такий підхід дозволяє створити компактний і захищений підпис сертифіката.

Після того як сертифікат підписано, він готовий до використання. Сертифікат можна передати іншим пристроям або системам, які будуть перевіряти його для автентифікації вашого пристрою.

Коли інший пристрій отримує сертифікат і хоче перевірити його автентичність, він використовує публічний ключ, що є публічним по відношенню до того приватного, яким цей сертифікат був підписаний. Тобто у випадку самопідписання сертифікату, при перевірці використовується публічний ключ що міститься в самому сертифікаті, а у випадку підписання сертифікату для пристрою іншим довіреним суб'єктом, використовується саме публічний ключ цього суб'єкта. Тож перевіряючий пристрій використовуючи публічний ключ, щоб отримати оригінальний хеш, який було зашифровано під час створення підпису. Потім цей пристрій знову обчислює хеш отриманого сертифіката за

допомогою тієї ж хеш-функції, яка використовувалась при створенні підпису. Якщо обчислений хеш і той, що був отриманий з підпису, збігаються, це означає, що сертифікат не змінювався з моменту підписання, і він справді був підписаний власником приватного ключа, що підтверджує автентичність того пристрою, що перевертається.

Однак у випадку децентралізованої мережі, використання самопідписаного сертифікату, тобто такого, що створений пристроєм або користувачем для підтвердження своєї ж автентичності, не може використовуватись через відсутність попередньої довіри до цього пристрою. У такому сертифікаті пристрій самостійно підписує свій публічний ключ, підтверджуючи його справжність своїм власним підписом, тобто приватним ключем. Основна проблема такого підходу полягає у відсутності зовнішньої перевірки, яка б гарантувала, що сертифікат дійсно належить заявленому пристрою або користувачу, і що підписаний сертифікат є справжнім. У централізованих системах ієрархії сертифікаційних органів використовуються так звані «ланцюжки довіри» для перевірки автентичності сертифікатів. У peer-to-peer мережах така центральна перевірка відсутня, тому кожен учасник повинен мати можливість самостійно перевірити справжність іншого пристрою без звернення до центрального органу. Якщо пристрій надсилає самопідписаний сертифікат, інші вузли не можуть бути впевнені, що цей сертифікат не є підробкою або що він дійсно належить заявленому пристрою. Сам підпис не надає достатніх доказів автентичності, оскільки відсутня стороння перевірка, тож будь-який зловмисник може створити сертифікат із довільними даними і самостійно його підписати. У цьому випадку інші учасники мережі мають ризик прийняти підроблений сертифікат за справжній, що дозволяє зловмиснику видавати себе за довіреного учасника, перехоплювати або змінювати дані, порушуючи безпеку мережі.

Для забезпечення надійної автентифікації у децентралізованих мережах потрібні системи, які дозволяють кожному учаснику підтвердити автентичність іншого пристрою, не покладаючись на централізовану перевірку. Одним із

підходів є використання системи «блокчейн» [20] для зберігання інформації про сертифікати. Блокчейн - це розподілений реєстр, який зберігає інформацію в послідовних блоках, де кожен новий блок пов’язаний з попереднім. Усі записи в блокчейні незмінні та відкриті для перевірки всіма учасниками мережі. Це дозволяє зберігати дані про сертифікати в децентралізованому реєстрі, де кожен може переконатися в їхній справжності.

Сам процес створення блоку та додавання інформації про новий пристрій в блокчейн складається з декількох етапів. На першому етапі пристрій генерує пару ключів - приватний і публічний. Приватний ключ зберігається на пристрої і використовується для підписування повідомлень, а публічний ключ буде доступним для всіх і слугуватиме ідентифікатором пристрою. Щоб зареєструватися в мережі, новий пристрій формує транзакцію з даними про себе. Як згадувалось раніше, це можуть бути будь-які важливі для ідентифікації дані, зокрема, унікальний ID пристрою або серійний номер, який дозволить іншим учасникам однозначно ідентифікувати його, а також публічний ключ самого пристрою.

Після формування транзакції пристрій підписує її своїм приватним ключем, як це вже було розглянуто вище. Далі ця підписана транзакція передається в мережу, де інші вузли приймають її для обробки. Щоб додати нову транзакцію в блокчейн, мережа використовує механізм консенсусу, який передбачає, що більшість вузлів у мережі мають погодитися на її включення в ланцюг. Кожен вузол перевіряє транзакцію, розшифровуючи підпис за допомогою публічного ключа, що міститься в транзакції та порівнюючи отриманий хеш з даних та той що був розшифрований, таким чином підтверджуючи, що підпис дійсно належить пристрою, що надав ці дані. Якщо транзакція виглядає справжньою, її додають у чергу для включення в блокчейн.

Далі вузли, які займаються створенням нових блоків, вибирають певну кількість підтверджених транзакцій з пулу і починають формувати з них новий блок. У процесі створення блоку важливу роль відіграє механізм консенсусу, який

забезпечує узгодження між вузлами, хто саме має право створювати блок. Існує кілька різних алгоритмів консенсусу, таких як Proof of Work та Proof of Stake, які застосовуються залежно від конкретної блокчейн-мережі. Наприклад, у випадку Proof of Work, вузли-конкуренти, які називаються майнерами, змагаються за те хто першим знайде рішення для складної криптографічної задачі. Так вони мають знайти спеціальне число, зване «nonce», яке, будучи доданим до блоку разом з іншими даними, генерує хеш із певними властивостями. Зазвичай, хеш повинен починатися з певної кількості нулів, і знайти nonce який задовільнив би таку вимогу дуже важко з обчислювальної точки зору. Майнери постійно перебирають варіанти різні nonce, щоб знайти хеш, що відповідає вимогам мережі. Як тільки один із них знаходить такий хеш, він оголошує про це всій мережі, і блок вважається готовим для включення в блокчейн. Інші вузли швидко перевіряють хеш і підтверджують, що він дійсно відповідає критеріям, після чого блок приймається мережею.

Коли блок остаточно сформовано, до нього додається спеціальний хеш попереднього блоку. Цей хеш є частиною заголовка нового блоку, і його використання забезпечує ланцюговий зв’язок між усіма блоками в блокчейні. Цей хеш попереднього блоку є критично важливим для збереження послідовності та цілісності блокчейну, оскільки будь-яка спроба змінити навіть один блок в послідовності призведе до зміни всіх наступних хешів, що швидко виявлять інші вузли мережі.

Після підтвердження та створення нового блоку з транзакцією він додається в блокчейн, де запис стає доступним для всіх учасників мережі. Блокчейн, як розподілений реєстр, зберігає цю інформацію незмінно разом з іншими даними про пристрої. Це забезпечує незмінність та прозорість, що унеможливлює підробку чи видалення даних про автентичність пристрою. Так будь-який учасник мережі має доступ до історії всіх зареєстрованих пристроїв і може використовувати її для перевірки автентичності будь-якого пристрою.

Коли інший пристрій мережі хоче перевірити справжність нового пристрою,

він може звернутися до блокчейну і знайти запис, що відповідає публічному ключу або ID цього пристрою. Для перевірки автентичності інший пристрій може надіслати новому пристрою певне повідомлення і попросити його підписати це повідомлення своїм приватним ключем. Новий пристрій підписує повідомлення і повертає цей підпис. Далі пристрій, що ініціював перевірку, бере публічний ключ нового пристрою з блокчейну і використовує його для перевірки підпису отриманого повідомлення. Якщо підпис дійсний і співпадає з даними, збереженими в блокчейні, це означає, що новий пристрій є автентичним і має доступ до приватного ключа, відповідного публічному ключу в реєстрі. Таким чином, блокчейн виступає надійним джерелом інформації про автентичність пристроїв, оскільки записані в ньому дані незмінні, доступні для всіх і підтверджені мережею. Це дозволяє будь-якому пристрою в мережі перевірити автентичність іншого пристрою без необхідності звертатися до центрального органу, використовуючи лише дані, доступні в блокчейні.

Для забезпечення децентралізованого зберігання реєстру блокчейну, кожен з пристроїв має зберігати хоча б більшу частину всього блокчейну. В контексті мережі пристроїв з обмеженим об’ємом пам’яті, блокчейн не є оптимальним вибором через високі вимоги до обчислювальних ресурсів, зберігання даних і енергоспоживання. Алгоритми консенсусу, такі як Proof of Work, вимагають значної обчислювальної потужності, оскільки вузли повинні розв'язувати складні криптографічні задачі. Це робить підтримку блокчейну надто ресурсоємною для пристроїв з обмеженими можливостями.

Крім того, оскільки кожен пристрій повинен зберігати повну копію ланцюга блоків, це спричиняє значне навантаження на пам'ять та обсяг зберігання. Для малопотужних пристроїв, у яких обсяг пам’яті тривалого зберігання та оперативної пам'яті обмежений, зберігання великого та постійно зростаючого блокчейну є проблематичним. І нарешті, для роботи в режимі реального часу блокчейн не підходить через затримки, викликані перевіркою транзакцій і досягненням консенсусу, що також може не відповідати потребам пристроїв, які

мають обмежений доступ до джерел живлення або працюють в умовах економії енергії.

Таким чином, для малопотужних пристроїв потрібна більш проста та менш ресурсоємна система, і такою системою є технологія «Web of Trust» [21]. Ця система побудована на принципах взаємної довіри між учасниками мережі, без потреби в обчислюванні важких алгоритмах консенсусу чи зберіганні великих обсягів даних, як у блокчейні. Web of Trust ефективно використовує існуючі зв'язки між учасниками, які вже довіряють один одному, створюючи мережу довірених пристроїв, що дозволяє новим пристроям приєднуватися до мережі через затвердження з боку інших довірених вузлів.

На першому етапі Web of Trust кожен пристрій генерує пару криптографічних ключів. Приватний ключ зберігається на пристрої і використовується для підписання даних, підтверджуючи, що повідомлення або транзакції дійсно надіслані цим пристроєм. Публічний ключ буде доступним іншим пристроям у мережі для перевірки підписів і його можна вважати ідентифікатором пристрою.

Далі кожен пристрій, який вже має довірені відносини з іншими пристроями в мережі, може підписати публічні ключі цих пристроїв, створюючи свідоцтво про довіру, яке називається «trust certificate». Ці попередні довіреності можуть бути обумовлені належністю пристроїв до одного користувача або організації, чи це можуть бути пристрої людей, які знайомі один з одним у реальному світі і тому вони знаються, що можуть довіряти один одному. Тому це свідоцтво, або ж цифровий підпис, означає, що підписувач довіряє автентичності пристрою, чий ключ він підписує. Наприклад, якщо пристрої *A* і *B* вже довіряють один одному, пристрій *A* може підписати публічний ключ пристрою *B*, підтверджуючи його як надійного учасника. При цьому пристрій *A* зберігає в себе публічний ключ пристрою *B,* який буде використовувати не тільки для комунікації з нам, а й для перевірки сертифікатів, підписаних пристроєм *B* для інших пристроїв. А пристрій *B* в свою чергу буде зберігати сертифікат, підписаний приватним ключем

пристрою *A* для публічного ключа пристрою *B*. Таким чином коли інший пристрій, який ще не має довіри до пристрою *B*, але при цьому має довіру до пристрою *A*, захоче перевірити автентичність пристрою *B*, за допомогою публічного ключа пристрою *A*, до якого в нього вже є довіра, він зможе перевірити trust certificate наданий пристроєм *A* для пристрою *B*.

В іншій ситуації, коли новий пристрій приєднується до мережі, він має отримати схвалення від одного чи кількох довірених пристроїв, щоб інші пристрої могли йому довіряти. Наприклад, пристрій *C* хоче приєднатися до мережі, де вже існують довірені зв'язки між пристроями *A* і *B*. Для цього пристрій *C* спочатку звертається до пристрою *B* для отримання його схвалення. Якщо пристрій *B* довіряє пристрою *C*, він підписує його публічний ключ, таким чином створюючи нове свідоцтво про довіру. Ця підписана інформація додається до профілю пристрою *C* як доказ його автентичності в мережі.

Тепер інші пристрої в мережі можуть проводити перевірки автентичності в такій системі довіри, що може включати декілька етапів. Припустимо пристрій *C* хоче відправити повідомлення або запит до пристрою *D*. Для цього він підписує повідомлення своїм приватним ключем. Пристрій *D*, який отримав повідомлення від пристрою *C*, використовує публічний ключ пристрою *C* для первинної перевірки підпису. Ця перевірка підтверджує, що повідомлення дійсно було підписано власником приватного ключа і не зазнало змін. Однак сам факт підпису не гарантує автентичності відправника, якщо пристрій *D* ще не довіряє пристрою *C* безпосередньо. У такому випадку *D* шукає ланцюжок довіри через інші пристрої, яким він довіряє. Першим кроком пристрій *D* перевіряє, чи є в його списку довірених пристроїв інший пристрій, наприклад, пристрій *B*, який підписав публічний ключ пристрою *C*. Якщо такий підпис знайдено, *D* визнає *C* автентичним, оскільки *B* довіряє *C*, а *D* довіряє *B*. Якщо ж пристрій *D* ще не довіряє пристрою *B*, тоді *D* шукає в своїй базі довіри наступний пристрій, якому він довіряє, наприклад, *A*, та перевіряє, чи той підписав ключ *B*, а *B*, у свою чергу, підписав ключ *C*. У цьому випадку формується ланцюжок довіри: $D \rightarrow A \rightarrow B \rightarrow$

C. Таким чином, *D* може довіряти *C* через проміжні довірені зв’язки. Такий ланцюжок довіри може включати кілька проміжних пристроїв, проте із збільшенням кількості проміжних вузлів довіра може знижуватися, що може бути враховано в налаштуваннях системи. Деякі системи обмежують довжину ланцюжка для швидшої перевірки або на випадок зниження рівня довіри.

Коли пристрій *D* знаходить потенційний ланцюжок довіри, він перевіряє підпис на кожному кроці цього ланцюга. Наприклад, якщо *D* довіряє *A*, він спочатку перевіряє підпис *A*, щоб переконатися, що ключ *B* дійсно підписано *A*. Потім перевіряє підпис *B*, щоб упевнитися, що ключ *C* підписано *B*. Цей процес дозволяє пристрою *D* поступово підтвердити автентичність *C*, не потребуючи централізованого сховища, незалежної довіреної сторони або складних обчислювальних операцій, що робить цю систему ідеальною для малопотужних пристроїв.

Таким чином, технологія Web of Trust у поєднанні з цифровими сертифікатами та цифровими підписами забезпечує надійну основу для перевірки автентичності в децентралізованих мережах. Така комбінація технологій дозволяє створювати мережу довіри між пристроями, що взаємодіють без централізованого органу контролю, що особливо корисно для систем, де учасники можуть не мати попередніх знань один про одного.

Цифрові сертифікати підтверджують автентичність публічних ключів, використовуючи підписи довірених учасників або пристроїв, які вже мають підтверджену репутацію. Завдяки цьому будь-який пристрій у мережі може перевірити підпис іншого пристрою через ланцюжок довіри, перевіряючи послідовність підписів до знайомого йому довіреного пристрою.

Поєднання цих технологій дозволяє створити безпечний процес аутентифікації, знижуючи ризик несанкціонованого доступу та підробки ідентифікаційних даних, навіть у повністю децентралізованому середовищі.

3 РОЗРОБЛЕННЯ ВАРІАНТУ ТЕХНОЛОГІЇ СТВОРЕННЯ ШИФРОВАНОЇ КОМУНІКАЦІЇ PEER-TO-PEER ПРИСТРОЇВ НА БАЗІ ТЕХНОЛОГІЇ LORA

3.1. Визначення додаткових особливостей і ключових елементів алгоритму для створення шифрованої комунікації

Після того як було розглянуто основні методи захисту комунікації за допомогою асиметричних алгоритмів шифрування, методи автентифікації пристроїв у децентралізованій мережі, а також технологію LoRaWAN та її особливості і аспекти захисту, які вона надає для технології LoRa, тепер треба сформулювати додаткові вимоги до протоколу, який забезпечить надійне та безпечне з’єднання для peer-to-peer комунікації.

Однією з загроз, від яких не захищають алгоритми шифрування та методи автентифікація, є replay-атаки. Цей тип атак полягає у тому, що зловмисник може перехопити повідомлення та повторно відправити його до одержувача, навіть не розшифровуючи вміст. У децентралізованій LoRa мережі це може призвести до серйозних наслідків, особливо якщо пристрій реагує на отримане повідомлення автоматично, наприклад, виконуючи певну дію. В умовах мережі IoT це може, зокрема, вплинути на з'єднання датчиків та виконавчих пристроїв, створюючи вразливість до несанкціонованих команд. Такі повтори, зважаючи на обмеженість ресурсів пристроїв, можуть порушити їх нормальне функціонування, призводячи до перевантаження або небажаних результатів. Щоб уникнути цієї загрози, доцільно впроваджувати механізм динамічних тегів або унікальних ідентифікаторів, як наприклад, часових відміток чи лічильників, які гарантуватимуть унікальність кожного повідомлення та дозволять приймачу ігнорувати вже отримані раніше копії.

Захист цілісності даних є ще однією важливою вимогою для забезпечення надійної комунікації між пристроями LoRa. Під цілісністю даних розуміється гарантія того, що повідомлення не було змінено або спотворено під час передачі. Це особливо критично для LoRa мереж, оскільки повідомлення в цій технології зазвичай мають невеликий розмір, що робить навіть невелику зміну інформації вразливою до її неправильного тлумачення одержувачем. Наприклад, незначна зміна у кількох байтах може призвести до того, що сенсорні дані будуть інтерпретовані як зовсім інші значення, або команда для пристрою буде виконана неправильно, що є небезпечним для систем, де точність і надійність передачі є критичними.

Механізм захисту цілісності даних може бути реалізований шляхом додавання цифрових підписів або MIC кодів до повідомлень, які дозволяють одержувачу перевіряти, що повідомлення не зазнало змін на шляху між відправником та приймачем. MIC повідомлення є простішою версією цифрового підпису, яка не потребує затрат ресурсів на проведення обчислень та підписання хешу повідомлення, натомість обмежується простим хешуванням повідомлення разом з спільним секретним кодом, який дві сторони з'єднання мають тримати в секреті. Так наприклад в LoRaWAN використовується алгоритм AES-CMAC, який використовує 128-бітний блоковий шифр AES [22] у режимі коду автентифікації повідомлень на основі шифру. MIC у LoRaWAN обчислюється на основі таких даних, як зміст самого повідомлення, ідентифікатор пристрою, лічильник кадрів або тег, та спеціальний ключ NwkSKey, за допомогою якого і відбувається розрахунок. Всі ці компоненти поєднуються і пропускаються через AES-CMAC, результатом чого є 128-бітне значення MIC. Однак для економії пропускної здатності в LoRaWAN використовується тільки перші 4 байти цього значення. Хоча 4 байти MIC здаються короткими, вони забезпечують достатній рівень захисту, враховуючи обмеження на пропускну здатність технології LoRa і енергозатрати. 32-бітний MIC дає 2^{32} можливих комбінацій, що є прийнятним компромісом між безпекою та ефективністю для більшості застосувань у

LoRaWAN, де головною метою є виявлення випадкових спотворень у повідомленні, для уникнення випадкової обробки неправильної команди чи отримання хибних показників. Також, у випадку LoRaWAN, додаткова захищеність досягається ще й за рахунок розділення ключів на ключ для шифрує повідомлення і ключ для обраховує MIC. Таким чином навіть при скомпрометованому ключі шифрування, зловмисник зможе тільки читати повідомлення, але не змінювати їх. Схожа система перевірки повідомлень має бути імпліментована і в децентралізованій peer-to-peer мережі. Але при цьому, ще однією важливою вимогою стає забезпечення наявності спільного секретного ключа для симетричного шифрування та можливості створення MIC на обох пристроях.

Перехід на симетричне шифрування вирішує одразу дві проблеми. Важливим фактом є те, що для пристроїв з незначними обчислювальними ресурсами та обмеженим живленням, критично важливо витрачати якомога менше енергії на шифрування та дешифрування повідомлень. Так як алгоритми асиметричного шифрування є значно складнішими та потребують більших обчислювальних потужностей, порівняно з симетричними алгоритмами, то є сенс використовувати симетричний алгоритм шифрування, наприклад AES-128 або AES-192, як основний алгоритм для обміну даними, але тільки після встановлення безпечного з'єднання за допомогою процесу автентифікації та узгодження сесійних ключів і спільного секрету за допомогою асиметричного алгоритму шифрування. Слід зазначити, що в цей спосіб можна налаштувати два сесійних ключі: один для шифрування самих повідомлень, а другий для створення коду перевірки цілісності повідомлення MIC. Таким чином, підхід встановлення з’єднання зі зміною типу шифрування, по-перше забезпечує надійний і добре зашифрований канал зв’язку для створення та обміну всією необхідною інформацією для створення сесії, а по-друге, після успішного створення сесії прибирає зайве навантаження на пристрій, що пов’язане з обрахунками для асиметричного шифрування.

Останньою вимогою до peer-to-peer мережі є реалізація протоколу «ACK» [23], який забезпечує гарантію доставки повідомлень між пристроями. Щоб забезпечити збереження енергоефективності та зменшення кількості повідомлень між пристроями, до структури повідомлення має бути додане поле «ACK» розміром в один біт, яке буде необов'язковим. Це значно зменшить навантаження на мережу і дасть користувачам можливість обирати, які повідомлення потребують сповіщення про успішну доставку, а які ні.

Сама реалізація протоколу сповіщення складається з декількох пунктів. Спершу, коли пристрій *A* хоче передати критично важливе повідомлення пристрою *B* і отримати підтвердження отримання, він відправляє повідомлення з прапором «ACK» у заголовку позначеним як 1. Цей прапор сигналізує пристрою *B*, що він має надіслати відповідь-підтвердження. Окрім прапору на запит підтвердження, в заголовку повідомлення також мають бути такі важливі поля, як «ID повідомлення», в ролі якого може виступати nonce або тег повідомлення, та поле часу очікування на ACK. Після передачі повідомлення пристрій *A* відкриває приймальне вікно протягом зазначеного часу, щоб отримати ACK від пристрою *B*. Це вікно повинне бути достатньо коротким, щоб мінімізувати витрати енергії, але при цьому забезпечити достатній час для доставки ACK повідомлення.

Наступним кроком пристрій *B*, отримавши і перевіривши повідомлення з вимогою про сповіщення про отримання, формує і надсилає ACK-повідомлення, при цьому перевіряючи час, наданий пристроєм *A* на очікування, щоб зрозуміти доцільність такої передачі. Повідомлення ACK має включати в себе поле «ID отриманого повідомлення», а також поле розміром в один біт, що буде прапором успішного отримання повідомлення.

Останнім кроком, пристрій *A* отримує ACK від пристрою *B* та перевіряє ідентифікатор повідомлення і код цілісності. Якщо всі перевірки пройдено успішно, *A* вважає, що повідомлення доставлене, і припиняє будь-які повторні спроби передачі. У випадку якщо ж пристрій *A* не отримує ACK протягом вказаного часу, він повторно відправляє повідомлення з тим самим

ідентифікатором, але збільшуючи свій лічильник зроблених спроб. Повторення може відбуватись певну кількість разів або з поступовим збільшенням інтервалів очікування для зменшення навантаження на мережу. Після досягнення максимального числа повторів без отримання повідомлення ACK, пристрій *A* вважає передачу невдалою та повідомляє про це користувача. Це дозволяє уникнути нескінченних повторів і надмірного навантаження на мережу.

Варто зазначити, що реалізація протоколу підтвердження отримання повідомлення є ефективним методом захисту від атак Block&Replay. Один зі сценаріїв такої атаки може виглядати наступним чином. Зловмисник перехоплює перше зашифроване повідомлення від відправника і блокує його, не надсилаючи отримувачу. Замість спроби розшифрувати дані, зловмисник зберігає повідомлення в оригінальному вигляді і чекає на повторну передачу цього ж повідомлення з новим одноразовим числом nonce. Після того, як відправник надсилає те ж повідомлення вдруге з іншим nonce, зловмисник знову перехоплює та зберігає його, не даючи цьому дійти до отримувача. Натомість зловмисник відправляє перше перехоплене повідомлення, що без протокола ACK для відправника виглядає як успішне отримання повідомлення з другої спроби. Таким чином у зловмисника залишається друге перехоплене повідомлення, яке він може надіслати отримувачу в будь-який момент, при цьому отримувач прийме це повідомлення як легітимне, бо для нього порядок параметра nonce в послідовності отриманих повідомлень не буде порушеним. Така вразливість може стати критичною, особливо у випадку такої махінації з управляючими повідомленнями.

Наявність протоколу ACK значно знижує ризик цієї атаки, адже якщо отримувач надіслав підтвердження про отримання першого повідомлення, відправник вважатиме його доставленим і не надсилатиме його повторно. Це ускладнює можливості для зловмисника зберігати різні копії одного й того ж повідомлення з різними значеннями nonce.

3.2. Розробка варіанту алгоритму створення шифрованої комунікації між пристроями в peer-to-peer мережі на основі технології LoRa

В цьому розділі буде детально описана реалізація технології створення та забезпечення шифрованого з’єднання між пристроями в peer-to-peer мережі на основі технології LoRa. Основний виклик при розробці такого протоколу полягає у тому, що LoRa-мережі характеризуються низькою пропускною здатністю та обмеженими як обчислювальними, так і енергетичними ресурсами пристроїв. Тому захист має бути реалізований максимально ефективно, використовуючи мінімальну кількість додаткових даних. Запропонований протокол забезпечує такі критично важливий функціонал, як шифрування повідомлень, автентифікацію пристроїв, захист від Replay та Block&Replay атак, підтвердження доставки, а також перевірку цілісності повідомлень за допомогою коду MIC.

Першим етапом алгоритму є «Підготовка для приєднання до мережі». На цьому етапі пристрій, який ще не приєднувався до мережі і буде робити це в перший раз, генерує для себе два ключі, приватний та публічний, використовуючи при цьому стандарт NIST P-256 [24] з асиметричної криптографії на еліптичних кривих, як було описано в розділі 2.1. Публічний ключ використовується в ролі ідентифікатора пристрою в мережі, в той час як приватний зберігається в таємниці. Варто зазначити, як було розглянуто в розділі 2.1, криптографія на еліптичних кривих хоч і є ефективнішим варіантом порівняно з алгоритмами RSA та Diffie-Hellman, але теоретично являється вразлива до атак за допомогою квантових комп’ютерів, тому як більш безпечний, але й більш затратний варіант, може бути використаний алгоритм NTRU, по стандарту IEEE 1363.1 [25]. Також, для зменшення часу в обох випадках, обрахування ключів може бути зроблено за допомогою сторонніх і більш потужні пристроїв, після чого обраховані ключі будуть просто збережено на пристрій, що ініціюватиме підключення до мережі.

Ще одним важливим моментом на цьому етапі є збереження публічних

ключів пристроїв, до яких користувач даного пристрою вже має довіру і з якими він планує налаштовувати з'єднання в першу чергу. Ці збережені ключі є необхідним елементом при наступних перевірках автентифікації інших пристроїв за допомогою системи Web of Trust. Також, на цьому етапі є можливим обмін сертифікатами довіри з перевіреними пристроями, щоб зробити процес автентифікації пристрою ще ефективнішим.

Другий етап – це встановлення з'єднання з іншим пристроєм, тобто підключення до мережі (рис. 3.1).

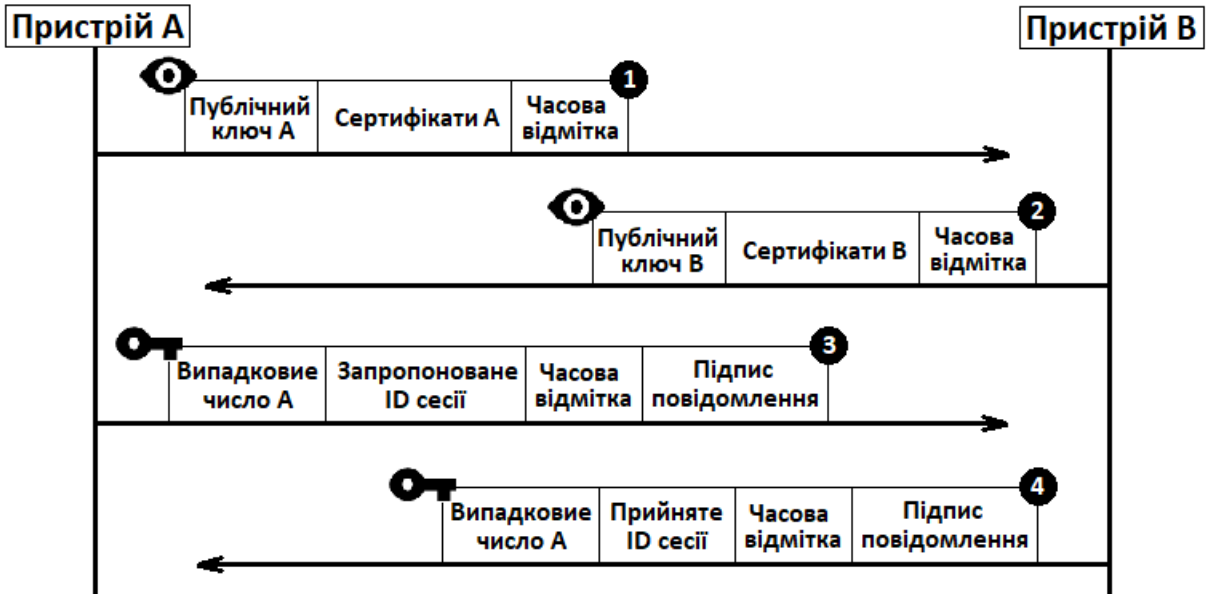

Рис. 3.1. Схема створення сесії між двома пристроями

На цьому етапі пристрій *A*, який ініціюючи з'єднання, надсилає пристрою *B* повідомлення, в якому він вказує свій публічний ключ, список сертифікатів, які йому були надані іншими пристроями щоб підтвердити свою ідентичність, та часову відмітку, щоб запобігти можливі replay атаки. Слід зазначити, що ці данні відправляються у незашифрованому вигляді. Наступним кроком, коли пристрій *B* отримує повідомлення, він перевіряє, чи є серед сертифікатів пристрою *A* підписані тими, кому пристрій *B* довіряє. Якщо такий сертифікат є, то пристрій *B* перевіряє його дійсність збереженим в себе публічним ключом довіреного пристрою, і якщо перевірка пройшла успішно, то надсилає пристрою *A*

повідомлення зі своїм публічним ключем, сертифікатами і часовою відміткою. У випадку ж відсутності підходящого сертифікату, пристрій B може скористуватись ланцюжком довіри і попросити пристрої яким він довіряє перевірити валідність таких сертифікатів, або просто відхилити запит на з'єднання. Ефективність такого методу ланцюжка довіри зменшується з довжиною самого ланцюжка, тому максимальна довжина такого алгоритму може бути індивідуально налаштована в залежності конкретної мережі, пристрою та вимог до нього.

Після того як пристрій A отримав відповідь від пристрою B, він проробляє точно таку ж автентифікацію пристрою B за допомогою системи Web of Trust. Після успішної автентифікації пристрою B, починається етап створення сесії та переходу на симетричне шифрування. Для цього пристрій А надсилає, зашифроване за допомогою публічного ключа пристрою В, повідомлення, де вказує випадково згенероване число R_A розміром 4 байти, запропонований ідентифікаційний код сесії розміром 4 байти, а також часову відмітку та цифровий підпис цього повідомлення, який забезпечує не тільки перевірку цілісності, а й підтверджує, що числа в цьому повідомлення були обрані саме пристроєм A. При цьому, запропонований ID сесії має бути таким, щоб будь-яке значення в межах від запропонованого і більше задовольняло пристрій А і не спричиняло колізій з іншими номерами сесій, які він може мати в цей момент.

По отриманню цього повідомлення, пристрій B робить перевірку цілісності та автентичності повідомлення з допомогою підпису пристрою A, якщо ця перевірка вдала, то пристрій B формує таке ж повідомлення зі своїм випадково згенерованим числом R_B, обраним ідентифікатором сесії, часовою відміткою та власним цифровим підписом цього повідомлення. При чому він може погодитись прийняти запропоноване пристроєм A значення ідентифікатору сесії, якщо воно підходить для нього і не викликає колізій. Якщо ж таки колізії виникають і запропоноване число не може бути використаним, тоді пристрій B обирає наступне з доступних, що йдуть після запропонованого. Такий підхід значно зменшує кількість повідомлень, потрібних для встановленні сесії.

По отриманню повідомлення, пристрій A перевіряє валідність повідомлення, використовуючи цифровий підпис повідомлення та публічний ключ пристрою B. Після чого обидва пристрої використовують отримані випадкові значення один одного R_B і R_A, номер сесії, та згідно з вимогами стандарту NIST SP 800-108 для створення спільного секрету на основі вхідних даних та з використанням алгоритму AES в режимі псевдовипадкової функції, незалежно генерують два сесійних ключі: перший – MsgKey для шифрування повідомлень, і другий – IntKey для обрахування коду перевірки цілісності повідомлень. Таке розділення функціоналу гарантуватиме цілісність повідомлень навіть при скомпрометованому MsgKey. Після успішного обрахування сесійних ключів, пристрої починають використовувати отримані ключі для захищеного спілкування. У випадку ж невдалого встановлення сесії, пристрої можуть повернутись на крок три та повторити кроки встановлення сесії, використовуючи нові випадкові числа R_B і R_A та новий номер сесії.

Після успішного встановлення сесії, пристрої переходять на третій етап з використанням симетричного алгоритму шифрування на основі ключа MsgKey для захищеної комунікації, який буде тривати до закінчення строку дії сесії. На цьому етапі структура повідомлень (таблиця 3.1) набуває стандартизований вигляд і складається з наступних полів: номер повідомлення, поле управління, дані, та код цілісності повідомлення MIC.

Таблиця 3.1.

Структура повідомлення на третьому етапі

№ повідомлення	Поле управління	Дані	MIC
3 байти	1 байт	0-255 байт	6 байти

В такій структурі, кожна зі сторін веде нумерацію повідомлень незалежно один від одного використовуючи поле «номер повідомлення» розміром 3 байти. Окрім інкрементації номеру вихідних повідомлень, пристрій має стежити за тим, щоб відхиляти будь-які повідомлення з номером меншим або рівним номеру

останнього отриманого повідомлення. Таким чином, поле нумерації повідомлень виконує дві функції. По-перше, воно захищає систему від replay-атак, а по-друге, забезпечує основу для контролю отримання повідомлень, так як це дозволяє легко ідентифікувати, на яке повідомлення було отримано сповіщення ACK.

Наступне поле, поле управління розміром в один байт, реалізує механізм сигналізування призначення повідомлення. Так, наприклад, перший біт відповідає за потребу в сповіщенні про отримання даного повідомлення. Положення першого біта в значені 0, означає що сповіщення про тримання не потрібно, але якщо ж значення першого біта дорівнює 1, то це дає знати, що пристрій має надіслати сповіщення. Другий біт поля управління відповідає за те, чи є дане повідомлення сповіщенням про отримання до іншого повідомлення. Решта бітів цього поля можуть бути використані для інших механізмів контрою призначення повідомлень, реалізованих в наступних версіях технології.

Після поля управління іде поле з корисним навантаженням, яке не має фіксованого розміру і може варіюватися від 0 до 255 байт. Слід зазначити, що при сповіщенні про отримання повідомлення, саме поле корисного навантаження використовується для вказання номеру отриманого повідомлення.

Останнім полем повідомлення є поле контролю цілісності повідомлення. Це поле розраховується згідно зі стандартом AES-CMAC, визначеним в документі NIST SP 800-38B [26], і окрім всіх вище зазначених полів повідомлення, бере до обрахунку також довжину даних цього повідомлення та публічний ключ отримувача. Далі, використовуючи алгоритм AES-CMAC та ключ IntKey, отримує результат обчислення розміром 16 байтів, але для поля MIC використовуються лише перші 6 байти. Таким чином, це значно зменшує можливість колізії в цьому полі та забезпечує надійнішу перевірку цілісності повідомлення, порівняно з MIC, що використовується в LoRaWAN.

В результаті, описана структура надає зручний, ефективний та захищений механізм для обміну та керування повідомленнями, який враховує особливості роботи малопотужних пристроїв в мережі з обмеженою пропускною здатністю.

ВИСНОВКИ

1. Проведено аналіз проблематики створення захищеного зв’язку в децентралізованих мережах на базі технології LoRa. Встановлено, що існуючі рішення, зокрема LoRaWAN, мають значні обмеження для використання у децентралізованих мережах через залежність від спеціалізованої архітектури мережі та попереднього узгодження ключів.

2. Розроблено алгоритм створення захищеного з’єднання, який охоплює всі етапи процесу: підготовку пристроїв до підключення, з’єднання, автентифікацію та узгодження сесійних ключів для симетричного шифрування.

3. Досліджено механізм автентифікації через систему Web of Trust, який забезпечує можливість пристроям самостійно перевіряти валідність сертифікатів та підтримувати високий рівень безпеки у децентралізованому середовищі.

4. Впроваджено механізми захисту від атак, включаючи replay-атаки, спотворення даних та атаки типу "людина посередині". Ці механізми включають використання цифрових підписів, часових міток, унікальної нумерації повідомлень та коду цілісності повідомлень.

5. Оптимізовано ефективність алгоритму для роботи у LoRa-мережах із низькою пропускною здатністю. Реалізовано використання компактних форматів повідомлень, скорочених MIC і оптимізованих криптографічних методів, що знижує навантаження на малопотужні пристрої, забезпечуючи енергоефективність та мінімізацію обчислювальних витрат.

6. Запропонований алгоритм може бути використаний для побудови захищених peer-to-peer мереж у різних галузях, включаючи IoT, промислову та військову автоматизацію. Його гнучкість дозволяє адаптуватися до потреб конкретної мережі та специфічних вимог до рівня безпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Frequency Plans by Country. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/ (дата звернення: 20.10.2024).

2. Jian Qin, Zhuoqun Li, Rui Wang, Li Li, Zhe Yu, Xun He & Ying Liu. Industrial Internet of Learning (IIoL): IIoT based pervasive knowledge network for LPWAN—concept, framework and case studies. January 05, 2021. URL: https://link.springer.com/article/10.1007/s42486-020-00050-2 (дата звернення: 20.10.2024).

3. Michael Alba. The Guide to Low-Power Wide Area Networks. April 30, 2018. URL: https://www.engineering.com/the-guide-to-low-power-wide-area-networks/ (дата звернення: 20.10.2024).

4. Alireza Maleki, Ha H. Nguyen, Ebrahim Bedeer, Robert Barton. Tutorial on Chirp Spread Spectrum Modulation for LoRaWAN: Basics and Key Advances. 25 July 2024 URL: https://ieeexplore.ieee.org/document/10609524 (дата звернення: 22.10.2024).

5. LoRa. Веб-сайт. URL: https://lora.readthedocs.io/en/latest/ (дата звернення: 22.10.2024).

6. EU863-870 MHz Band. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/eu868/ (дата звернення: 22.10.2024).

7. Sakshama Ghoslya. LoRa: Symbol Generation. URL: https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html?m=1 (дата звернення: 23.10.2024).

8. LoRa Physical Layer Packet Format. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/ (дата звернення: 23.10.2024).

9. Duty Cycle. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/duty-cycle/ (дата звернення: 23.10.2024).

10. Vincent Savaux, Christophe Delacourt, Patrick Savelli. Considering Sync Word and Header Error Rates for Performance Assessment in LoRa System. URL: https://hal.univ-lorraine.fr/BCOM_AC/hal-03200449v1 (дата звернення: 26.10.2024).

11. LoRaWAN. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/ (дата звернення: 26.10.2024).

12. End Device Activation. The things network. URL: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/ (дата звернення: 30.10.2024).

13. LoRaWAN™ SECURITY. FULL END–TO–END ENCRYPTION FOR IoT APPLICATION PROVIDERS. URL: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf (дата звернення: 30.10.2024).

14. November 2016. K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. URL: https://www.rfc-editor.org/rfc/rfc8017 (дата звернення: 03.11.2024).

15. June 1999. E. Rescorla. Diffie-Hellman Key Agreement Method URL: https://www.rfc-editor.org/rfc/rfc2631 (дата звернення: 03.11.2024).

16. February 2011. D. McGrew, K. Igoe, M. Salter. Fundamental Elliptic Curve Cryptography Algorithms. URL: https://www.rfc-editor.org/rfc/rfc6090 (дата звернення: 07.11.2024).

17. J. Hoffstein, J.Pipher, J. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. URL: https://www.ntru.org/f/hps98.pdf (дата звернення: 07.11.2024).

18. The ElGamal Public Key Encryption Algorithm. URL: https://homepages.math.uic.edu/~leon/mcs425-s08/handouts/el-gamal.pdf (дата звернення: 10.11.2024).

19. A. Gillis. X.509 certificate. URL:

https://www.techtarget.com/searchsecurity/definition/X509-certificate (дата звернення: 10.11.2024).

20. Blockchain. December 20, 2021. URL: https://www.nist.gov/blockchain (дата звернення: 13.11.2024).

21. What is Web of Trust? 02 May, 2024. URL: https://www.geeksforgeeks.org/what-is-web-of-trust/ (дата звернення: 23.10.2024).

22. National Institute of Standards and Technology. Advanced Encryption Standard (AES). URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf (дата звернення: 13.11.2024).

23. ACK in Networking. November 25, 2023 URL: https://networkencyclopedia.com/ack/ (дата звернення: 15.11.2024).

24. National Institute of Standards and Technology. Efficient and Secure Elliptic Curve Cryptography Implementation of Curve P-256 NIST P-256. URL: https://csrc.nist.gov/csrc/media/events/workshop-on-elliptic-curve-cryptography-standards/documents/papers/session6-adalier-mehmet.pdf (дата звернення: 15.11.2024).

25. 1363.1-2008 - IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. URL: https://ieeexplore.ieee.org/document/4800404 (дата звернення: 15.11.2024).

26. M. Dworkin. Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication. URL: https://csrc.nist.gov/pubs/sp/800/38/b/upd1/final (дата звернення: 16.11.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)