Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка захищеного протоколу безпеки для бездротового з'єднання»

Виконав: студент групи К20-1
Спеціальність 122 «Комп'ютерні науки»
Курбацька Є. С.
Керівник: к. ф.-м. н., доц. Лебідь О. Ю.
Рецензент: Спеціалізоване управління розробки та супроводження програмного забезпечення Департамент з питань цифрового розвитку, цифрових трансформацій та цифровізації ДМСУ
(місце роботи)
головний державний інспектор відділу розробки програмного забезпечення
(посада)
Бахтін О. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Курбацька Є. С. Розробка захищеного протоколу безпеки для бездротового з’єднання.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 “Комп’ютерні науки”. – Університет митної справи та фінансів, Дніпро, 2024.

Дана кваліфікаційна робота присвячена розробці захищеного протоколу безпеки для бездротового з’єднання. В якості бездротової технології для якої створюється захищений протокол використано технологію зв’язку Bluetooth Low Energy. Розглянута бездротова технологія, її актуальність та сфери застосування. Змодельовано можливі загрози для технології, проаналізовані існуючі методи захисту каналу передачі даних та їх проблеми. Було спроектовано можливий протокол безпеки для захисту з’єднання, а також, девайс, що демонструє можливість реалізацію описаного захисту. Отримані результати мають практичне значення, оскільки сприяють поліпшенню безпеки каналу передачі даних в технології бездротового з’єднання Bluetooth Low Energy та може бути використана як основа для створення подібного протоколу для інших технології бездротового з’єднання. Розроблений протокол є ефективним та дозволяє створити надійний канал передачі даних між девайсами, використовує сучасні криптографічні алгоритми, такі як протоколи безпечного обміну ключами по не захищеному каналу передачі даних та авторизація на основі сертифікатів, та є важливим кроком в підвищенні захисту девайсів в різних сферах.

Ключові слова: РОЗРОБКА, БЕЗПЕКА, ПРОТОКОЛ, КРИПТОГРАФІЯ, СЕРТИФІКАТИ, BLUETOOTH LOW ENERGY, ECDH, DHM.

ABSTRACT

Kurbatska E. S. Development of a secure security protocol for a wireless connection.

Qualification work for obtaining a bachelor’s degree in specialty 122 "Computer science". – University of Customs and Finance, Dnipro, 2024.

This qualification work is devoted to developing a secure security protocol for a wireless connection. Bluetooth Low Energy communication technology is used as a wireless technology for which a secure protocol is created. Wireless technology itself, its relevance, and areas of application are considered. Possible threats to the technology were modeled, and existing data channel protection methods and their problems were analyzed. A possible security protocol was designed to protect the connection, as well as a device that demonstrates the possibility of implementing the described protection. The obtained results are of practical importance, as they contribute to improving the security of the data transmission channel in the Bluetooth Low Energy wireless connection technology and can be used as a basis for creating a similar protocol for other wireless connection technologies. The developed protocol is effective and allows the creation of a reliable data transmission channel between devices, uses modern cryptographic algorithms, such as secure key exchange protocols over an unsecured data transmission channel and certificate-based authorization, and is an important step in increasing the protection of devices in various fields.

Keywords: DEVELOPMENT, SECURITY, PROTOCOL, CRYPTOGRAPHY, CERTIFICATES, BLUETOOTH LOW ENERGY, ECDH, DHM.

ЗМІСТ

ВСТУП

Актуальність дослідження. Розробка протоколів передачі даних для Bluetooth Low Energy (BLE) залишається актуальною, оскільки ця технологія вже поширена та продовжує активно поширюватися в різних галузях. Особливої популярності вона набува наступних пристроях: медичні пристрої, споживча електроніка, промислова автоматизація та інші. Із все більшою популярністю використання технології зростає і потенційна загроза безпеки. Основна причина, з якої розробка та удосконалення вже існуючої системи безпеки для технології Bluetooth Low Energy залишаються затребуваною є збільшення кількості загроз безпеці. У зв'язку з розширенням використання Bluetooth Low Energy, збільшується й кількість потенційних атак і загроз безпеці. Це включає в себе можливість підслуховування, підробку даних, атаки з переповнення буфера, а також атаки на основі вразливостей у самому протоколі. Крім того, з більшим обсягом конфіденційної і особистої інформації, яка передається через Bluetooth Low Energy, стає дедалі важливіше забезпечити надійний захист цих даних від несанкціонованого доступу.

Отже, розробка ефективного протоколу захисту для технології Bluetooth Low Energy залишається актуальною і важливою задачею для забезпечення конфіденційності, цілісності та доступності даних, що передаються через цей протокол.

Метою роботи є розробка протоколу, який підвищить безпеку передачі даних під час передачі даних за допомогою технології Bluetooth Low Energy.

Методи дослідження: метод теорії інформації, обробка та аналіз інформації, методи проектування та розробки програмного забезпечення.

У відповідності до поставленої мети в кваліфікаційній роботі ставились та вирішувались наступні завдання дослідження.

1. Дослідження предметної області.

2. Огляд технології Bluetooth Low Energy та можливих загроз для цієї технології.

3. Аналіз існуючих в технології Bluetooth Low Energy засобів захисту, їх переваги та недоліки.

4. Розробка власного засобу захисту для технології Bluetooth Low Energy.

5. Розробка алгоритму роботи та практичного девайсу для демонстрації роботи нового протоколу.

Об’єктом дослідження є процес попередження атак при користуванні Bluetooth Low Energy.

Предметом дослідження є програмний протокол для реалізації захисту даних під час їх передачі між девайсами.

Результати продемонстровані в кваліфікаційній роботі відповідають наступним результатам навчання освітньої програми 122 «Комп’ютерні науки»:

– ПР01. Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.;

– ПР06. Уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення.;

– ПР09. Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення.;

– ПР10. Проводити передпроектне обстеження предметної області, системний аналіз об’єкта проектування.;

– ПР13. Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.

Практичне значення одержаних результатів полягає у спрощенні та підвищенні рівня безпеки технології Bluetooth Low Energy.

Структура кваліфікаційної роботи: вступ, 3 розділи, висновки, перелік використаних джерел, 2 додатки. Обсяг роботи – 57 сторінок, робота містить 18 рисунків, перелік використаних джерел складається з 30 посилань, додатки розміщено на 24 сторінках.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про Bluetooth Low Energy

Bluetooth Low Energy (BLE) – це бездротова технологія зв’язку, розроблена для забезпечення можливості комунікації для девайсів, які потребують низького споживання електроживлення, оскільки вони спроектовані бути автономними. Технологія забезпечує можливість обміну даними на невеликі відстані зазвичай до 100 метрів, підтримує різні швидкості передачі даних (низькою є швидкість до 125 кбіт/с, високою є швидкість до 2 Мбіт/с), а також дозволяє працювати з короткими інтервалами передачі даних, це дозволяє швидко встановити зв’язок між девайсами і передати інформацію з малими затримками. Дана технологія ґрунтується на технології передачі даних за допомогою радіочастотного зв’язку. Кінцеві девайси використовують Технологію через спеціалізовані чипи Bluetooth зв’язку, що вбудовуються в дані девайси, такі чіпи окрім модуляції даних для передачі по радіоканалу також реалізують логіку технології, додають до інформації, яка повинна бути відправлена, заголовки для розпізнавання, створюють службові пакети інформації, а також мають реалізацію базових криптографічних протоколів. Додатково більшість сучасних чипів Bluetooth low Energy виконують функції центрального процесора, мають вбудовані оперативну і постійну пам’ять а також підтримують велику кількість різноманітних фізичних інтерфейсів для комунікації з різними модулями корисного навантаження, що створює можливість виконання девайсу на базі єдиного центрального чипа з кількома датчиками, що в свою чергу зменшує ціну кінцевих девайсів для продуктів яким достатньо подібного функціоналу.

Основними компонентами технології є:

1. Передавачі і приймачі. Кожен пристрій працює в режимі центрального або периферичного девайсу в залежності від призначення девайсу, але можлива динамічна зміна режиму. Центральним зазвичай виступає девайс користувача

який ініціює так керує зв’язком. Він може сканувати навколишній простір на наявність периферійних пристроїв і встановлювати з’єднання з ними. Центральним пристроєм може бути наприклад смартфон, або комп’ютер користувача. Периферійним пристроєм є девайс, який відповідає на запити центрального пристрою. Зазвичай це малопотужні пристрої з обмеженими ресурсами, такі як наприклад датчики, фітнес-годинники, медичні сенсори, їх головним завданням в контексті передачі інформації з використанням цього протоколу є відповідь на запити центрального девайсу, або транслювання змін в своїх характеристиках. Основною причиною розділення девайсів на центральні та периферійні полягає в оптимізації споживання енергії. В той час, як центральні девайси можуть мати значні запаси енергії або достатньо часте обслуговування, периферійні девайси, як правило, є малопотужний пристроями без значних запасів енергії. Тому більшу частину часу останні перебувають в режимі очікування, активуючись лише при необхідності передачі даних, це дозволяє їм працювати автономно протягом довгих періодів часу.

2. Рекламні пакети і пакети даних. Периферійні пристрої транслюють короткі повідомлення, відомі як рекламні пакети, для привернення уваги центральних девайсів. Ці пакети містять інформацію про девайс і його можливостію Основною причиною існування подібних пакетів є об’явлення про існування девайсу і його можливостей. Також ці пакети можуть бути відправлені у відповідь на активне сканування центральним девайсом. При необхідності передачі даних створення з’єднання та будь якої іншої комунікації між девайсів окрім рекламування своїх можливостей використовуються пакети даних, через них відправляються різного роду запроси і команди між девайсами, а також, цей тип пакетів використовується для не опосередкованій передачі даних між девайсами.

3. Сервіси і характеристики. Характеристики – це тип даних, який використовується для комунікації між девайсами. Кожна характеристика має свій унікальний UUID (Universally Unique Identifier) і може бути доступною для читання, запису, або використовуватися як нотифікація, через таку

характеристику відображаються зміни в значеннях. Сервіси в собі об’єднують характеристики в логічні групи. Кожен сервіс також має свій UUID і може містити як одну так і декілька характеристик.

4. Профілі. Використовуються для визначення конкретного використання Bluetooth Low Energy девайсу. Існує велика кількість профілів для різного призначення. Наприклад, профіль “Heart Rate Monitor” описує, як використовувати канал даних для передачі інформації про вимірювання серцевого ритму що використовується в фитнес-трекерах, смарт годинниках та інших пристроях для вимірювання фізичного здоров’я. Профіль “Battery Service” описує використання каналу для передачі інформації про рівень заряду батареї. Використовується в різних девайсах які працюють на батареях. А “Proximity Profile” використовується для визначення відстані між двома пристроями. Це може бути корисно для створення систем сигналізації на основі відстані між девайсами.

5. Характеристики безпеки. Протокол має вбудовані механізми безпеки для створення безпечного каналу зв’язку. Будучи заснованим на технології класичного Bluetooth з відхиленням в енергоефективність технологія Bluetooth Low Energy має пом’якшення до процедур і алгоритмів узгодження захисту каналу передачі даних. Одним з нововведень є можливість повного відключення захисту для каналу даних, подібну можливість зазвичай використовують різні датчики основною метою яких є повідомлення інших пристроїв про зміну своїх характеристик зазвичай зумовлену зміною в навколишньому середовищі, при цьому для таких девайсів бажано залишатися в робочому стані якомога довше, а самі дані можуть не бути секретними. Для такого роду девайсів не потрібна можливість створювати захищений канал з’єднання. Відсутність подібної потреби зменшує необхідну кількість переданих між девайсами пакетів що напряму зменшує електроспоживання пристрою. І, хоча, для деяких призначень безпечний канал не потрібен, все ж, наявність механізмів захисту в протоколі робить безпечним передачу даних для профілів, яким важливо мати шифрований зв’язок. Проте наявність пом’якшень і специфіка девайсів для яких було

розроблено протокол створює проблеми для безпечного узгодження захищеного каналу зв’язку за допомогою лише стандартних механізмів захисту.

Оскільки технологія Bluetooth Low Energy була в першу чергу створена для оптимізації споживання енергії девайсами які його використовують найчастіше вона зустрічається в продуктах і системах, де необхідна автономність і швидкий обмін невеликими даними. Для прикладу, такими системами можуть бути: системи контролю доступу, медичні датчики, трекери вантажів та інше. Також однією з популярних сфер, де використовується технологія Bluetooth Low Energy є фітнес-трекери та смарт годинники, оскільки можливість не обслуговувати, не підзаряджати в даному випадку свої девайси, а як слідство подовжити безперервне користування девайсом є дуже привабливою можливістю для користувача що стало одним з головних факторів для кінцевих споживачів. Ще однією сферою використання є системи автоматизації, оскільки можливість швидкого обміну інформацією між багатьма девайсами може бути використана в системах по типу рою дронів, де переміщення дронів коригуються без участі оператора або центральної системи, базуючись на інформації про маршрути оточуючих дронів.

1.2 Огляд можливих загроз технології Bluetooth Low Energy

Технологія Bluetooth Low Energy використовується в багатьох продуктах, де необхідно мати захищений канал передачі даних. При цьому, для деяких продуктів безпечна передача даних є критично важливою. Таким чином, розкриття інформації, яка передається, наприклад, фітнес-трекером, може створити ризики для конфіденційності користувача, а перехоплення даних, які були відправлені системі контролю доступу можуть бути використані для неправомірного доступу. До того ж, перехоплення і підміна прошивки будь-якого девайсу створює риски повної компрометації цього девайсу. Для вказаних вище систем безпека каналу комунікації є надзвичайно важливою. Проте, для таких девайсів як медичні датчики або систем автоматизації безпека каналу

передачі даних є критично важливою, оскільки підміна медичних показників може становити загрозу життю і здоров'ю пацієнта, а підміна даних в системі автоматизації, наприклад, на основі автономних дронів на складі може призвести до аварій.

Описані вище приклади втручання атакуючої сторони в канал комунікації можуть бути реалізовані за допомогою наступних атак проти технології Bluetooth Low Energy:

1. Перехоплення трафіку (eavesdropping). Дана атака базується на принципі передачі радіосигналу. Оскільки дані передаються в спільне середовище не лише адресат може отримати ці дані а й усі інші сторони які мають можливість приймати радіосигнали на тій же частоті, окрім можливості перехоплення даний фактор також створює перешкоди для одночасного використання багатьох девайсів, тому зазвичай чіпи, які реалізують комунікацію з використанням технології Bluetooth Low Energy мають алгоритми фільтрації та перемикання частот для відсіювання перешкод у вигляді даних переданих іншими девайсами. Проте атакуючий що має має можливість перехоплювати усі пакети даних, що відправляються у спільний простір. На рисунку 1.1 наведено алгоритм атаки під назвою «Перехоплення трафіку».

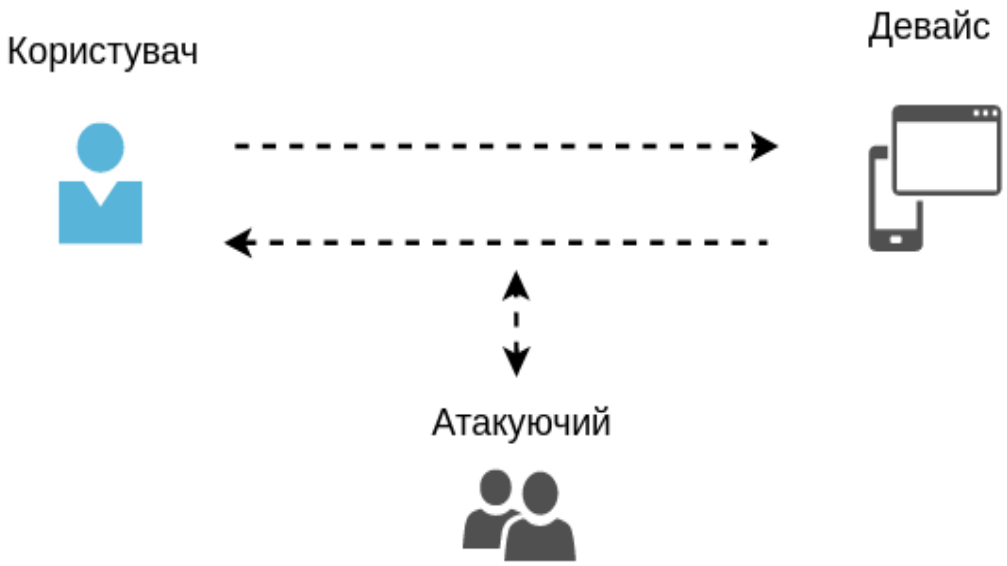

Рисунок 1.1 – Алгоритм атаки «Перехоплення трафіку»

В такому разі при подальшому аналізі атакуючий може віднайти інформацію яка була передана конкретним каналом передачі даних. Щоб запобігти такій атаці необхідно захистити канал за допомогою шифрування з використанням секретів які не можуть бути відомими для атакуючого.

2. Атака на відмову в обслуговуванні (DoS). Ця атака базується на обмежених ресурсах девайсу проти якого ця атака здійснюється. Оскільки щоб мати можливість комунікувати з іншими пристроями, девайс повинен приймати та обробляти пакети що надходять до нього. Зазвичай дізнатися чи справді отримані дані становлять інтерес для пристрою можна лише обробивши пакет даних як мінімум службову інформацію про нього. Відправляючи велику кількість запитів до пристрою, що можуть навіть не мати сенсу можливо вичерпати усі ресурси пристрою жертви, що зробить неможливим подальше отримання даних навіть від легітимних сторін.

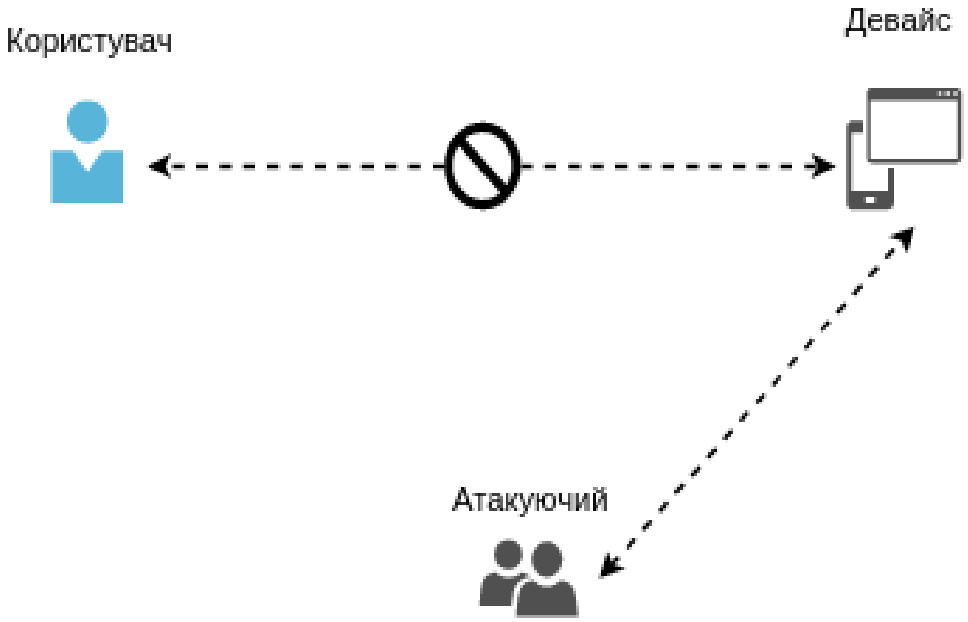

Рисунок 1.2 – Алгоритм атаки “Відмова в обслуговуванні”

На рисунку 1.2 наведено алгоритм атаки під назвою “DoS”. Захиститися від цієї атаки дуже складно і навіть інколи немає можливості завчасно знати чи є пакет з даними корисним чи шкідливим. Але головним моментом пом’якшення

цієї проблеми є створення бізнес логіки з урахуванням можливості подібної атаки. Алгоритм обробки пакетів має витрачати мінімум часу на пакети які є шкідливими, тим самим роблячи необхідним збільшення можливостей атакуючого до відправки шкідливих пакетів щоб атака могла бути проведена вдало.

3. Атака "Man-in-the-Middle" (MitM). Ця атака схожа на атаку підслуховування і базується на тому ж самому принципу, але для її проведення необхідно також мати можливість передавати свої дані під ідентифікаторами пристроїв жертв, або мати можливість за допомогою створити два девайси клони, що будуть незалежно комунікувати з обома сторонами передаючі між собою і віддаленими сторонами усі пакети даних с корисним навантаженням, маючи при цьому можливість для прослуховування або підробки даних. На рисунку 1.4 наведено алгоритм атаки під назвою “Man-in-the-Middle”. Подібна атака дозволяє підмінювати будь-які дані що передаються між девайсами. Окрім підміни самих даних бізнес логіки, якими можуть бути інформація з датчикив або інформація про внутрішній стан та інше, також подібна атака дозволяє підмінити інформацію що використовується для узгодження захищеного каналу зв’язку що дозволяє примусити девайси створити подібний канал саме з девайсом посередником створивши можливість для останнього скомпрометувати зашифрований канал. Для захисту від подібної атаки потрібно створювати захищене з’єднання на базі або з використанням секретів недоступних для атакуючої сторони, а також криптопротоколів автентифікації сторін комунікації.

Окремим особливим випадком виконання подібної атаки є подовження каналу зв’язку між легітимними девайсами (рисунок 1.3). Дана атака націлена на проходження авторизації в системах контролю доступу які надають цей доступ коли легітимний девайс знаходиться поруч, отримуючи рекламні пакети від такого девайсу система контролю доступу ініціалізує підключення і обмін інформацією щоб авторизувати девайс і його користувача та надати доступ. Для виконання такої атаки необхідно два девайси що будуть виконувати ролі

посередника і комунікувати між собою, а також пересилати пакети корисного навантаження через мережу інтернет. Цей окремий випадок атаки посередника не намагається прочитати чи змінити дані що передаються через канал передачі даних вони лише пересилають ці дані подовжуючи з’єднання.

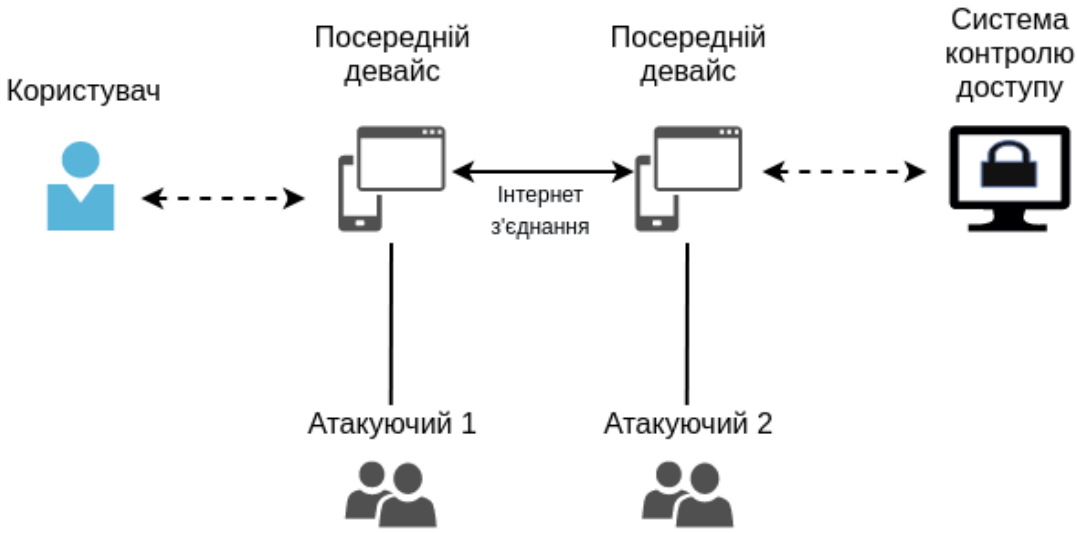

Рисунок 1.3 – Алгоритм атаки “Man-in-the-Middle”

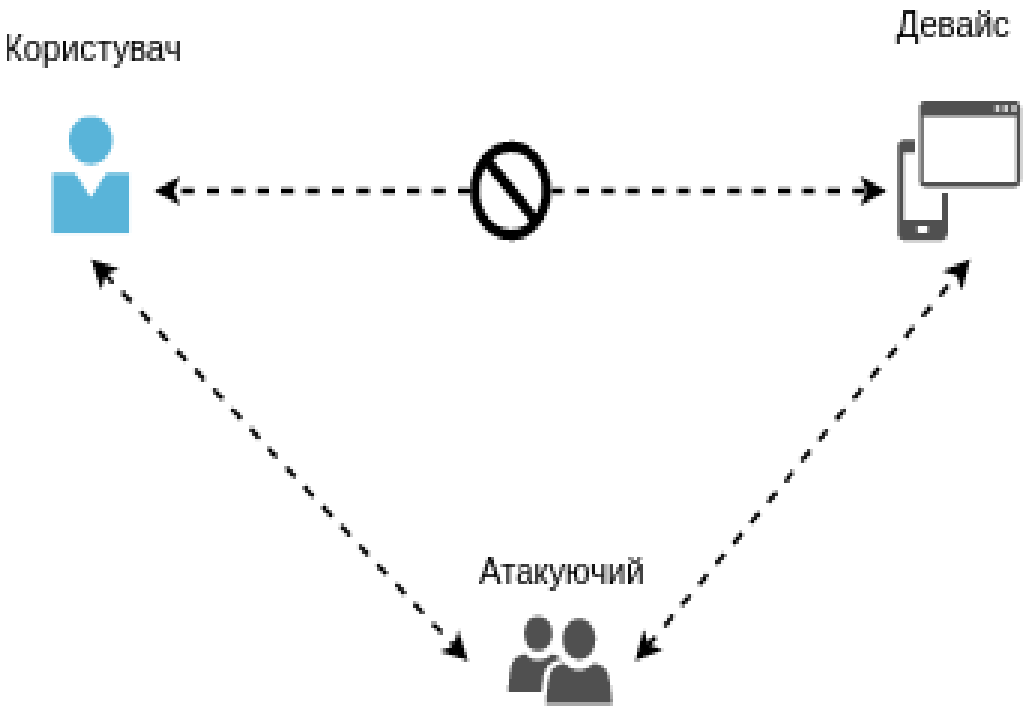

Рисунок 1.4 – Алгоритм атаки “Man-in-the-Middle”

Для реалізації один з девайсів посередників має знаходитися біля системи контролю доступу, а другий безпосередньо біля ідентифікатора (ключа) до цієї системи пересилаючи пакети корисного навантаження система з двох посередніх девайсів дозволить атакуючому отримати доступ.

4. Викрадення ідентифікаторів (Spoofing). Ця атака також як і попередні базується на тому що пакети даних посилаються пристроями в суспільне середовище де вони можуть бути підслухані. Сенс цієї атаки полягає в можливості прослуховування каналу передачі даних виявлення ідентифікаторів і клонування пристрою на основі цих перехоплених ідентифікаторів. На рисунку 1.5 наведено алгоритм атаки під назвою “Викрадення ідентифікаторів”.

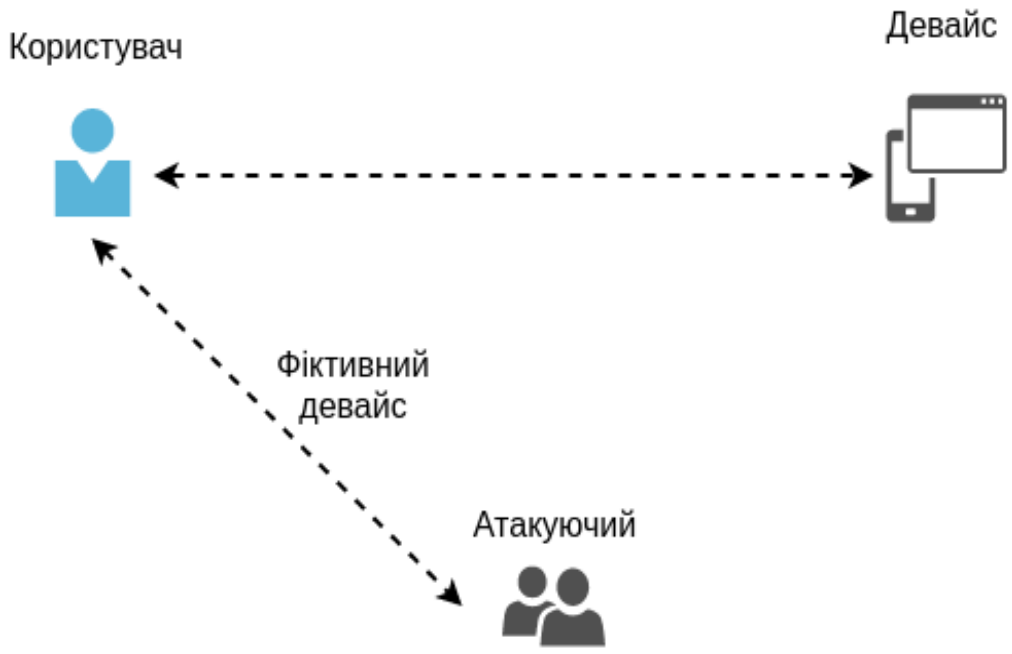

Рисунок 1.5 – Алгоритм атаки “Викрадення ідентифікаторів”

Подібна атака може бути частиною більш складних атак таких як “Man-in-the-Middle”, або бути використана для отримання неправомірного доступу. Як приклад останнього варіанту ця атака може бути використана для клонування пристрою автентифікації користувача в системі контролю доступу, якщо надання доступу відбувається лише за рахунок звірення ідентифікатора з базою відомих, новий клонований девайс матиме можливості такого самого доступу як

і девайс з якого він був клонований. Для захисту від подібної атаки потрібно щоб бізнес логіка подібних систем окрім перевірки ідентифікаторів девайсу проводила авторизацію на основі інших більш довірених факторів.

5. Атака дійника. Ця атака базується на створенні фейкового девайсу що представляє себе як реальний, і паралельно блокує реальний девайс (рисунок 1.6).

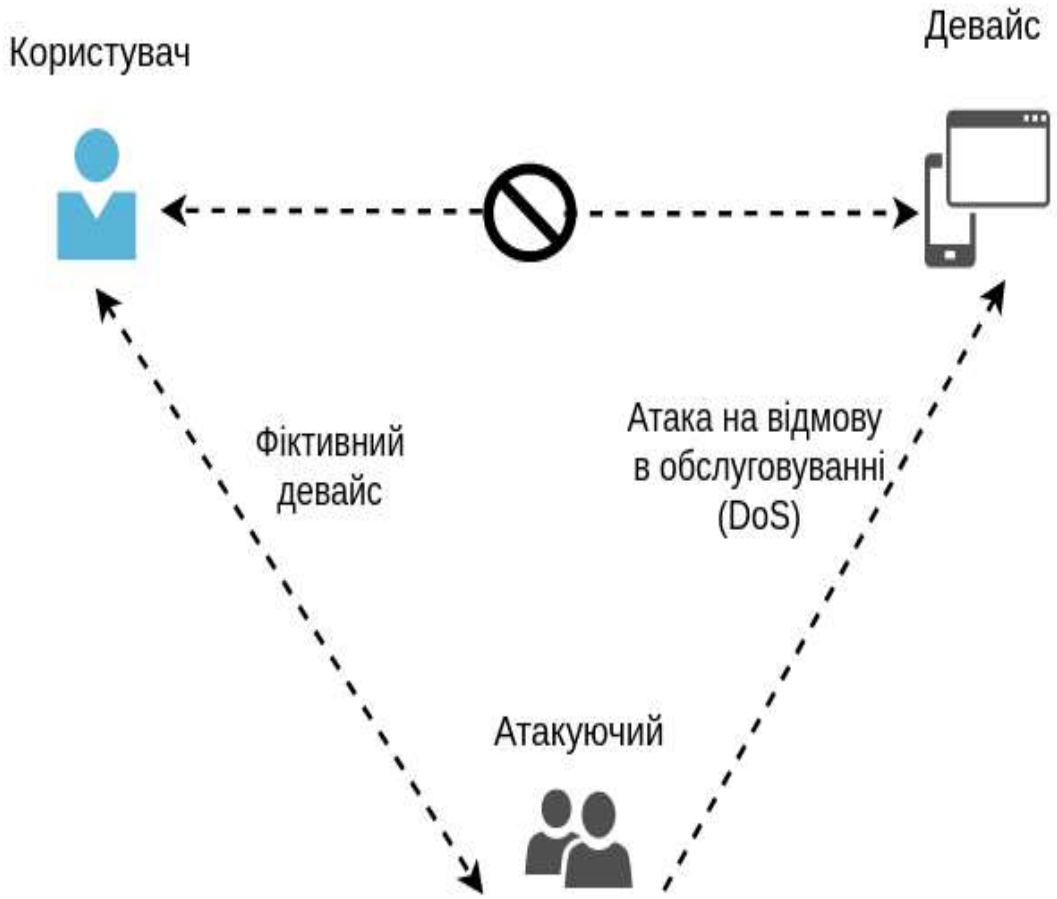

Рисунок 1.6 – Алгоритм атаки дійника

Ціль подібної атаки змусити одну з сторін підключитися до фіктивного девайсу щоб таким чином отримати інформацію від цього девайсу або користувача що його використовує. Частиною цієї атаки має бути атака відмови в обслуговуванні на один з девайсів щоб унеможливити підключення атакуємої сторони до реального девайсу. Також для того щоб зробити таку атаку можливою необхідно щоб фейковий девайс міг комунікувати з реальним використовуючи протокол та інші особливості що має реальний девайс що

робить необхідним використання такого ж самого девайсу що і блокується атакою на відмову в обслуговуванні, або реверс інжинірингом подібного девайсу. Захиститися від такої атаки можливо якщо протокол узгодження каналу передачі даних буде автентифіковувати обидві сторони. При цьому необхідно щоб дані що використовуються для автентифікації сторін були завірені спільним секретом або центром сертифікації а також повинні бути унікальними саме для цієї пари девайсів, оскільки якщо буде можливо підключити декілька девайсів один до одного без визначення конкретних факторів то подібного девайсу або викрадення секретів що використовуються в групі девайсів буде достатнім для проведення подібної атаки.

1.3 Постановка завдання

Оглянута технологія має свої переваги що робить її гарним варіантом для використання в багатьох схемах, а також наявність на ринку великої кількості апаратних компонентів що реалізують технологію і мають можливість інтеграції свого коду робить цю технологію популярною. Проте недоліки у вигляді можливих атак створює проблеми для використання в сферах де захист даних є дуже важливим. Тому метою цієї роботи є створення протоколу що може забезпечити надійний захист від існуючих атак на технологію. Згідно з поставленою метою в кваліфікаційній роботі ставилися та вирішувалися наступні завдання дослідження:

1. Проаналізувати методи захисту що існують в технології та визначити їх проблеми.

2. Спроектувати новий протокол що не буде підвержений описаним атакам.

3. Розробити девайс для практичної демонстрації роботи протоколу.

1.4 Висновки до першого розділу

В той час як захистити девайс від атак на відмову в обслуговуванні може бути доволі складно, оскільки атакуючий може мати набагато більше ресурсів ніж пристрій, що атакується. В той же час одним з найлегших варіантів захисту від інших атак що були наведені вище може бути створення безпечного каналу передачі даних з авторизацією сторін з’єднання. Такий протокол має захищати дані що передаються через нього за допомогою шифрування, яке у свій час має бути створене з використанням спільних секретів які були узгоджені за допомогою алгоритму що не дозволить атакуючому дізнатися спільні секрети. Передача самих даних в такому протоколі має бути захищена використанням алгоритму шифрування що дозволить перевірити автентичність отриманих даних, а також мати можливість перевірити автентичність додаткових не зашифрованих даних що можуть використовуватися для захисту від просунутих атак і часткового полегшення впливу атаки відмови в обслуговуванні.

При використанні протоколу що забезпечує авторизацію сторін на базі спільного фактору починаючи з самого початку з’єднання, використання протоколу узгодження спільного секрету по незахищеному каналу, має можливість додати не зашифровані дані, що допоможуть зменшити витрати на обробку пакетів в деяких випадках, та використовує надійне шифрування можливо захиститися від більшості атак. Наявність такого протоколу допоможе покращити безпеку в багатьох сферах що використовують бездротову передачу даних на основі оглянутої технології.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ В ТЕХНОЛОГІЇ BLUETOOTH LOW ENERGY ЗАСОБІВ ЗАХИСТУ

2.1 Огляд сучасних методів захисту в технології Bluetooth Low Energy

Технологія передачі даних Bluetooth Low Energy створювався для більш ефективного використання електроживлення в порівнянні з класичним Bluetooth, новий протокол хоча і заснований на класичній версії технології проте має пом’якшення в стандарті захисту, оскільки для деяких випадків стандарти безпеки які були введені в класичній версії не потрібні, при цьому деякі механізми захисту в класичному Bluetooth не є можливим для девайсів яким потрібна більша автономія. Введення подібних пом’якшення зменшило необхідну кількість ітерації передачі даних, що в свою чергу зменшило електроживлення девайсів, а також можливість відновити з’єднання через деякий час без потреби тримати його активним разом зі зменшенням електроспоживання дозволяє передавати дані з невеликою затримкою.

Технологія Bluetooth Low Energy вводить режими безпеки які визначають, який захист для каналу даних потрібен для девайсу, визначає потрібний режим саме девайс до якого підключаються, а саме периферійний пристрій. Можливими варіантами режимів безпеки є:

1. “No security” – це режим, в якому немає а ні аутентифікації, а ні шифрування. Такий режим зазвичай використовується в продуктах які не передають важливої інформації, такими можуть бути наприклад сенсори освітлення або датчики температури. Проте насправді зважаючи на те що подібні девайси є частиною більшої системи можливість підмінити дані з них може вплинути на функціонування інших систем, прикладом такої ситуації може бути увімкнення опалення через низьку температуру в приміщенні, якщо система опалення не має власного датчику температури, а лише довіряє даним які приходять з подібного девайсу, не маючи захисту для каналу даних атакуючий може увімкнути опалення в приміщенні що потенційно може призвести до

псування майна яке там знаходиться, особливо якщо рішення про вимкнення опалення також приймається на основі даних температури з Bluetooth Low Energy датчику.

Даний режим безпеки не має механізмів захисту переде атаками тож: Трафік може бути прослуханий використовуючи атаку прослуховування. Трафік може бути прослуханий через виконання атаки Man-in-the-Middle. Девайс може бути підроблено використовуючи атаку двійника. І ідентифікатори девайсу можуть бути скопійовані за допомогою атаки копіювання ідентифікаторів. Цей режим також не може убезпечити від атаки на відмову в обслуговуванні.

2. “Just works” – це режим, в якому передбачено шифрування, але не передбачено автентифікації. В цьому режимі спільний секрет для шифрування каналу даних узгоджується через небезпечний канал даних. І хоча дані зашифровані з використанням спільного секрету отриманого таким способом відновити буде складно, проте виконання “Man-in-the-Middle” атаки з підміною параметрів для вираховування спільного секрету змусить пристрої створити захищений канал даних з посереднім пристроєм що у свою чергу створює можливість для компрометації такого каналу. Також криптографічний протокол використаний в цьому режимі виявився недостатньо складним що дозволяє при деяких умовах вирахувати спільний секрет іншим девайсом без необхідності проводити “Man-in-the-Middle” атаку.

Даний режим безпеки має базовий захист після узгодження спільного секрету дані що передаються каналом буде зашифровано, а отже атака прослуховування не дасть результатів. Проте оскільки канал попередньо не автентифікується це створює можливість використання атаки Man-in-the-Middle що дозволяє примусити девайси на утворення каналу зв’язку через посередній пристрій що в свою чергу дозволить підміняти дані які проходять цим каналом. Також цей режим безпеки немає механізму захисту від викрадення ідентифікаторів чи захисту від атаки двійника оскільки відсутність авторизації робить неможливим впевнитися що девайс з яким відбувається комунікація саме

той що треба, а створенню спільного секрету нічого не перешкоджає. Також цей режим не може забезпечити безпеку від атаки на відмову в обслуговуванні.

3. “Authenticated Pairing with Encryption” – це режим, в якому передбачено шифрування, але для створення каналу необхідно авторизувати з’єднання. Такий режим потребує залучення людини задля введення додаткового секрету або виконання “Out of Band” процедури під час якої використовується додатковий фізичний канал передачі даних. Прикладом може бути з’єднання з медичним приладом який може передавати показники на телефон пацієнта. В такому разі для узгодження каналу передачі даних може потребуватися ввести один і той же пароль на обох девайсах. Оскільки атакуючий в теорії не може отримати цей пароль не будучи поміченим або взагалі пароль для нього не доступний цей пароль буде використано для створення спільного секрету каналу передачі даних що буде захищений від втручання. Цей режим безпеки забезпечує захист від більшості атак через залучення в процес узгодження секрету людини. Тому виконання атаки прослуховування не матиме результатів тому що після узгодження секрету дані будуть зашифровані. Цей режим шифрування має авторизацію обох сторін з використанням спільного секрету який вводиться користувачем тому атака Man-in-the-Middle не матиме результату, так само як і з атакою двійника підключитися до фейкового або посереднього девайсу неможливо оскільки цей девайс не знає секрету необхідного для створення захищеного з’єднання. Атака викрадення ідентифікатора також не матиме сенсу при використанні цього режиму. Цей режим безпеки не може захистити девайси від атаки відмови в обслуговуванні як і інші режими захисту які існують в технологоії Bluetooth Low Energy.

Усі вищеперераховані режими мають свої недоліки. З них безпечним є тільки останній режим через те, що він унеможливлює створення каналу даних, за умови, якщо автентифікація з’єднання була провалена. Мінусом же режиму “Authenticated Pairing with Encryption” є необхідність залучення людини або іншого фізичного каналу передачі даних, що інколи є неприйнятним для продукту.

Використання додаткового фізичного каналу може бути неприйнятним для продукту з наступних причин.

1. Збільшення ціни. Додаткові канали передачі даних потребують додавання пасивних або активних елементів схеми, що буде впливати на кінцеву вартість девайсу.

2. Збільшення електроспоживання. Використання додаткових каналів передачі даних також потребує електроживлення. У випадку, коли додатковий канал повинен залишатися доступним, то весь час, щоб залишатися автентичним, додаткові фізичні канали будуть потребувати більшого використання електроживлення, ніж завдяки використанню технології Bluetooth Low Energy.

3. Нерентабельність. Додатковий канал передачі даних має завчасно бути безпечним і аутентифікованим, в такому разі використання інших бездротових технологій знизить до нуля усі плюси від використання Bluetooth Low Energy.

Залучення людини до процесу створення з’єднання може бути неприйнятним з наступних причин.

1. Потреба в додаванні інтерфейсів вводу/виводу. Щоб людина могла підтвердити автентичність з’єднання, вона повинна перевірити, що комунікація відбувається між потрібними девайсами. Оптимальним способом є введення або порівняння номеру на обох девайсах. Проте, зазначений спосіб потребує додавання інтерфейсів вводу/виводу(наприклад, дисплей та клавіатура), що збільшує ціну, підвищує складність реалізації та електроспоживання девайсів. Можливий також варіант із натисканням однієї кнопки на девайсі, але він не забезпечує необхідного рівня автентифікації. До того ж, така дія може бути виконана помилково.

2. Потреба в постійному втручанні в створення з’єднання на кожному девайсі. Схема з залученням людини може бути прийнятною для таких продуктів, як фітнес датчики або медичні сенсори, але подобна схема не зможе використовуватися в системах автоматизації. Наприклад, переміщення товарів на складі автономними дронами. Цей варіант є неприйнятним оскільки в ситуації

з використанням великої кількості девайсів, працеспроможність яких залежить від каналу передачі даних потреба в ручному завіренні зв’язку на кожному нівелює переваги від використання автоматичних автономних систем, а також викличе проблеми якщо таке ручне завірення буде потрібне в процесі перепідключення.

Згідно з наведеними причинами усі режими мають наступні недоліки.

1. Режим без захисту не є прийнятним для продуктів яким потрібна цілісність, конфіденційність та автентичність даних. Оскільки цей режим характеризується практично повною відсутністю безпеки.

2. Режим “Just works” може захистити канал передачі даних лише за умови якщо атакуючий не має можливості передавати (підроблювати) дані. При умові що атакуючий може підробити дані на початку процесу створення спільного секрету канал передачі даних більше не можна вважати захищеним.

3. Режим шифрування з автентифікацією може бути неприйнятним через необхідність залучення користувача або додатковий фізичний канал передачі даних в процес узгодження спільного ключа для каналу передачі даних

2.2 Висновок до другого розділу

Зважаючи на описані вище проблеми існуючих механізмів захисту з’єднання використання їх не є зручним в усіх випадках. Така ситуація існує тому, що за основу взята необхідність можливості з’єднання будь-якого девайсу Bluetooth Low Energy з будь-яким іншим девайсом цієї технології.

І все ж, коли використання додаткового фізичного каналу даних, такого як NFC (Near Field Communication), не є проблемою, або, коли користувач може власноруч перевірити автентичність з’єднання, режим шифрування з автентифікацією є оптимальним варіантом, за умови, якщо застосовувати тільки стандартні методи захисту. Так оптимальним варіантом для медичних сенсорів може бути використання NFC, оскільки використання даного фізичного інтерфейсу потребує надзвичайно близької (кілька сантиметрів) відстані між

девайсами, а для фітнес трекерів, які зазвичай мають екран, найбільш оптимальним, за умови використання лише стандартних методів захисту, буде використання схеми порівняння чисел на обох девайсах. Проте, фактор зручності користування девайсами може бути причиною небезпечного рішення використання режиму “Just Works”, оскільки він не вимагає будь яких дій від кінцевого споживача.

В інших типових випадках використання технології Bluetooth Low Energy подібні варіанти можуть бути неприйнятними. Наприклад, в системах з автоматизації з використанням автономних дронів залучення людини відмінить усі переваги автоматизації, а використання таких фізичних інтерфейсів як NFC потребує створення інфраструктури і впровадження цієї технологій у кожний автономний пристрій. Не визначеним, до того ж, залишається варіант з втратою підключення в процесі роботи, адже для перепідключення потрібно буде знову використати NFC, що потребує зближення з інфраструктурою автентифікації. Це може знизити ефективність роботи системи. У випадку, коли з’єднання має бути швидко встановлене з будь-яким з інших дронів для коригування маршрутів та попередження аварій, використання стандартного режиму шифрування з автентифікацією не може бути використано, оскільки процес аутентифікації не може бути проведений швидко.

РОЗДІЛ 3. РОЗРОБКА ВЛАСНОГО ЗАСОБУ ЗАХИСТУ ДЛЯ ТЕХНОЛОГІЇ BLUETOOTH LOW ENERGY

3.1 Теоретична частина реалізації захисту технології Bluetooth Low Energy

Вирішенням проблеми надійної автоматичної авторизації девайсів за технологією Bluetooth Low Energy без залучення людини може бути здійснено завдяки виконанню авторизації з використанням криптографічних протоколів на базі використання попередньо згенерованих сертифікатів, завірених єдиним центром сертифікації.

В такій схемі девайси, які в подальшому повинні утворювати захищене з'єднання і комунікувати між собою, програмуються з попередньо згенерованим сертифікатом та приватним ключем, завіреними центром сертифікації виробника або адміністратора девайсу. Під час створення захищеного з'єднання, девайси передають свої сертифікати один одному і перевіряють валідність сертифікату іншої сторони. Також, використання сертифікатів створює можливість адміністратору відкликати сертифікат конкретного девайсу, що зробить подальшу комунікацію з ним неможливою. Аналогічні міри можуть бути використані для попередження будь-яких атак на інфраструктуру, в якій використовується велика кількість девайсів, у разі викрадення або втрати одного з девайсів. Використання ж сертифікатів з обох сторін дозволяє обом сторонам впевнитися з ким утворюється з'єднання, що є набагато кращим варіантом, ніж використання сертифікату лише з однієї сторони, оскільки, навіть якщо центральний девайс буде захищений від підміни переферального девайсу, такий самий захист для периферійного девайсу може забезпечити впевненість в тому, що аніякі дані з периферійного девайсу не будуть передані неавторизованній стороні, що, в свою чергу, попередить витоки інформації.

Після підтвердження валідності сертифікатів повинна бути проведена фаза підтвердження володіння приватним ключем (proof of possession). Це пов'язано з тим, що сертифікати є публічними і перехоплення їх не буде становитиме

проблеми. Стадія підтвердження володіння забезпечує виконання фази автентифікації, підтвердження належності віддаленої сторони до інформації яка була нею передана.

Виконання валідації сертифікатів перед проведенням фази авторизації може зменшити можливість реалізації для “атаки у відмові в обслуговуванні” за умови, що атакуючий не володіє валідним ключем, оскільки це зменшить кількість криптографічних операції, що в свою чергу, зменшить навантаження на девайс.

Після проходження фаз авторизації і автентифікації, слід створити спільний секрет, який буде використано для шифрування каналу даних. Завдяки тому, що використання асиметричного ключа для потокового шифрування є неефективним, надалі слід створити спільний симетричний ключ. Такий підхід може бути реалізований шляхом залучення алгоритмів симетричного шифрування. Оскільки для створення ключа потрібно обмінюватися спільними параметрами які можуть бути перехоплені(що може призвести до відновлення атакуючою стороною ключа потокового шифрування), надалі, слід використовувати наявні асиметричні ключі задля узгодження захисту передачі параметрів для спільного секрету.

Оптимальним способом для узгодження захисту передачі параметрів для спільного секрету буде застосування протоколів узгодження спільного секрету на базі асиметричних ключів, реалізацією якого може бути “Elliptic-curve Diffie-Hellman”, протокол Діффі-Хеллмана на еліптичних кривих. Протокол Діффі-Хеллмана дозволяє безпечно узгодити спільний секрет через незахищене з’єднання за допомогою математичних операцій. Однак класичний варіант цього протоколу вразливий до атак, в яких атакуючий може виступати посередником, через те, що в ньому не передбачена авторизація даних. Це призводить до того, що атакуючий може змусити обидва девайси створити спільний секрет з посереднім девайсом і, в подальшому, останній матиме можливість контролю сесії, розшифровуючи дані від однієї сторони, маніпулюючи ними і відправляючи іншій стороні з використанням ключа іншої сторони.

Водночас, використання вдосконаленого алгоритму який використовує приватні ключі як частину алгоритму унеможливить атаку посередника, оскільки ключі вже було автентифіковано, а також зменшить кількість ітерацій між девайсами тому, що додаткові дані передавати буде не потрібно. Дана фаза може бути до того ж використана задля підтвердження володіння, оскільки використати спільний секрет буде можливо тільки за умови, що приватні ключі на обох девайсах відповідають публічним ключем, які заздалегідь були завірені центром сертифікації.

З використанням спільного секрету в потоковому шифрування канал даних між девайсами можна вважати захищеним. Атакуючий наразі не може дізнатися, що за дані передаються цим захищеним каналом. Втім, використання згенерованого секрету не є стовідсотково безпечним методом. Небезпека пов’язана з тим, що секрет буде однаковим для кожної ітерації створення безпечного з’єднання це створює ризики майбутнього дешифрування каналу даних для отримання інформації, що була ним передана за умови, що атакуючий зберігає усі передані зашифровані дані. Для проведення такої атаки потрібно знати обидва приватні та публічні ключі, використанні в схемі ECDH (elliptic curve Diffie-Hellman), що якщо і не реалістично в моменті, однак, зможе бути реалістичним в майбутньому.

Для вдосконалення схеми оптимальним буде використання створеного симетричного ключа для захищеного виконання класичного алгоритму Діффі-Хеллмана. Оскільки канал вже вважається захищеним від маніпуляції, вистачить використання класичної схеми, під час якої для генерації секрету використовуються згенеровані випадковим чином значення, які спиратимуться з пам’яті заледве спільний секрет буде узгоджено. Подібні дії дозволяють створити матеріал для спільного ключа, який неможливо буде відновити з легкістю навіть за умови компрометації шифрування попереднього рівня.

Для збільшення ентропії спільного ключа і як результат ускладнення його відновлення можливим буде використання функції введення ключа на кшталт типу “HKDF”, який є варіантом “KDF” (key derivation function) функції введення

ключа на основі коду автентифікації повідомлення HMAC. Отриманий в результаті виконання даної функції спільний секрет є стійким і може використовуватися для подальшого потокового шифрування. Поза тим, використання алгоритму HMAC як частини алгоритму потокового шифрування дозволить впевнитися що зашифровані дані не мають у собі примусово доданих помилок атакуючою стороною.

Ще одним механізмом додаткового захисту каналу комунікації даних від дешифрування атакуючою стороною в майбутньому є періодична зміна спільного секрету. В такому випадку якщо припустити що атакуючий має можливість підібрати спільний секрет, відновити буде можливо лише дані передані під час цього періоду. Водночас, в подальшому використанні все того ж алгоритму Діффі-Хеллмана для оновлення спільного секрету отримати достатньо інформації для відновлення наступного ключа буде неможливо. За для того щоб розшифрувати наступні дані потрібно буде знову з нуля підібрати спільний секрет.

Описаний алгоритм створення захищеного з’єднання задовольняє вимоги в автентифікації за шифрування з’єднання. Крім того, він робить неможливим відновлення всієї інформації в майбутньому, навіть враховуючи, що обидва девайси були захоплені атакуючою стороною. Проте, компрометація приватних ключів обох сторін створює загрозу для майбутніх спроб створення захищеного з’єднання якщо атакуючий може провести атаку посередника і підмінити параметри, що використовуються в алгоритмі Діффі-Хеллмана під час створення ключа сесії, змусивши обидва девайси утворити спільний секрет з девайсом посередником.

Отриманий спільний секрет сесії можна використовувати як секрет шифрування в потоковому шифрі, реалізованому на стороні чіпу блютуз, визначивши його як секрет для режиму “just works”. У зв’язку з тим, що авторизація каналу даних і його захист було завершено, це нівелює проблеми режиму “just works”. Іншим варіантом є використання програмного потокового шифрування як було описано вище.

3.2 Практична частина реалізації захисту Bluetooth Low Energy

Однією з сфер використання технології Bluetooth Low Energy є Human Input Devices. Такими девайсами є клавіатури і комп'ютерні миші. Бездротові версії вищезазначених девайсів створюють ризики для безпеки системи з якою вони застосовуються за умови використання слабкого захисту каналу передачі даних. Так, наприклад, використання режиму "Just Works" в таких дивайсах створює можливості для атакуючого, що може виконати активну атаку посередника отримавши контроль над каналом передачі даних, отримувати інформацію про натиснуті клавіші або навіть посилати на приймаючий девайс власні послідовності натискання клавіш, що дозволяє атакуючому виконати будь-які дії або код на віддаленій машині. Погіршує проблему те, що подібні дії не можуть бути відслідковані за допомогою програмних комплексів, встановлених на віддаленій машині як протиправні, оскільки віддалена машина повинна довіряти даним які отримані від подібного класу девайсів.

Для практичної реалізації описаного захисту зроблена фізична версія Human Input Device з бездротовим з'єднанням, а саме, допоміжна бездротова клавіатура, натискання на клавіші якої виконує завчасно запрограмовані дії на віддаленому девайсі. Фізична сторона девайсу складається з:

1. NRF52 Dev Board (рисунок 3.1) – плата для розробки девайсів на базі технології Bluetooth Low Energy, плата містить на собі чіп nrf52832 який виконує бездротову комунікацію та може бути запрограмований на виконання будь-яких дій з використанням пінів вводу-виводу які присутні на платі в достатній кількості. Буде використана на стороні девайсу що віддає команди [12].

2. Надрукований на 3д принтері корпус девайсів.

3. Матрична клавіатура розмірами 4 на 4 клавіші (рисунок 3.2) - використовуються для отримання натиску клавіш від користувача.

Рисунок 3.1 – Фото NRF52 Dev Board

На рис. 3.2 подано вигляд матричної клавіатури розмірами 4 на 4 клавіші.

Рисунок 3.2 – Фото матричної клавіатури розмірами 4 на 4 клавіші

4. nRF52840 Dongle (рисунок 3.3) – невелика плата розробки з наявним чипом, що програмується Bluetooth Low Energy а також форм-фактором невеликого USB девайсу. Буде використано на стороні приймаючого девайсу [14].

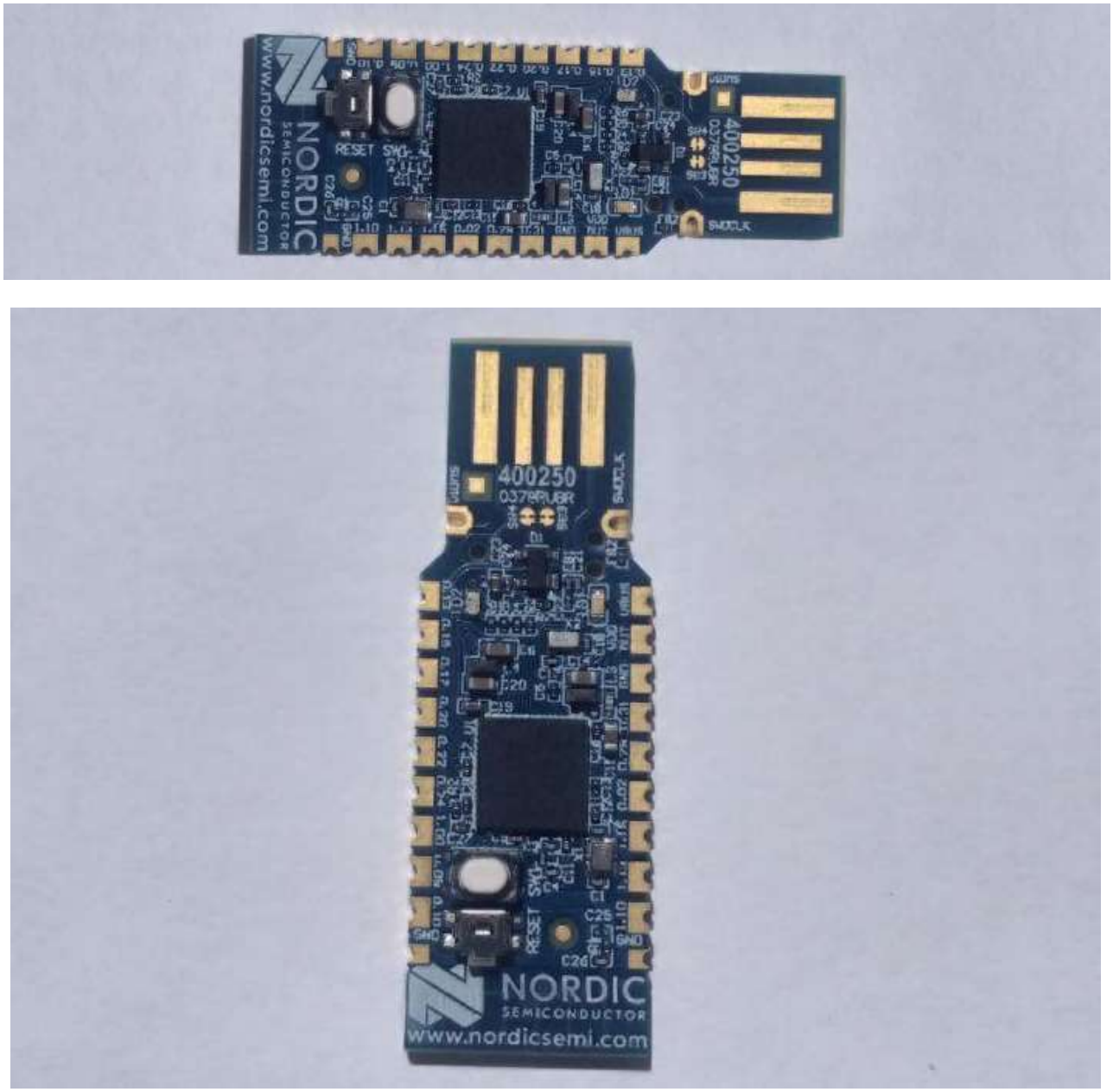

Рисунок 3.3 – Фото nRF52840 Dongle

Вибір даних девайсів обумовлено легкість створення на їх базі прототипів нових девайсів. Оскільки друкована плата NRF52 Dev Board містить всі необхідні для роботи компоненти залишається з’єднати матричну клавіатуру з самою платою. Схема з’єднання показана на рисунку 3.4.

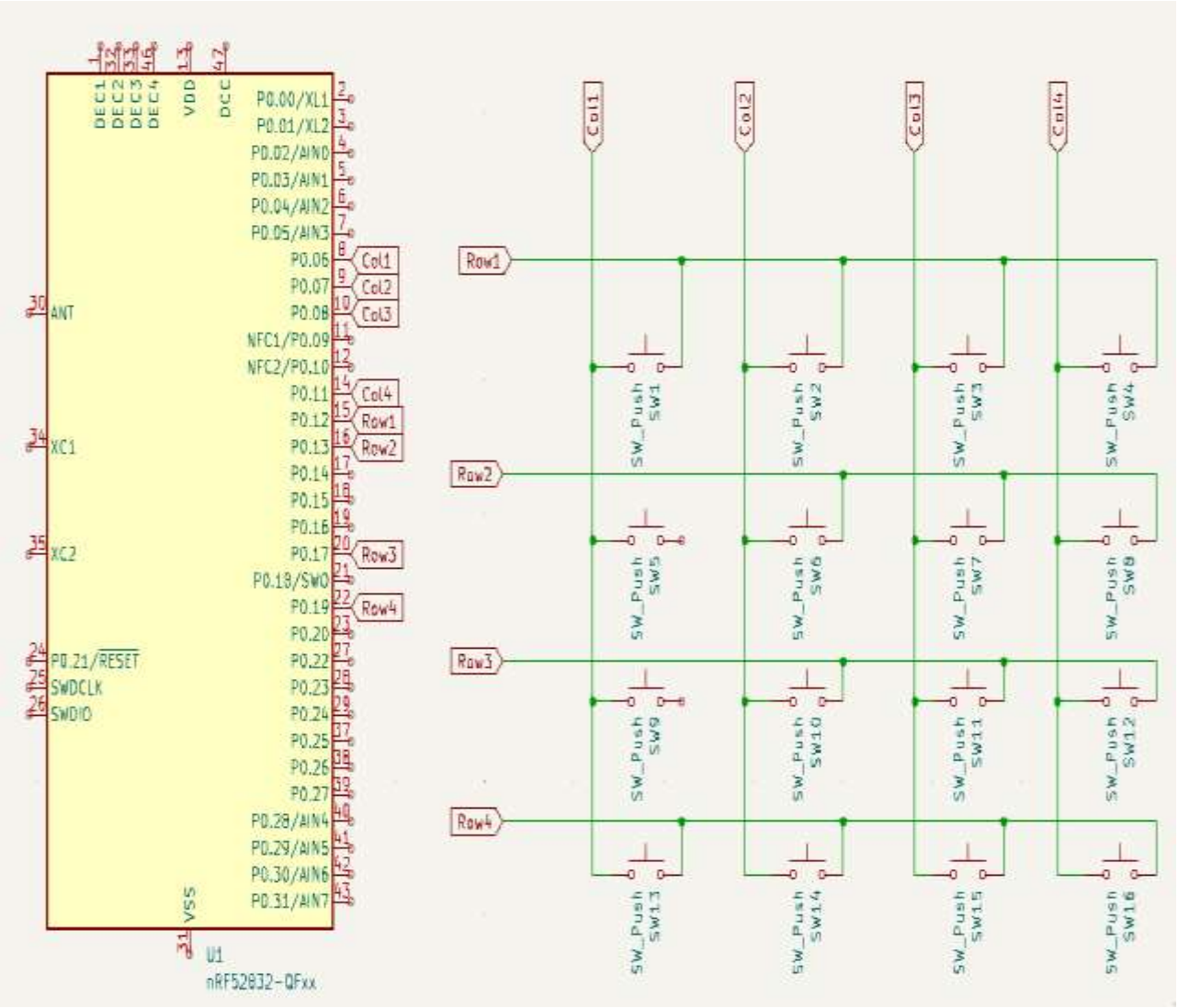

Рисунок 3.4 – Принципова схема девайсу

Оскільки nrf52840 Dongle є готовим до роботи девайсом, більше дій зі сторони заліза не потребується.

Програмна сторона девайсів містить наступні компоненти:

1. nRF5 SDK (Software Development Kit) - набір програмних компонентів а також заголовкових файлів від постачальника, для розробки програмного забезпечення на базі вибраних девайсів. Заголовочні файли в цьому SDK містять інформацію про функції “Softdevice” які можна викликати для виконання дій з різними інтерфейсами чіпі в[11].

2. S132 Softdevice – пропрієтарна прошивка від постачальника, яка реалізує низькорівневі функції девайсу. Підходить для використання з чіпом nRF52832, буде використана на стороні передаючого девайсу [13].

3. S140 Softdevice – пропрієтарна прошивка від постачальника, яка реалізує низькорівневі функції девайсу. Підходить для використання з чіпом nRF52840, буде використано на стороні приймаючого девайсу[15].

4. MbedTLS v3.6.0 LTS – відкрита бібліотека, що реалізує необхідні криптографічні функції. Вибір цієї бібліотеки обумовлений позиціюванням її як легкої в сенсі використання апаратної пам’яті бібліотеки для легкого інтегрування в різні девайси [16-17].

Описаних програмних компонентів достатньо для реалізації описаного в попередньому розділі алгоритму узгодження безпечного каналу передачі даних.

На рисунку 3.5 позначено криптографічні алгоритми, які використовуються в реалізації програмного забезпечення.

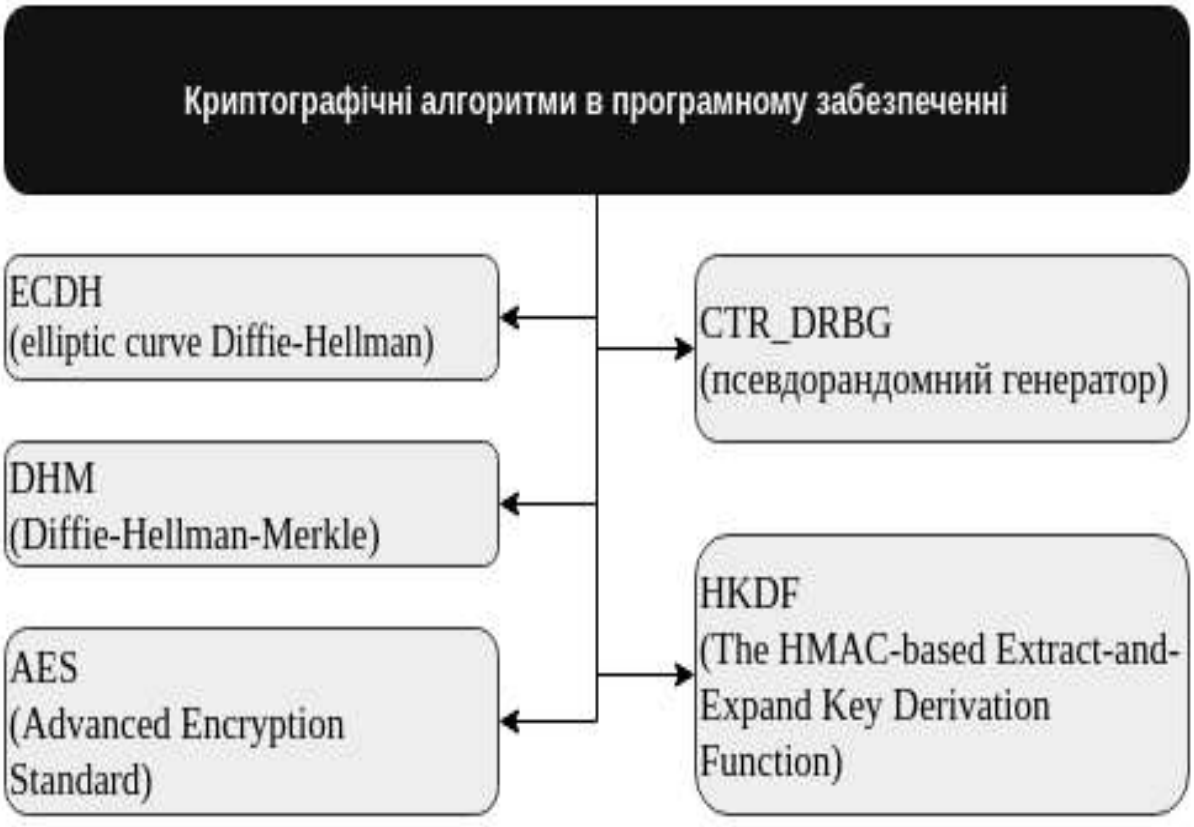

Рисунок 3.5 – Криптографічні алгоритми, що використовуються в програмному забезпеченні

ECDH (elliptic curve Diffie-Hellman) – криптографічний алгоритм узгодження спільного секрету через небезпечний канал передачі даних. Алгоритм роботи ECDH наведений на рисунку 3.6. Використовує криптографію на еліптичних кривих. Опис функції доступний в заголовочному файлі “mbedtls/ecdh.h”.

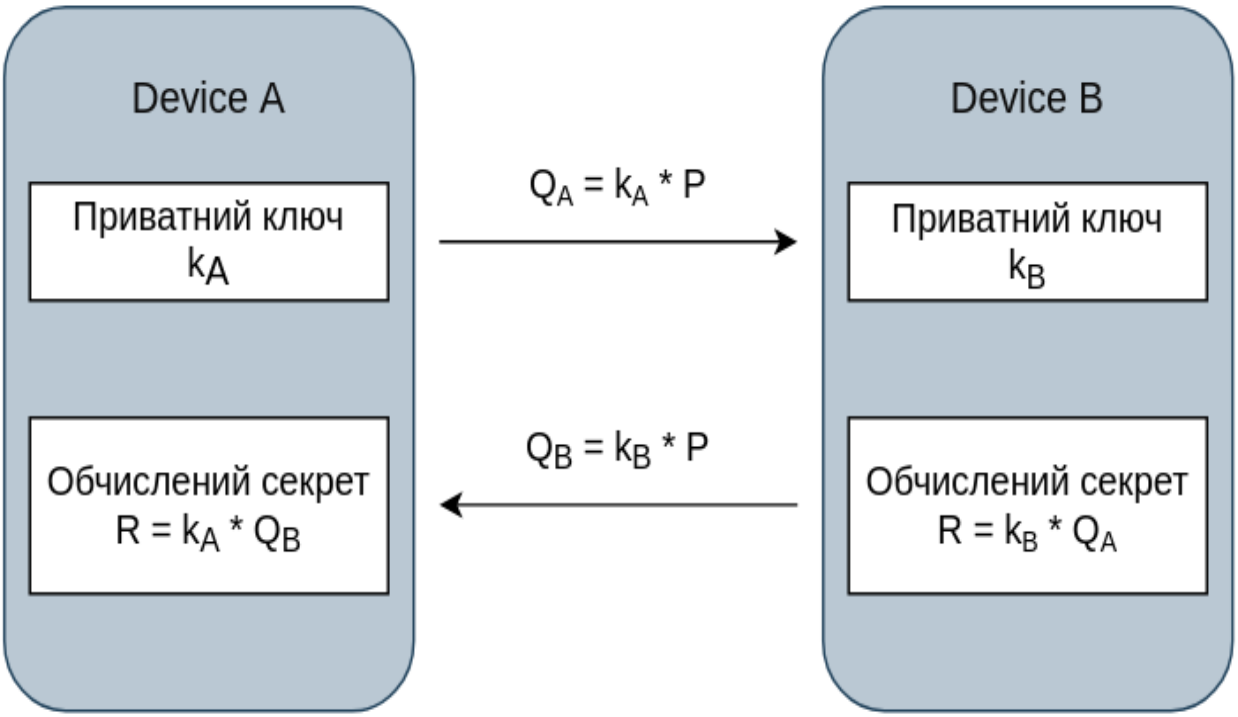

Рисунок 3.6 – Алгоритм роботи ECDH

DHM (Diffie-Hellman-Merkle) – криптографічний протокол узгодження спільного секрету через небезпечний канал передачі даних (рисунок 3.7)[18]. На відміну від ECDH він не може бути використаним через незахищене від підробки з’єднання оскільки атакуючий може підмінити параметри надіслані небезпечним каналом передачі даних і таким чином змусити створити пару з собою а не з реальною іншою стороною. Опис функції доступний в заголовочному файлі “mbedtls/dhm.h”.

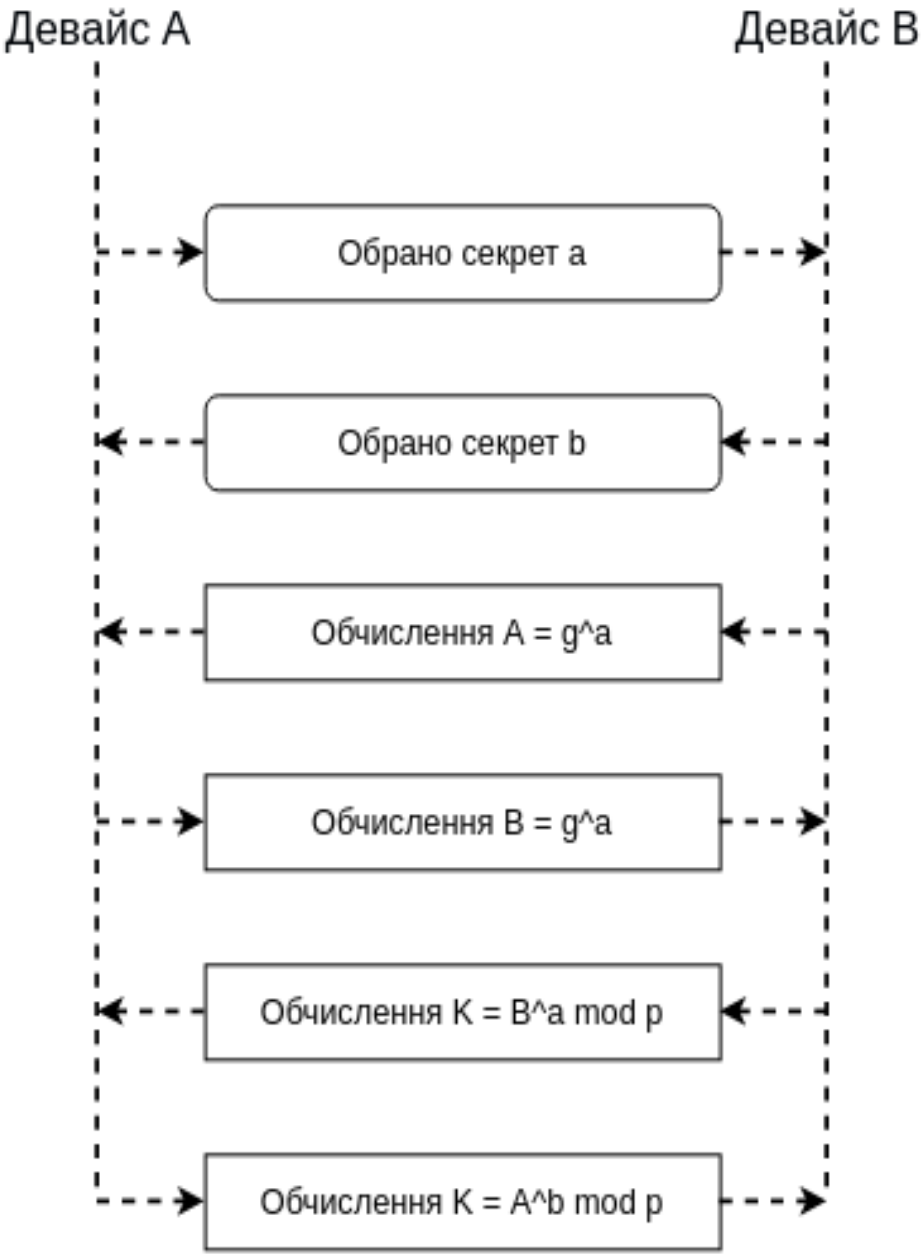

Рисунок 3.7 – Алгоритм роботи DHM

HKDF (The HMAC-based Extract-and-Expand Key Derivation Function) – криптографічний алгоритм збільшення ентропії ключового матеріалу на основі алгоритму HMAC(hash-based message authentication code). Він використовується для посилення спільних секретів. Опис функції доступний в заголовочному файлі “mbedtls/hkdf.h”. На рисунку 3.8 наведено алгоритм роботи HMAC [19].

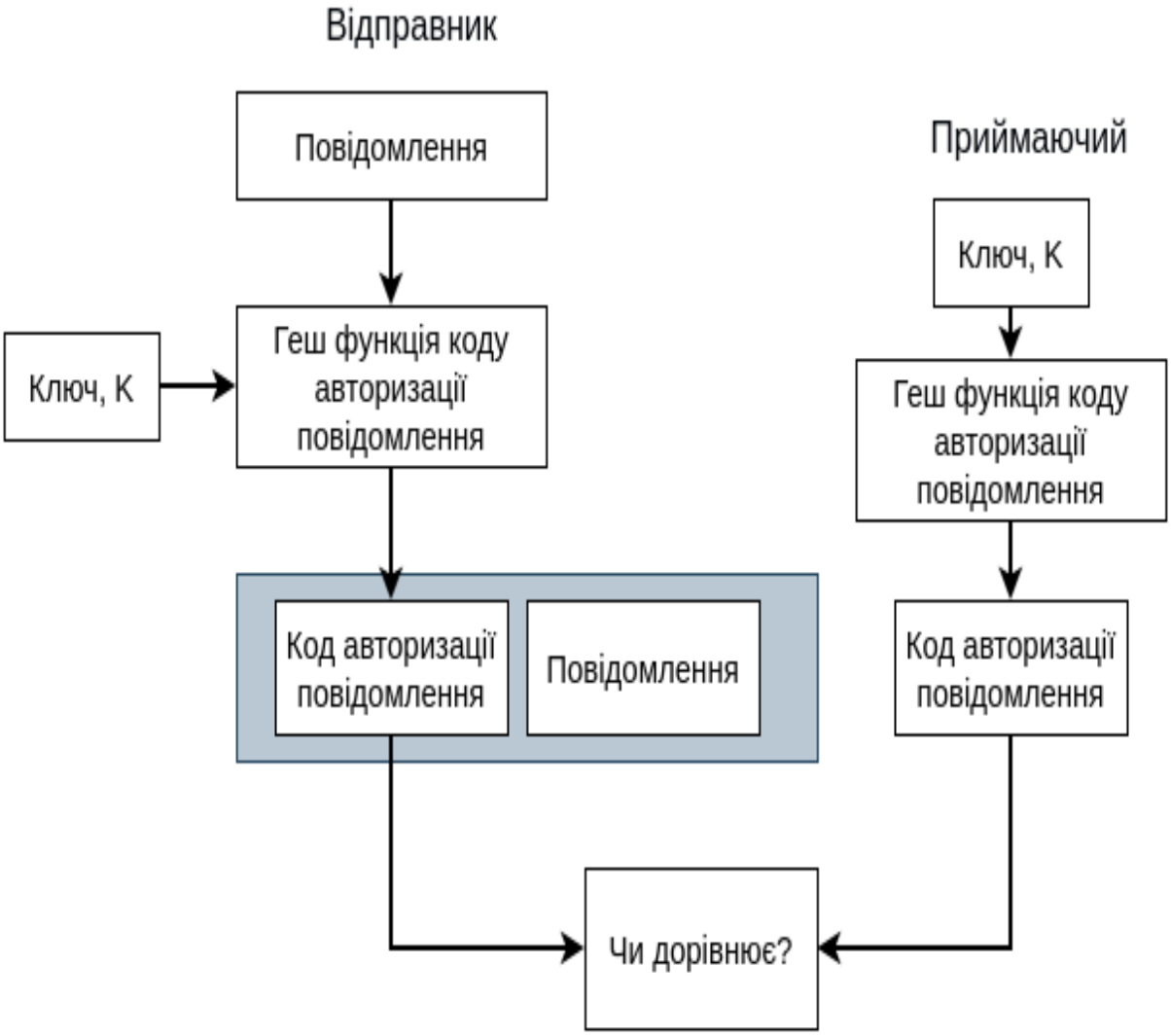

Рисунок 3.8 – Алгоритм роботи HMAC

AES (Advanced Encryption Standard) в режимі GCM(Galois/Counter Mode) - криптографічний алгоритм шифрування в режимі лічильника з автентифікацією (рисунок 3.9) [20]. Використовується для потокового шифрування та дешифрування даних, а також верифікацію їх цілісності. Опис функції доступний в заголовочному файлі “mbedtls/cipher.h”.

CTR_DRBG псевдорандомний генератор – алгоритм генератору псевдо рандомних чисел з використанням AES-256 для збільшення ентропії. Використовується для генерації псевдо рандомних чисел, які будуть використані з алгоритмом DHM. Опис функцій доступний в заголовочному файлі “mbedtls/ctr_drbg.h”.

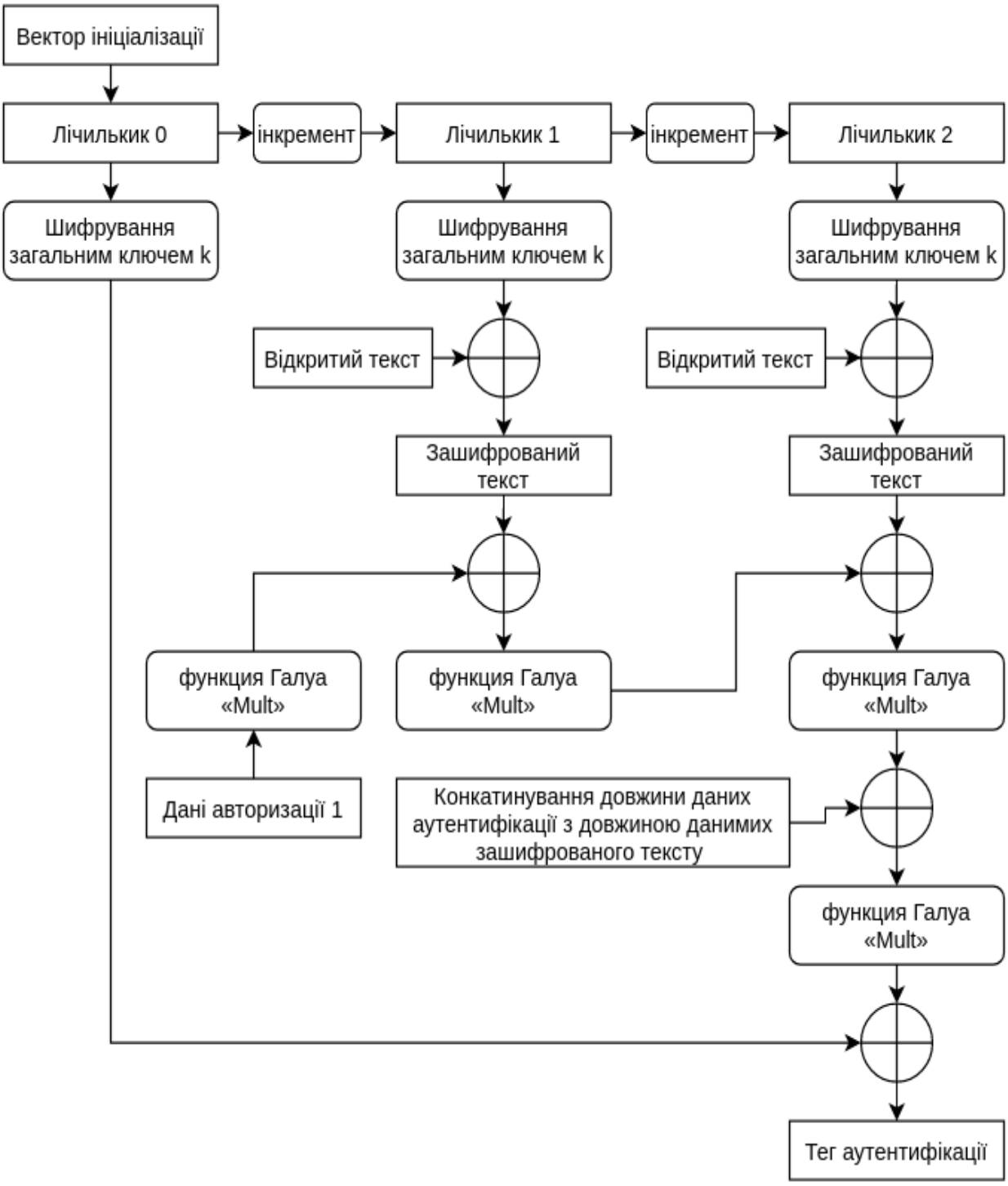

Рисунок 3.9 – Алгоритм роботи AES в режимі GCM

Окрім описаних алгоритмів також використовуються деякі допоміжні механізми для встановлення захищеного з’єднання:

1. Робота з сертифікатами. Функції криптографічної бібліотеки дозволяють проводити обробку сертифікатів в форматі x509 перевіряти їх

відповідність та верифікувати підпис спільного центру сертифікації. Опис функцій доступний в заголовочному файлі “mbedtls/x509_crt.h”.

2. Робота з еліптичними кривими. Вибір використання еліптичних кривих було зроблено оскільки вони потребують менше ресурсів системи ніж ключи системи rsa та є основою реалізації алгоритму ECDH, функції криптографічної бібліотеки дозволяють імпортувати та виконувати математичні дії з цією криптографічною системою. Опис функцій доступний в заголовочному файлі “mbedtls/ecp.h”.

3. Генератор ентропії. Важливою частиною нового протоколу є використання алгоритму узгодження ключа DHM. В цьому алгоритмі використовуються рандомні числа для збільшення складності спільного секрету. Щоб отримати найбільш рандомні числа найкращим варіантом є використання шумів які може зчитати мікроконтролер під час своєї роботи. Такі шуми не можна вгадати, а, отже, вони гарно підходять для генерації рандомних чисел. Ще одним можливим варіантом отримання ентропії є додавання частини інформації отриманої під час роботи з зовнішніми інтерфейсами або обробки деякої інформації, хоча такі способи отримання ентропії є менш надійними, все ж, вони гарно підходять для подібних цілей. Криптографічна бібліотека має реалізацію функції, які допомагають отримувати ентропію для генерації рандомних чисел. Опис функцій доступний в заголовочному файлі “mbedtls/entropy.h”.

4. Робота з даними в форматі DER (Distinguished Encoding Rules). DER - це формат для збереження та передачі сертифікатів, в ньому використовується кодування даних в ASN.1(Abstract Syntax Notation One). Дане кодування представляє структури даних у криптографії. Вибір використання стандарту DER був зроблений, оскільки він зберігає інформацію в вигляді байтів, що примусово впорядковані для забезпечення однозначності. Використання ASN.1 в цьому форматі робить анотацію заголовків розмірів та тегів рівними декільком байт, зазвичай одному, що дозволяє сильно зменшити розміри файлу, що є важливим при використанні з мікроконтролерами, які мають обмежений розмір пам’яті. Проте, на відміну від деяких інших структур, зберігання є складним для

розуміння людиною. Криптографічна бібліотека має функції для роботи з даними в форматі DER та з структурами ASN.1. Опис функцій доступний в заголовочному файлі “mbedtls/asn1.h” та “mbedtls/x509_ctr.h”.

5. Криптографічна геш-функція SHA (Secure Hash Algorithm) з використанням довжини в 256 біт. SHA є одним із стандартів криптографічних геш-функції що використовується для генерації хеш-значення фіксованої довжини. В розробленому протоколі використовується довжина гешу в 256 біт, що є дуже надійним. Геш використовується в генераторі рандомних чисел, системі HMAC, перевірці валідності сертифікатів та інше. Криптографічна бібліотека реалізує функції для генерації геш значень. Опис функцій доступний в заголовочному файлі “mbedtls/sha256.h” а також частина функції та інтерфейсів описана в файлі “mbedtls/md.h”.

Описаних криптографічних алгоритмів та допоміжних функції достатньо для реалізації описаного в попередньому розділі алгоритму узгодження спільного секрету та виконання потокового шифрування та дешифрування.

Алгоритм роботи програмного забезпечення є наступним:

1. Девайс передавач надсилає рекламні пакети для того, щоб бути знайденим девайсом приймачем
2. Девайс приймач отримує рекламний пакет і ініціалізую процесс узгодження спільного секрету. Відправляє передавачу свій сертифікат і очікує на сертифікат передавача.
3. Передавач отримує сертифікат приймача і перевіряє що він дійсно завірений їх спільним центром сертифікації, перевіривши передавач надсилає свій сертифікат у разі якщо отриманий сертифікат є валідним або сигнал відмови якщо він таким не був і повертається на початок алгоритму починаючи розсилати рекламні пакети. Передавач також генерує спільний секрет з використанням свого приватного ключу і отриманого публічного ключа за допомогою алгоритму ECDH. Спільний ключ проходить через HKDF для посилення його складності.
4. Приймач отримує сертифікат передавача і перевіряє що він справді

підписан спільним центром сертифікації. У разі якщо отриманий сертифікат є валідним приймач генерує спільний секрет з використанням свого приватного ключу і отриманого публічного ключа за допомогою алгоритму ECDH. Спільний ключ проходить через HKDF для посилення його складності. У разі якщо верифікація передавача була провалена, приймач передає сигнал помилки і повертається на початок алгоритму. Приймач шифрує за допомогою спільного секрету запит на створення секрету сесії та дані для створення секрету і надсилає його передавачу. При шифруванні використовується алгоритм AES в режимі CGM. Дані вираховуються для спільного секрету сесії генеруються з використанням псевдо рандомного генератору чисел.

5. Передавач отримує запит на створення ключа сесії і дані для створення цього секрету по зашифрованому каналу, розшифровує параметри та генерує секрет сесії, шифрує свої параметри для генерації спільного секрету і відправляє їх приймачу. Під час дешифрування використовується той же алгоритм що і підчас шифрування, а спільний секрет сесії вираховується за допомогою алгоритму DHM, а дані для вирахування спільного секрету сесії зі сторони передавача генеруються за допомогою генератора псевдо рандомных чисел. Спільний секрет сесії проходить через алгоритм HKDF для посилення його складності.

6. Приймач отримує параметри для вираховування спільного секрету сесії по зашифрованому каналу, дешифрує та вираховує спільний секрет сесії за допомогою алгоритму DHM. Спільний секрет сесії проходить через алгоритм HKDF для посилення його складності. Приймач надсилає сигнал переходу на використання секрету сесії для шифрування і віднині використовує його для шифрування майбутніх даних.

7. Передавач отримує сигнал переходу на використання секрету сесії і надалі використовує його для шифрування.

8. Передавач очікує на натискання клавіш на клавіатурній матриці, як тільки було отримано натискання інформація про це шифрується за допомогою ключа сесій і відправляється приймачу.

9. Приймач отримує команду про натискання клавіші дешифрує її і відправляє на користувацький девайс (комп’ютер) для виконання.

10. Кожні 10 секунд приймач відправляє команду перевірки з’єднання на яку повинен відповісти приймач. Відсутність відповіді, або інший статус у відповіді означає що канал передачі даних було скинуто і потрібно провести процедуру заново.

11. Кожні 10 хвилин приймачем відправляється сигнал генерації нового секрету сесії, після чого приймач і передавач обмінюються інформацією для генерації нового секрету сесій, генеруються новий секрет і переходять на його подальше використання для шифрування та дешифрування усіх відправлених даних.

12. При отриманні сигналу відключення канал передачі даних вважається скинутим і для встановлення з’єднання потребується провести процедуру погодження ключів з самого початку.

Спрощено новий алгоритм можна поділити на декілька стадій.

Стадія 1: Ідентифікація (рисунок 3.10).

Клавіатура надсилає рекламні пакети для позначення свого існування, а приймач очікує отримати рекламні пакети перед тим як ініціювати з’єднання. При цьому, якщо клавіатура не отримує з’єднання у відповідь до розісланих рекламних пакетів, починається етап сну, з якого вона буде пробуджена після деякого інтервалу щоб повторити розсилання рекламних пакетів. Це зроблено для економії електроживлення якщо приймач не було підключено.

В свою чергу, приймач не переходить в сон оскільки очікується, що підключений до більш міського електроживлення. Приймач постійно сканує ефір в пошуку клавіатури.

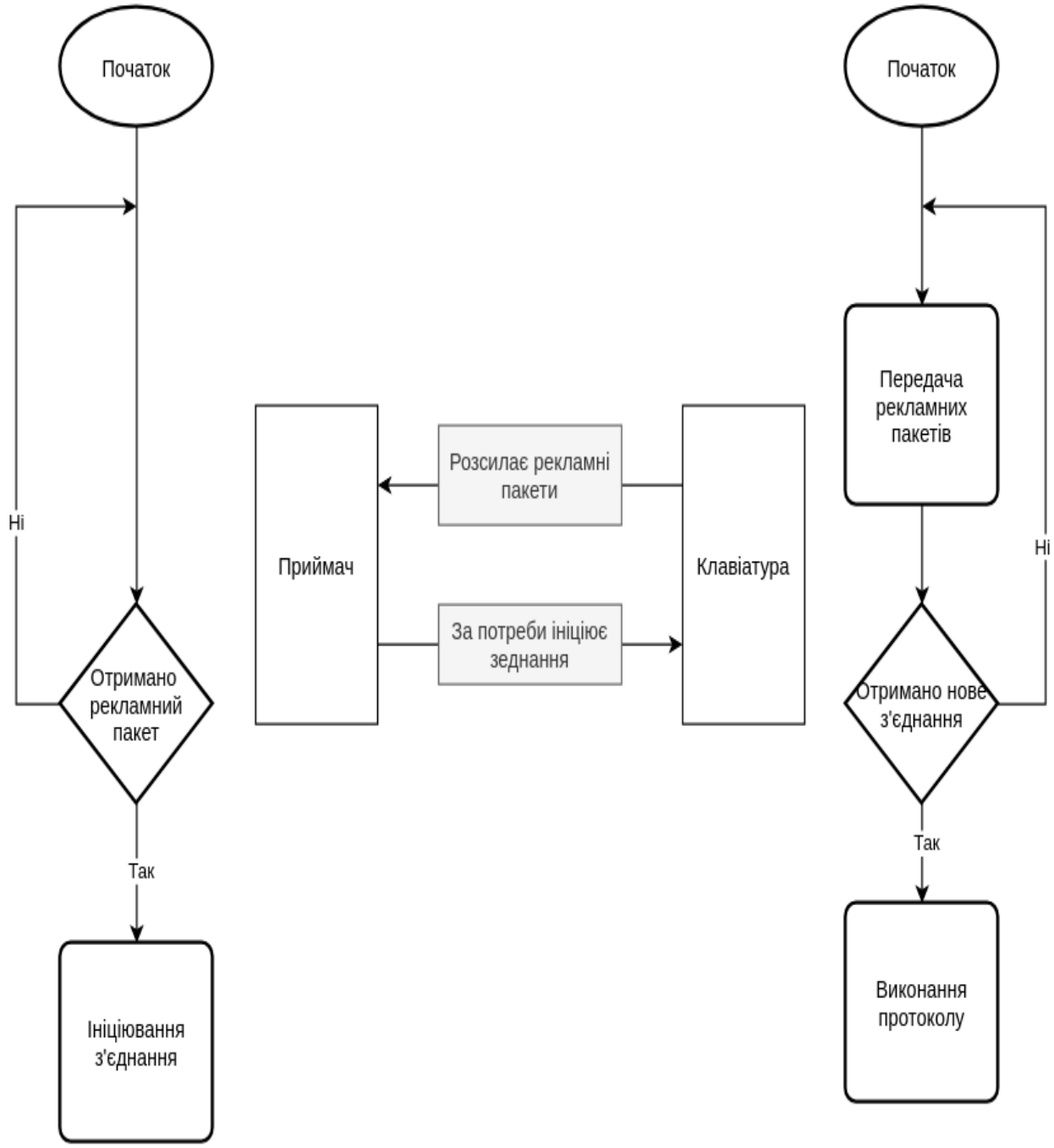

Рисунок 3.10 – Принцип роботи стадії 1

Стадія 2: Автентифікація (рисунок 3.11).

Дана стадія починається разом з ініціалізацією з’єднання. Спочатку приймач надсилає клавіатурі свій сертифікат і очікує відповіді. Клавіатура перевіряє що сертифікат правильний і якщо ні відправляє пакет з позначкою помилка, а якщо все гаразд відправляє свій сертифікат до приймача і генерує секрет на основі ключів з використанням ECDH.

Приймач перевіряє що сертифікат клавіатури правильний і якщо ні, то відправляє пакет з позначкою помилка, а в разі правильності - генерує секрет на основі ключів з використанням ECDH.

Отримання пакету з позначкою “помилка” скидає прогрес до стадії 1.

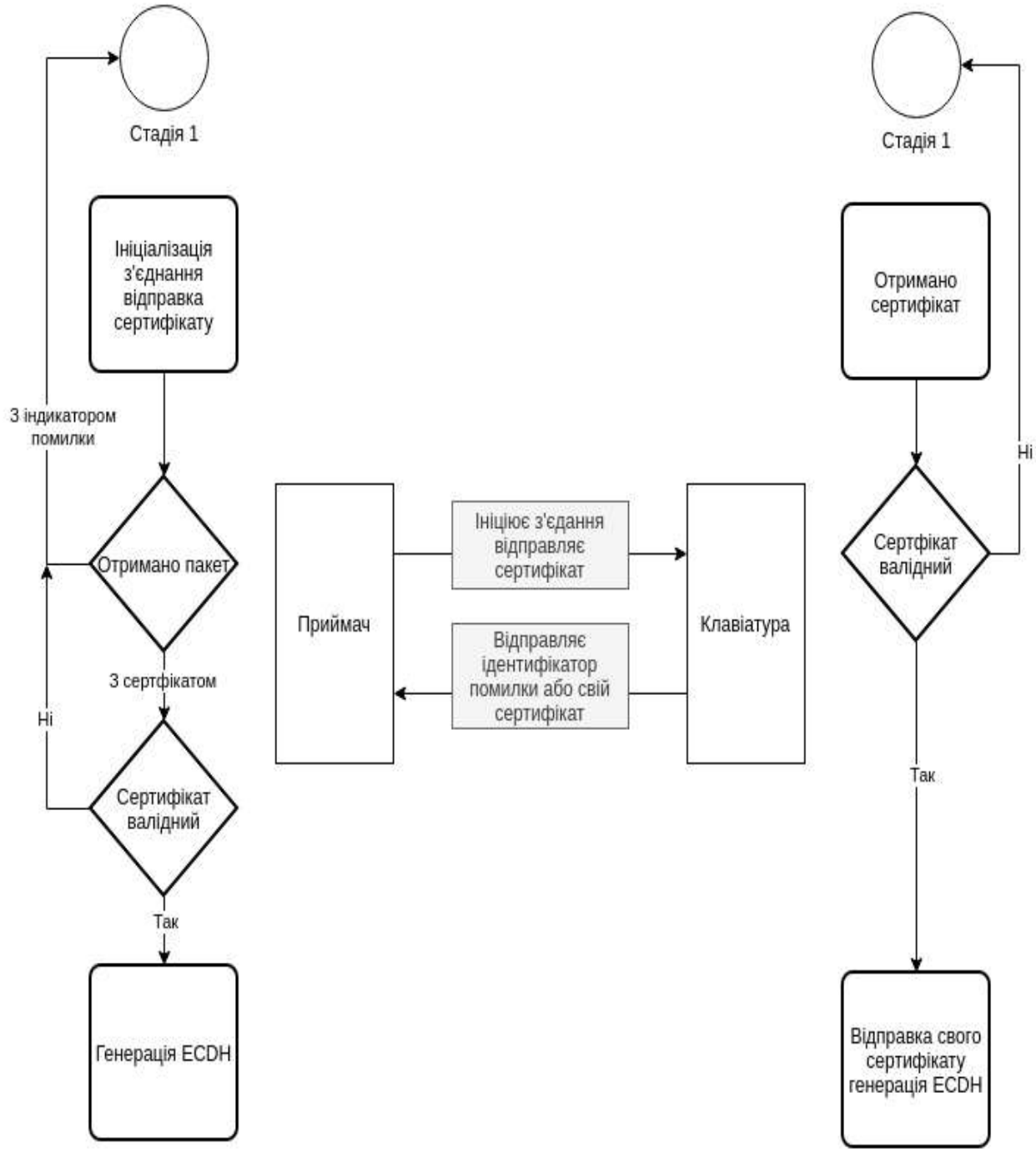

Рисунок 3.11 – Принцип роботи стадії 2

Стадія 3: Генерація секрету сесії(рисунок 3.12).

Після генерації статичного секрету ECDH обидва девайси його використовують для шифрування каналу передачі даних. Тепер необхідно згенерувати секрет сесії для шифрування в майбутньому вже саме даних про натискання клавіш.

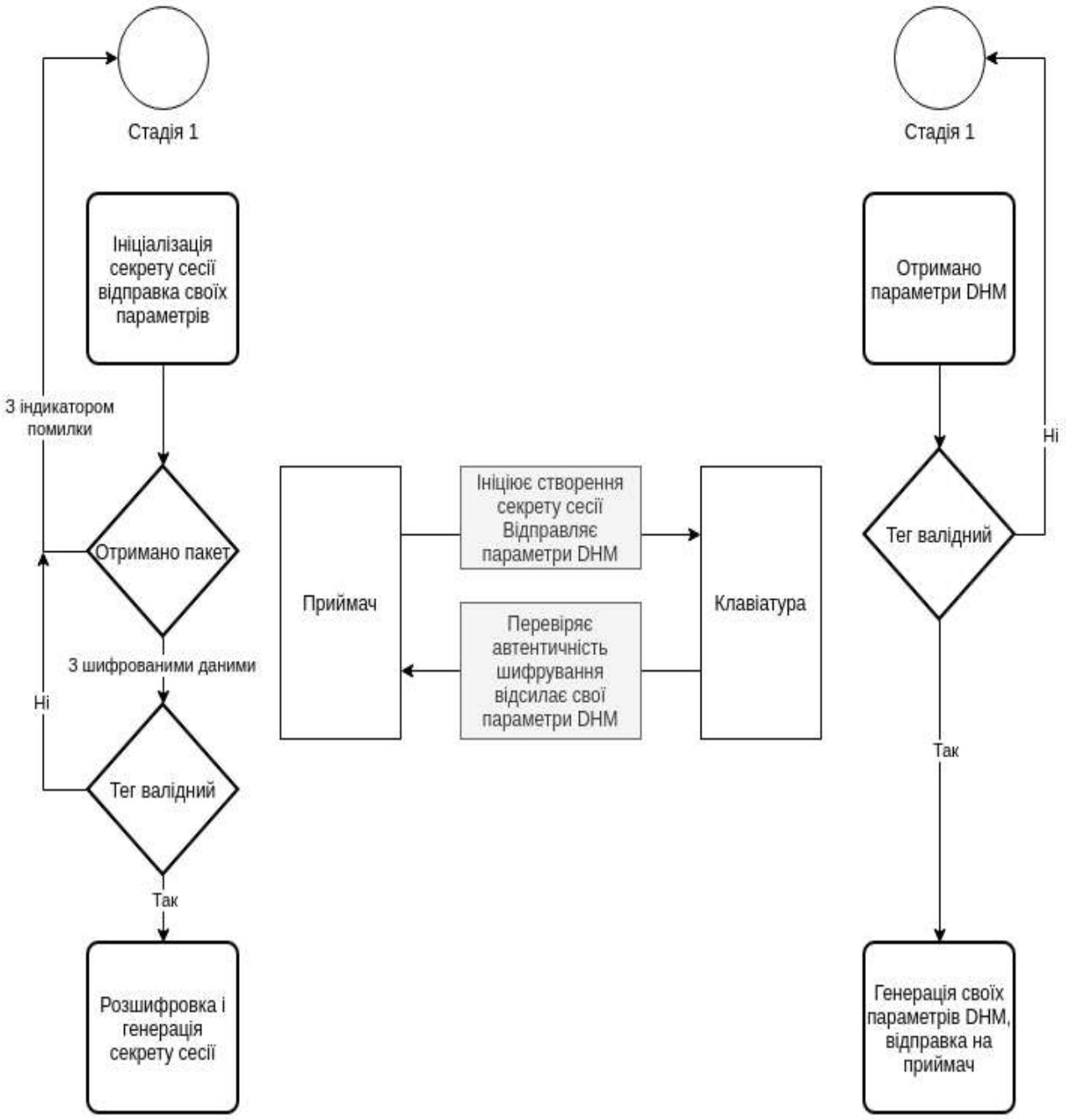

Рисунок 3.12 – Принцип роботи стадії 3

Для цього приймач відправляє пакет з маркером необхідності створення ключа сесії, а також, параметрами ECDH, дані в цьому випадку підписано через використання AES CGM. Клавіатура приймає дані і перевіряє тег автентифікації AES CGM в процесі розшифрування. Якщо він неправильний - з'єднання вважається поламаним і процес скидається до стадії 1. Якщо тег правильний – створюються власні параметри DHM, шифруються та відправляються на приймач, а також генерується спільний секрет з використанням параметрів приймача.

Приймач отримує пакет з зашифрованими даними передавача і перевіряє тег автентифікації. Якщо він неправильний з'єднання вважається поламаним і прогрес відкидається до стадії 1. Якщо тег правильний – розшифровані параметри використовуються для генерації секрету сесії.

Стадія 4: Передача даних (рисунок 3.13).

Тепер передавач відправляє дані про натиснуті клавіші використовуючи ключ сесії. Приймач отримує шифровані пакети і перевіряє тег автентифікації. Якщо дані було змінено, то тег буде не валідним. В такому разі приймач відправляє пакет з ідентифікатором помилки, велика кількість таких пакетів призведе до скидання прогресу на стадію 1.

Періодично відправляється пакет keep-alive для підтримання з'єднання, його відсутність або відсутність відповіді протягом 1 хвилини поверне прогрес на стадію 1.

Якщо передавач або приймач бажає завершити комунікацію, можлива відправка пакету з ідентифікатором end, такий пакет також є зашифрованими і перед закінченням з'єднання перевіряється тег. Подібний валідний пакет призведе до скидання прогресу на стадію 1.

Приведений алгоритм є реалізацією описаного в попередньому пункті захисту бездротового каналу передачі даних, а виконаний девайс є прикладом практичного використання описаного вище способу захисту технології Bluetooth Low Energy.

Код реалізованого протоколу наведений в додатку А та додатку Б даної роботи.

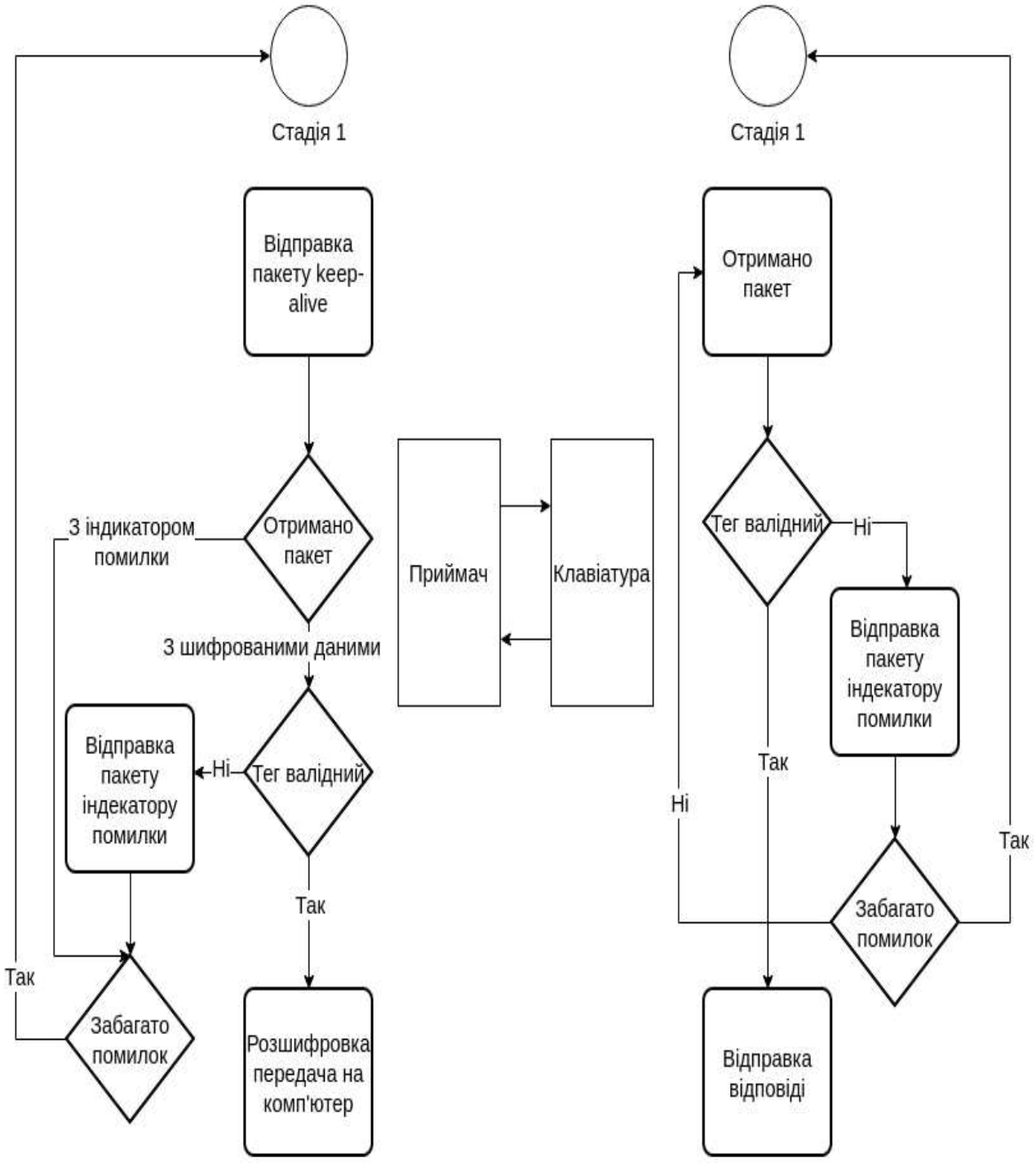

Рисунок 3.13 – Принцип роботи стадії 4

При передачі даних між девайсами окрім самого корисного навантаження, яким може бути як інформація для вираховування спільного секрету, так і інформація про натиснуті клавіші, також передається службова інформація. Ця

службова інформація передається в кожному пакеті даних в заголовку перед самими даними. Структура цього заголовку наведена в таблиці 3.1.

Таблиця 3.1

Структура заголовку пакету даних службової інформації

Октет (байт)		0								1								2								3							
	Біт	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	0	Status																Session secret generation number															
4	32	Message iteration number																Payload size															

1 Status (2 байти) – бітове поле з інформацією про статус з’єднання, відображає статус шифрування, зашифроване повідомлення чи ні, необхідність відправки відповіді на команду перевірки з’єднання, необхідність генерацій нового ключа сесії та статус дисконету.

2 Session secret generation number (2 байти) – номер генерації секрету сесії для конкретного повідомлення, збільшується кожного разу колу новий ключ було згенеровано, використовується для визначення який секрет сесії потрібно використати для дешифрування повідомлення.

3 Message iteration number (2 байти) – номер повідомлення, збільшується кожного разу коли нове повідомлення було відправлене, при отриманні кожне повідомлення з номером меншим за вже опрацьоване буде проігнороване.

4 Payload size (2 байти) – розмір зашифрованих даних.

Заголовок повідомлення має статичну структуру і тому його розмір завжди буде однаковим. При цьому, щоб запобігти підміні даних в заголовку ці дані додаються до шифрованого повідомлення під час шифрування в режимі GCM як додаткові дані, що дозволяє перевірити справжність даних. При отриманні великої кількості помилок, канал з’єднання вважається втраченим і для

повторного встановлення каналу зв’язку потрібно виконання процедури погодження з самого початку.

Для реалізації логіки роботи протоколу було створено інтерфейс, що складається з функції обробки узгодження ключа та функції шифрування/дешифрування даних.

Функція обробки стадій узгодження ключів називається process_stage, приймає на вхід буфер з даними отриманими з бездротового інтерфейсу. Очікується, що в цьому буфері знаходяться повідомлення, які починаються з заголовку повідомлення та містять дані після нього, довжина буферу з даними, буфер для даних, що має бути відправлені іншій стороні і максимальний можливий розмір даних, що зможе туди поміститися.

Функція обробляє повідомлення на вході в залежності від даних в заголовку повідомленні та стадії каналу передачі даних на якому знаходиться з’єднання в момент обробки. Інформація про стадію каналу, а, також, допоміжні змінні, що використовуються різними криптографічними алгоритмами в момент обробки повідомлень. В залежності від стану повідомлення може відбуватися створення нового ключа сесії, індикація стану каналу передачі даних або інші стани повідомлення. В залежності від стадії створення каналу передачі даних, з отриманими пакетами проводяться відповідні дії.

Перевірка сертифікатів, генерація спільного статичного секрету з використанням “Elliptic Curve Diffie-Hellman”, динамічного секрету сесії з використанням “Diffie-Hellman-Merkle” та поліпшення стійкості секретів сесії реалізовані та відбуваються саме в цій функції. В залежності від стану каналу передачі даних функція обробляє відправку повідомлень про помилки. Функція повертає розмір даних, що були прочитані з вхідного буферу та даних, що були записані в вихідний буфер, повертаються через посилання на відповідні змінні, які були передані як параметри для цієї функції. В залежності від результату обробки повідомлення функція також повертає код результату виконання.

Функція шифрування та дешифрування даних називається process_cipher. Вона приймає на вхід буфер з даними для шифрування або дешифрування,

розмір цих даних та адреса вихідного буферу в який повинно бути записано результат виконання шифрування/дешифрування, максимальний розмір даних, що може бути записано в цей буфер, та необхідна операція.

Функція шифрує та шифрує інформацію в залежності від обраної функції з використанням ключа, який в цей момент використовується для забезпечення шифрованості каналу передачі даних. Обробка даних відбувається зважаючи на порядковий номер генерації сесійного ключа. Якщо порядковий номер в заголовку повідомлення менший за той, що зберігається в структурі, повідомлення вважається протермінованим і функція повертає стан похибки.

Дана функція запускається при отриманні інформації про натискання клавіш користувачем на клавіатурі або через функцію обробки стадій, яка попередньо відсіює всі повідомлення, ідентифікатор котрих менше за номер збережений в структурі даних каналу комунікації. Зважаючи на використовуємий в той момент часу номер генерації ключа сесії. При умові, що повідомлення є застарілим або функція шифрування-дешифрування повертає помилку протилежній стороні відправляється повідомлення з станом похибки.

Функція шифрування та дешифрування даних повертає розмір оброблених даних та розмір записаних в вихідний буфер даних через посилання на відповідні змінні, що були передані як параметри для цієї функції. В залежності від результату обробки функція також повертає код результату виконання.

Також для забезпечення функціонування реалізовані функція ініціалізації пам’яті під структуру даних яка називається protocol_init та функція скидання стану структури даних каналу зв’язку яка називається protocol_reset. Функція ініціалізації запускається під час ініціалізації пристрою, а функція скидання – викликається під час скидання стану каналу передачі даних до першої стадії. Це обумовлено необхідністю очистки елементів структури, що використовується криптографічною бібліотекою.

Допоміжною функцією також виступає функція відправки особистого сертифікату під назвою send_own_cert яка отримує на вхід адресу вихідного буфера та максимальний розмір даних що може там помістится. Ця функція

записує в вихідний буфер власний сертифікат пристрою і повертає керування, вже після повернення керування в логіку самого пристрою дані з цього буферу буде надіслано через канал передачі даних на інший пристрій.

Сам сертифікат повинен бути згенерований стороннім програмним забезпеченням, наприклад OpenSSL [27] підписаний ним з використанням спільного попередньо згенерованого центру сертифікації. Інформація про константи, за якими можна звернутися до сертифікату, розташовано в файлі “protocol_cert.h” Включивши цей файл в основний файл з реалізацією протоколу можна звертатися до власного сертифікату та сертифікату центру сертифікації, що потрібен для перевірки правильності підпису власного сертифікату іншої сторони.

Функція відправки пакету з похибкою називається send_error та приймає на вхід адресу вихідного буферу та максимальний розмір даних що можуть там вміститися. Ця функція формує заголовок що містить біт індикатора помилки в в полі статусу та повертає керування.

Існують також функції відправки ping та pong пакетів вони називаються відповідно send_ping та send_pong обидва приймають на вхід адресу вихідного буферу та максимальну довжину даних що може там вміститися. Функція send_ping використовується на приймачі для відправки запиту присутності, а функція send_pong використовується на передавачі в цьому випадку клавіатурі щоб відправити відповідь з інформацією про присутність.

Функція встановлення ключа шифрування називається set_encryption_key та приймає на вхід адресу ключа та його довжину. Задача цієї функції оновити криптографічний контекст що використовується під час шифрування та дешифрування встановити вектор ініціалізації та відновити стан внутрішніх структур підготувавши криптографічний контекст для отримання нових даних. Якщо під час встановлення ключів виникають помилки функція повертає керування з передачею помилки ERR_INTERNAL_ERROR.

Функція відправки зашифрованих корисних даних що використовується саме для відправки інформації про клавіші котрі натискає користувач в контексті

створеного пристрою називається send_payload_data і використовується як функція яку викликає логіка пристрою отримуючи натискання клавіші користувачем і передає цій функції інформацію про вихідний буфер, максимально можливу довжину даних що може там помістится, а також інформацію про натиснуту клавішу що вже попередньо сформована та довжину цієї інформації. Ця функція формує заголовок і викликає функцію шифрування що власне і шифрує отримані дані про натиснуту клавішу. Після повернення керування сформоване повідомлення відправляється через канал передачі даних іншій стороні.

Створений протокол не має проблем, які існують в стандартних засобах забезпечення захисту технології Bluetooth low Energy. Крім того, новий протокол не потребує участі людини в процесі узгодження каналу комунікації та використання інших фізичних каналів передачі даних. Також новий протокол є стійким до більшості атак а саме:

1. Стійкість до атаки пасивного перехоплення даних: Подібна атака може надати інформацію для атакуючого на самому початку передачі даних, в такому разі може бути добута інформація про сертифікати девайсів, але оскільки, приватні ключі залишаються невідомими для атакуючого, скомпрометувати канал передачі даних неможливо. В майбутньому, після узгодження спільних секретів, трафік, який може бути захоплений атакуючим, буде зашифровано, єдиною інформацією що може бути отримана в такому разі буде інформація з заголовку, що може надати представлення котрий по рахунку секрет сесії використовується, як часто вони генеруються, який по рахунку пакет надіслано і стан з’єднання. Подібна інформація не надасть атакуючому можливості скомпрометувати канал передачі даних, але може бути використано для аналізу кількості переданих даних.

2. Стійкість до атаки викрадення ідентифікаторів: Подібна атака може бути використана як перша стадія більш складної атаки Man-in-the-Middle, але, оскільки, обидва девайси проходять авторизацію подібна атака з підміною ідентифікаторів не дозволить атакуючому скомпрометувати процес створення

спільного секрету через те, що авторизація з проміжним девайсом не буде пройдена, а часте надсилання рекламних пакетів дозволить визначити правильний девайс. Це робить виконання подібної атаки безрезультатним.

3. Стійкість до атаки двійника: Подібна атака схожа на атаку підміни ідентифікаторів, але націлена саме на те щоб змусити підключитися до фейкового девайсу. В такій атаці має досягатися відмова в обслуговуванні однієї з легетимних сторін щоб інша сторона гарантовано підключилася до фейкового девайсу. Прикладом подібної атаки в контексті зробленого девайсу може бути атака на клавіатуру, щоб отримати інформацію про натискання клавіш користувачем або атака на приймач, щоб передати фіктивні натискання клавіш, які можуть виконати шкідливі дії. Проте новий протокол примусово авторизує обидві сторони комунікації з використанням сертифікатів підписаних спільним центром сертифікації. Оскільки в атакуючого немає можливості створити сертифікат, який буде підписаний тим самим центром сертифікації що й легітимні девайси, реалізація подібної атаки проти девайсів з новим протоколом не матиме результатів.

4. Стійкість до атаки Man-in-the-Middle: Подібна атака може бути проведена для захоплення каналу передачі даних через примушування з’єднатися з проміжним девайсом та створити з ним захищене з’єднання. Попередньою потребою для атакуючого в такому разі є виконання реверс інжинірингу протоколу узгодження каналу передачі даних. Але, оскільки протокол потребує передачі валідних сертифікатів, що були завірені спільним центром сертифікації, така атака не матиме успіху тому, що атакуючий немає можливості створити сертифікат, підписаний тим же центром сертифікації, що й сертифікати валідних девайсів. Тому така атака може використовуватися для копіювання зашифрованих даних, або для подовження радіусу дії, в випадку, якщо використовується декілька посередніх девайсів, які пересилають дані між собою через мережу інтернет. Подібний варіант атаки може використовуватися, наприклад, для відкриття дверей автомобіля, що має бездротове Bluetooth Low Energy з’єднання з ключами. Для цього в атакуючого має бути спільник. Один

посередній девайс має знаходитися поряд з ключами, а інший - біля автомобіля. Проміжні девайси пересилатимуть інформацію між собою через мережу інтернет таким чином умовно збільшуючи дистанцію роботи технології Bluetooth Low Energy. Подібного роду атака може мати сенс лише в наведеному прикладі з ключами, але від такої загрози складно, а інколи, неможливо захиститися, оскільки немає можливості визначити на якій насправді дистанції знаходяться ключі. Іншим можливим варіантом виконання такої атаки є зміна даних в заголовці пакетів. Таким чином можна завадити нормальній комунікації між девайсами, але оскільки інформація в заголовках є частиною тега шифрування, а зміна інформації призведе до неправильного тегу, не існує можливості змінювати зашифровану інформацію. Отже такий пакет буде вважатися помилковим. Велика кількість помилкових пакетів призведе до сбросу прогресу до стадії один, отже, подібна атака не дозволяє скомпрометувати канал передачі даних.

Єдиною атакою, що все ще можлива в повному обсязі, є атака на відмову в обслуговуванні,. Можливостей для її реалізації декілька. Як мінімум, можна заблокувати з’єднання фіктивними запитами і, в такому разі, девайс не зможе відповісти реальній стороні. Також можливо додавати помилки в інформацію, що передається між девайсами, оскільки вся вона так чи інакше автентифікован. Редагування цієї інформації в процесі передачі призведе до її інтерпретування як помилкової і скидання каналу передачі даних. Найбільш класичним варіантом є надсилання великої кількості запитів, що повинні бути оброблені дивайсом. В такому разі легітимні пакети можуть бути проігноровані. Проблемою є те, що захиститися від подібних атак неможливо через те, що дані проходять через спільне середовище і будь-хто може відправити дані. Атаки на відмову в обслуговуванні схожі на використання спеціальних технічних пристроїв для глушіння радіосигналу. В обох випадках така атака зробить неможливим комунікацію з девайсом. Єдиною причиною використовувати саме такі атаки на відмову в обслуговуванні, на відміну від використання генераторів перешкод, є можливість реалізації на базі звичайних Bluetooth Low Energy передавачів, а

також, меньшим використанням енергії і відсутності завад для інших пристроїв. Тому подібна атака може бути використана як частина атаки двійника.

3.3 Висновок до третього розділу

Створений протокол передачі даних за технологією Bluetooth Low Energy забезпечує створення надійного каналу передачі даних між девайсами на основі шифрування з використанням спільного секрету та коду авторизації повідомлення. Створення спільного секрету сесії, що саме використовується для шифрування пакетів корисного навантаження, відбувається через попередньо захищений канал даних, який був створений на основі приватних і публічних еліптичних ключів, що є частиною сертифікатів завірених з використанням спільного центру сертифікації. Таким чином забезпечується автеризованість і конфіденційність каналу передачі даних.

Протокол не має проблем, які існують при використанні класичних механізмів захисту, реалізованих в стандарті технології. Протокол захищає канал передачі даних від більшості атак на технологію Bluetooth Low Energy та є гарною альтернативою стандартним механізмам захисту каналу передачі даних для реалізації в будь-яких бездротових девайсах.

ВИСНОВКИ

З метою покращення безпеки бездротового безпеки каналу передачі даних між девайсами, що використовують бездротову технологію Bluetooth Low Energy, задля поліпшення цілісності та конфіденційності даних, що передаються такими девайсами, в роботі було розглянуто та розроблено захищений протокол безпечної передачі даних. Цей протокол використовує сучасні криптографічні протоколи, та методи для автентифікації сторін комунікації та узгодження спільного секрету.

Дослідження актуальне, оскільки в сучасному світі зростає потреба в захисті інформації від підроблення та несанкціонованого ознайомлення. Розробка протоколів для вирішення таких завдань є важливим напрямом, оскільки вони дозволяють підвищити ступінь захисту для систем будь-якого рівня, що використовують дану технологію.

Таким чином, у результаті виконання кваліфікаційної роботи була досягнута її мета - розробка захищеного протоколу безпеки для бездротового з’єднання. В роботі було виконано наступне:

1. Досліджено предметну область. Оглянуто технологію Bluetooth Low Energy та можливі загрози для для неї.

2. Проаналізовано існуючі в технології Bluetooth Low Energy засоби захисту, їх переваги та недоліки

3. Розроблено власний засіб захисту для технології Bluetooth Low Energy, алгоритму роботи та практичного девайсу для демонстрації роботи нового протоколу.

У результаті дослідження була проаналізована технологія Bluetooth Low Energy, можливості її використання в сучасному світі та її переваги. Були визначені основні загрози для безпеки даних, що передаються з використанням технології та можливий вплив цих атак на різні сфери, які використовують дану технологію. Були проаналізовані існуючі механізми захисту каналу передачі даних при використанні цієї технології, їх переваги та недоліки, можливість

використання в різних сферах та причини прийнятності для використання в деяких випадках. Було проаналізовано необхідні механізми та криптографічні протоколи, що можуть надати необхідний рівень захисту для даних, що передаються технологією, розроблено алгоритм аутентифікації сторін, узгодження спільного ключа та додатковий захист для унеможливлення атак на збережений трафік в майбутньому.

В практичній частині було реалізовано розроблений протокол, описано технічні моменти його реалізації та розроблено девайс, що демонструє можливість використання описаного протоколу для забезпечення безпеки передачі даних користувацького вводу.

У частині протоколу було використано такі сучасні криптографічні протоколи, як авторизація на базі перевірки підпису сертифікату. Протоколи узгодження спільного секрету через незахищене з'єднання на такі як класичний протокол Diffie-Hellman-Merkle та версія, що використовує в своїй основі криптографію на основі еліптичних кривих Elliptic Curve Diffie-Hellman. В якості протоколу шифрування було використано AES з ключем 256 біт та режимом GCM (Galois/Counter Mode), що надає гарний рівень стійкості до атак на шифрування, а також, перед атаками повного перебору ключа. До того ж, використаний режим шифрування дозволяє додати та авторизувати не зашифровані дані, що в випадку створеного протоколу являють собою заголовки повідомлення, що використовуються для зменшення впливу деяких атак на протокол.

Використання деяких інших криптографічних механізмів дозволили покращити безпеку, зменшити можливість зламу створеного протоколу та зменшити розміри кінцевої програми для використання в вбудованих системах. Такими механізмами є механізми збору ентропії та створення криптографічно безпечних випадкових чисел, використання формату збереження сертифікатів pem та кодування збережених даних з використанням ASN.1

Для реалізації криптографічних механізмів та алгоритмів в кінцевій реалізації протоколу було використано криптографічну бібліотеку MbedTLS, яка

є достатньо популярною, особливо, для використання у вбудованих системах. Використання цієї бібліотеки пришвидшило розробку реалізації протоколу в кінцевому девайсі та надало практичного вигляду реалізації протоколу.

Отримані в ході роботи результати мають практичне значення, оскільки розвивають захист бездротового з’єднання на основі протоколу Bluetooth Low Energy та можуть бути основою для створення подібних протоколів для інших технологій, а також, сприяють покращенню безпеки в різних сферах, зокрема, використання створеного протоколу буде особливо корисна в використанні з медичними датчиками, системами контролю доступу та автоматизації.

Отже, розроблений протокол захисту каналу передачі даних в бездротових технологіях є важливим кроком покрашення безпеки подібних технології та особливо покращенню безпеки девайсів, що потребують низького електроспоживання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Colbach G. Bluetooth Tutorial: Design, Protocol and Specifications for BLE - Bluetooth Low Energy 4. 0 and Bluetooth 5. 2023. pp. 12–27.

2. Hariharan M. Master Thesis Simulation Models for the Performance Analysis of Bluetooth Low Energy In Multi-device Environment. 2017. pp. 16-22.

3. Heydon R. Bluetooth Low Energy The Developer's Handbook. 2012. - 218 p.

4. Arup B., Md. Abdullah A., Md. Shohrab H., Ekram H. Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey. pp. 255–266.

5. Developer Study Guide: Bluetooth Low Energy Security. 2023. URL: https://www.slideshare.net/slideshow/bluetooth-le-security-study-guide-v11pdf/266648142/

6. Parthavi P. Bluetooth Low Energy (BLE) Security and Privacy for IoT. 2023. URL: https://www.einfochips.com/blog/bluetooth-low-energy-ble-security-and-privacy-for-iot/

7. Nthatisi Hlapisi. Vulnerabilities and Attacks on Bluetooth LE Devices—Reviewing Recent Info. 2023. URL: https://www.allaboutcircuits.com/technical-articles/vulnerabilities-and-attacks-on-bluetooth-le-devicesreviewing-recent-info/

8. Sławomir Jasek, SecuRing. Gattacking bluetooth smart devices. 2016. pp. 5-12. URL: https://www.blackhat.com/docs/us-16/materials/us-16-Jasek-GATTacking-Bluetooth-Smart-Devices-Introducing-a-New-BLE-Proxy-Tool-wp.pdf

9. Rosa, T. Bypassing Passkey Authentication in Bluetooth Low Energy. 2013 URL: https://eprint.iacr.org/2013/309.pdf

10. Duque A. Deep Dive into Bluetooth LE Security. 2018. URL: https://medium.com/rtone-iot-security/deep-dive-into-bluetooth-le-security-d2301d640bfc

11. Офіційний сайт NORDIC. nRF52 DK. Product Specification. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s132.api.v6.0.0%2Fmodules.html

12. Офіційний сайт NORDIC. nRF52832 Product Specification. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s132.api.v6.0.0%2Fmodules.html

13. Офіційний сайт NORDIC. S132 SoftDevice v7.0.1 API. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s132.api.v6.0.0%2Fmodules.html

14. Офіційний сайт NORDIC. nRF52840 DK. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s132.api.v6.0.0%2Fmodules.html

15. Офіційний сайт NORDIC. S140 SoftDevice Specification. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.s132.api.v6.0.0%2Fmodules.html

16. Mbed TLS documentation hub. URL: https://mbed-tls.readthedocs.io/en/latest/

17. Mbed TLS API documentation. URL: https://mbed-tls.readthedocs.io/projects/api/en/development/

18. Ueli M. Maurer та Stefan Wolf. The Diffie-Hellman Protocol. Des. Codes Cryptography. 2000. - 19 p. URL: https://link.springer.com/article/10.1023/A:1008302122286

19. Cryptography. Absolute beginners. Message Authentication. URL: https://www.tutorialspoint.com/cryptography/message_authentication.htm

20. Lemsitzer S.; Wolkerstorfer J.; Felber, N. Braendli M. 2007. Multi-gigabit GCM-AES Architecture Optimized for FPGAs. 2007. In Paillier, P.; Verbauwhede, I. (eds.). Cryptographic Hardware and Embedded Systems - CHES 2007. Lecture Notes in Computer Science. Vol. 4727. Springer. pp. 227–229.

21. McGrew, David A.; Viega, John. The Galois/Counter Mode of Operation (GCM). 2005. - p. 5. URL: https://csrc.nist.gov/projects/block-cipher-techniques/bcm

22. Gomez C, Oller J, Paradells J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. 2012. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3478807/

23. Nthatisi Hlapisi. Understanding Security Keys in Bluetooth Low Energy. 2023. URL: https://www.allaboutcircuits.com/technical-articles/understanding-security-keys-in-bluetooth-low-energy/

24. Wei-Meng. Programming Bluetooth Low Energy. 2016. URL: https://av.tib.eu/media/50522

25. Yahya Tawil. Bluetooth Low Energy (BLE) 101 Tutorial: Intensive Introduction. URL: https://atadiat.com/en/e-bluetooth-low-energy-ble-101-tutorial-intensive-introduction/

26. Bluetooth Low Energy Software Developer's Guide. Bluetooth Low Energy Protocol Stack. URL: https://software-dl.ti.com/lprf/simplelink_cc2640r2_sdk/1.00.00.22/exports/docs/blestack/html/ble-stack/index.html

27. OpenSSL Documentation. URL: https://www.openssl.org/docs/

28. Getting started with Bluetooth Low Energy development. URL: https://6point6.co.uk/insights/getting-started-with-bluetooth-le-development/

29. Afaneh M. Bluetooth Low Energy: A Primer. URL: https://interrupt.memfault.com/blog/bluetooth-low-energy-a-primer

30. Ashish S. ESP32 Bluetooth Low Energy Tutorial with ESP-IDF: Menuconfig and Code Implementation Explained. URL: https://innovationyourself.com/esp32-bluetooth-low-energy-tutorial/

ДОДАТОК А

Файл protocol.h

```
#ifndef PROTOCOL_H
#define PROTOCOL_H


enum errcodes
{
ERR_RESET = -6,
ERR_INTERNAL_ERROR = -5,
ERR_WRONG_STAGE = -4,
ERR_WRONG_DATA = -3,
ERR_NO_MEM = -2,
ERR_BAD_PARAM = -1,
ERR_OK = 0,
ERR_DATA = 1,
} prot_func_errcodes_t;


int protocol_init();
int protocol_reset();


int process_stage(const unsigned char * input,
     size_t ilen, unsigned char * output,
     size_t *olen, size_t * used);


int send_payload_data(const unsigned char * input, size_t *ilen,
      unsigned cahr * output, size_t * olen);
```

```
int process_cipher(const unsigned char * input,
    size_t ilen, const unsigned char * ad
    size_t adlen,
    unsigned char * output,
    size_t *olen, bool send);
#endif
```

ДОДАТОК Б

Файл protocol.c

```
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>
#include <stdbool.h>

#include "mbedtls/dhm.h"
#include "mbedtls/hkdf.h"
#include "mbedtls/cipher.h"
#include "mbedtls/entropy.h"
#include "mbedtls/ctr_drbg.h"

#include "protocol.h"
#include "protocol_cert.h"// заголовковий файл з ключем і сертифікатом

enum
{
STAGE_PRE_CERT = 0,
STAGE_CERT_1,
STAGE_DHM_1,
STAGE_DHM_2,
STAGE_WORK,
} protocol_stages;

struct prot_conn
{
```

```
int stage; // показчик стадії
uint32_t last_enc_gen; // номер останнього згенерованого спільного ключа
uint32_t last_msg_num; // номер останнього повідомлення
mbedtls_entropy_context entropy; // контекст ентропії
mbedtls_ctr_drbg_context drbg_context; // контекст генератора чисел
mbedtls_cipher_context_t aes_crypt; // контекст шифрування
mbedtls_cipher_context_t aes_decrypt; // контекст дешифрування
mbedtls_x509_crt cert; // ланцюг власного сертифікату
mbedtls_x509_crt remote_cert; // віддалений сертифікат
mbedtls_ecp_keypair keypair; // пара ключів для ecdh
mbedtls_ecp_keypair remote_keypair; // пара віддалених ключів для ecdh
mbedtls_ecdh_context ecdh_context; // контекст ecdh
mbedtls_dhm_context dhm_context; // контекст dhm
bool inited; // показчик ініціалізованості
bool usb_side; // показчик сторони з’єдання (клавіатура \ приймач)
};

enum prot_status
{
STATUS_ERROR = 0x1,
STATUS_NEW_SESSION = 0x2,
STATUS_MSG_DATA = 0x4,
STATUS_RESET_CONN = 0x8,
STATUS_PING = 0x10,
STATUS_PONG = 0x20,
} prot_status;

struct prot_msg_header
{
```

```
uint16_t status;
uint16_t session_secret_generation_number;
uint16_t message_iteration_number;
uint16_t payload_size;
};

struct prot_conn conn = {0}; // контекст з’єдання
const char iv = "012345670123"; // захардкоджений вектор ініціалізації

// Функція ініціалізації
int protocol_init(bool usb_side)
{
int res = 0;
const mbedtls_cipher_info_t * cipher_info = NULL;

// ініціалізація генератора чисел
mbedtls_entropy_init(&conn.entropy);
mbedtls_ctr_drbg_init(&conn.drbg_context);

// ініціалізація контексту сертифікатів
mbedtls_x509_crt_init(&conn.cert);
mbedtls_x509_crt_init(&conn.remote_cert);
mbedtls_pk_init(&conn.pkey);

// ініціалізація контексту шифрування
mbedtls_cipher_init(&conn.aes_crypt);
mbedtls_cipher_init(&conn.aes_decrypt);
```

```
// ініціалізація контексту криптографічних алгоритмів
mbedtls_ecdh_init(&conn.ecdh_context);
mbedtls_dhm_init(&conn.dhm_context);
mbedtls_ecp_keypair_init(&conn.keypair);
mbedtls_ecp_keypair_init(&conn.remote_keypair);

// обробка локальних сертифікатів
res = mbedtls_x509_crt_parse(&conn.cert,
        protocol_cert_ca,
        sizeof(protocol_cert_ca));
if (res != 0) return ERR_INTERNAL_ERROR;

res = mbedtls_x509_crt_parse(&conn.cert,
        protocol_cert_own,
        sizeof(protocol_cert_own));
if (res != 0) return ERR_INTERNAL_ERROR;

res = mbedtls_pk_parse_key(&conn.pkey,
      protocol_cert_pk,
      sizeof(protocol_cert_pk),
      NULL,
      0,
      mbedtls_ctr_drbg_random,
      &conn.drbg_context);
if (res != 0) return ERR_INTERNAL_ERROR;

res = mbedtls_ecp_read_key(MBEDTLS_ECP_DP_SECP256R1,
      &conn.key_pair,
      protocol_cert_pk,
```

```
            sizeof(protocol_cert_pk));
        if (res != 0) return ERR_INTERNAL_ERROR;

        // підготовка контексту шифрування
        cipher_info                                                    =
bmedtls_cipher_info_from_type(MBEDTLS_CIPHER_AES_256_GCM);
        if (cipher_info = NULL) return ERR_INTERNAL_ERROR;

        res = mbedtls_cipher_setup(&conn.aes_crypt, cipher_info);
        if (res != 0) return ERR_INTERNAL_ERROR;
        res = mbedtls_cipher_setup(&conn.aes_decrypt, cipher_info);
        if (res != 0) return ERR_INTERNAL_ERROR;

        conn.usb_side = usb_side;
        conn.inited = true;
        return ERR_OK;
        }

        // Функція сбросу контексту з'єднання
        int protocol_reset()
        {
        mbedtls_ecdh_free(&conn.ecdh_context);
        mbedtls_dhm_free(&conn.dhm_context);
        mbedtls_x509_crt_free(&conn.remote_cert);
        mbedtls_ecp_keypair_free(&conn.remote_keypair);

        conn.stage = 0;
        conn.last_enc_gen = 0;
        conn.last_msg_num = 0;
```

```
mbedtls_ecdh_init(&conn.ecdh_context);
mbedtls_dhm_init(&conn.dhm_context);
mbedtls_x509_crt_init(&conn.remote_cert);
mbedtls_ecp_keypair_init(&conn.remote_keypair);

return ERR_OK;
}

// функція відправик повідомлення з маркером похибки
static int send_error(unsigned char * output, size_t * olen)
{
struct prot_msg_header * header = (struct prot_msg_header *)output;

if (*olen < 8)
{
  *olen = 0;
  return ERR_BAD_PARAM;
}

memset(output, 0, 8);
*header = (struct prot_msg_header) {
  .status = 0 | STATUS_ERROR,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
  .payload_size = 0,
};

conn.last_msg_num += 1;
```

```
*olen = 8;
return 0;
}

// функція відправик повідомлення з маркером ping
static int send_ping(unsigned char * output, size_t * olen)
{
struct prot_msg_header * header = (struct prot_msg_header *)output;

if (*olen < 8)
{
  *olen = 0;
  return ERR_BAD_PARAM;
}

memset(output, 0, 8);
*header = (struct prot_msg_header) {
  .status = 0 | STATUS_PING,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
  .payload_size = 0,
};

conn.last_msg_num += 1;

*olen = 8;
return 0;
}
```

```
// функція відправик повідомлення з маркером pong
static int send_pong(unsigned char * output, size_t * olen)
{
struct prot_msg_header * header = (struct prot_msg_header *)output;

if (*olen < 8)
{
  *olen = 0;
  return ERR_BAD_PARAM;
}

memset(output, 0, 8);
*header = (struct prot_msg_header) {
  .status = 0 | STATUS_PONG,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
  .payload_size = 0,
};

conn.last_msg_num += 1;

*olen = 8;
return 0;
}

// функція відправки власного сертифікату
static int send_own_cert(unsigned char * output, size_t * olen)
{
```

```
struct prot_msg_header * header = (struct prot_msg_header *)output;


if (*olen < sizeof(protocol_cert_own) + 8)
{
  *olen = 0;
  return ERR_BAD_PARAM;
}


memset(output, 0, sizeof(protocol_cert_own) + 8);
*header = (struct prot_msg_header) {
  .status = 0,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
  .payload_size = sizeof(protocol_cert_own),
};



conn.last_msg_num += 1;
*olen = sizeof(protocol_cert_own) + 8;


return 0;
}


// функція встановлення ключа шифрування
int set_encryption_key(unsigned char * key, size_t len)
{
int res = 0;
```

```
    res = mbedtls_cipher_setkey(&conn.aes_crypt, key, len,
MBEDTLS_ENCRYPT);
    if (res != 0) return ERR_INTERNAL_ERROR;
    res = mbedtls_cipher_setkey(&conn.aes_decrypt, key, len,
MBEDTLS_DECRYPT);
    if (res != 0) return ERR_INTERNAL_ERROR;

    res = mbedtls_cipher_set_iv(&conn.aes_crypt, iv, sizeof(iv));
    if (res != 0) return ERR_INTERNAL_ERROR;

    res = mbedtls_cipher_set_iv(&conn.aes_decrypt, iv, sizeof(iv));
    if (res != 0) return ERR_INTERNAL_ERROR;

    // reset функція повинна бути виконана після встановлення ключа
    // згідно з документацією
    res = crypt_cipher_reset(&conn.aes_crypt);
    if (res != 0) return ERR_INTERNAL_ERROR;
    res = crypt_cipher_reset(&conn.aes_decrypt);
    if (res != 0) return ERR_INTERNAL_ERROR;

    conn.last_enc_gen += 1;
    conn.last_msg_num = 0;

    return 0;
    }

    int process_stage(const unsigned char * input,
        size_t ilen, unsigned char * output,
        size_t * olen, size_t *used)
```

```
{
int res = 0;
uint32_t flags = 0;
unsigned char * key;
size_t key_len;
char ecdh_secret[32] = {0};
size_t len = 0;
size_t ecdh_secret_len = 0;
char dhm_buf[512] = {0};
char session_key[256] = {0};

const mbedtls_md_info_t * md_type;
struct prot_msg_header * header = (struct prot_msg_header *)input;

if (input == NULL || output == NULL || ! conn.inited)
  return ERR_BAD_PARAM;

switch (conn.stage)
{
  case STAGE_PRE_CERT:
   conn.stage += 1;
   if (conn.usb_side)
   {
    // приймач посилає сврій власний сертифікат
    res = send_own_cert(output, olen);
    return res;
   }
   // клавіатура нічого не робить на цій стадії
   // no break
```

```
case STAGE_CERT_1:
 if (ilen < header->payload_size + 4) return 0; // wait for all data
 used = header->payload_size + 4; // кількість байтів було прочитано

 res = mbedtls_x509_crt_parse(&conn.remote_cert,
        input + 4,
        header->payload_size);
 if (res != 0)
 {

  send_error(output, olen);
  return 0;
 }

 res = mbedtls_x509_crt_verify(&conn.remote_crt,
        &conn.crt,
        NULL,
        NULL,
        &flags,
        mbedtls_ctr_drbg_random,
        &conn.drbg_context);
 if (res != 0 || flags != 0)
 {
  send_error(output, olen);
  return 0;
 }

 res = mbedtls_ecp_read_key(MBEDTLS_ECP_DP_SECP256R1,
```

```
        &conn.remote_keypair,
        protocol_cert_pk,
        sizeof(protocol_cert_pk));
if (res != 0)
{
 send_error(output, olen);
 return 0;
}

res = mbedtls_ecdh_get_params(&conn.ecdh_context,
          &conn.remote_keypair,
          MBEDTLS_ECDH_THEIRS);
if (res != 0)
{
 send_error(output, olen);
 return 0;
}

res = mbedtls_ecdh_calc_secret(&conn.ecdh_context,
        &ecdh_secret_len
        &ecdh_secret,
        sizeof(ecdh_secret),
        mbedtls_ctr_drbg_random,
        &conn.drbg_context);
if (res != 0)
{
 send_error(output, olen);
 return 0;
}
```

```
md_type = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);

res = mbedtls_hkdf(md_type,
    NULL,
    0,
    &ecdh_secret,
    ecdh_secret_len,
    NULL,
    0,
    &session_key,
    sizeof(session_key));
if (res != 0)
{
 send_error(output, olen);
 return 0;
}

res = set_encryption_key(session_key, sizeof(session_key));
if (res != 0)
{
 send_error(output, olen);
 return 0;
}

if (! conn.usb_side)
{
 // клавіатура надислає свій власний сертифікат
 res = send_own_cert(output, olen);
```

```
 return 0;
 }

 // приймач нічого не робить на цій стадії
 conn.stage += 1;
 // no break here

case STAGE_DHM_1:
 len = sizeof(dhm_buf);
 res = mbedtls_dhm_make_params(&conn.dhm_context,
         128,
         dhm_buf,
         &len,
         mbedtls_ctr_drbg_random,
         &conn.drbg_context);
 if (res != 0)
 {
  send_error(output, olen);
  return 0;
 }

 header = (struct prot_msg_header *)output;
 *header = (struct prot_msg_header) {
  .status = 0 | STATUS_DATA,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
 };

 // дані шифруються з використанням ключа
```

```
res = process_cipher(&dhm_buf, len, header, 6, output + 8, olen, true);
if (res != 0)
{
 send_error(output, olen);
 return 0;
}
header->payload_size = *olen;
*olen += 4;
conn.last_msg_num += 1;

// дані відсилаються
conn.stage += 1;
if (conn.usb_side)
{
 // приймач тепер буде очікувати на параметри іншої сторони
 return 0;
}

// клавіатура вже має параметри приймача тож переходить до наступної стадії
// no break here
case STAGE_DHM_2:
 if (ilen < header->payload_size + 4) return 0; // wait for all data
 // дані дешифруються з використанням ключа
 res = process_cipher(input + 8, header->payload_size, header, 6, dhm_buf, &len, false);
 if (res == ERR_WRONG_TAG)
 {
  // reset process
  conn.stage = 0;
```

```
 return ERR_RESET;
}
if (res != 0)
{
 send_error(output, len);
 return 0;
}

res = mbedtls_dhm_read_public(&conn.dhm_context,
        dhm_buf,
        len);
if (res != 0)
{
 res = send_error(output, len);
 return 0;
}

// вираховується ключ сесії
res = mbedtls_dhm_calc_secret(&conn.dhm_context,
        &session_key,
        sizeof(session_key),
        &len,
        mbedtls_ctr_drbg_random,
        &conn.drbg_context);
if (res != 0)
{
 res = send_error(output, len);
 return 0;
}
```

```
 md_type = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);

 res = mbedtls_hkdf(md_type,
    NULL,
    0,
    &key,
    len,
    NULL,
    0,
    &session_key,
    sizeof(session_key));
 if (res != 0)
 {
  send_error(output, olen);
  return 0;
 }

 set_encryption_key(session_key, sizeof(session_key));
 // встановлючеться як ключ сесії

 return 0;
case STAGE_WORK:
 if (conn.usb_side)
 {
  res = send_ping(output, len); // приймач відправлляє ping
  return 0;
 }
 else
```

```
   {
    res = send_pong(output, len); // клавіатура відповідає pong
    return 0;
   }
   return 0;

  default:
   return ERR_BAD_PARAM;
}

}

int send_payload_data(const unsigned char * input, size_t *ilen,
      unsigned char * output, size_t *olen)
{
int res = 0, len = 0;
struct prot_msg_header * header = (struct prot_msg_header *)output;

olen -= 8;
*header = (struct prot_msg_header) {
  .status = 0 | STATUS_DATA,
  .session_secret_generation_number = conn.last_enc_gen,
  .message_iteration_number = conn.last_msg_num,
};
res = process_cipher(input, ilen, header, 6, header + 8, olen);
if (res != 0)
{
  return ERR_INTERNAL_ERROR;
}
```

```
header->payload_size = olen;
return 0;
}

// шифрування дешифрування та перевірка тегів
int process_cipher(const unsigned char * input,
    size_t ilen, const unsigned char * ad,
    size_t adlen,
    unsigned char * output,
    size_t *olen, bool send)
{
int res = 0;
if (input == NULL || output == NULL || ! conn.inited)
  return ERR_BAD_PARAM;

if (send)
{
  res = mbedtls_cipher_update(&conn.aes_crypt,
        input,
        ilen,
        output,
        olen);
  if (res != 0)
  {
   return ERR_INTERNAL_ERROR;
  }

  res = mbedtls_cipher_update_ad(&conn.aes_crypt,
        ad,
```

```
        adlen,
        output,
        olen);
  if (res != 0)
  {
   return ERR_INTERNAL_ERROR;
  }

  res = mbedtls_cipher_finish(&conn.aes_crypt,
        output,
        olen);
  if (res != 0)
  {
   return ERR_INTERNAL_ERROR;
  }

  res = mbedttls_cipher_write_tag(&conn.aes_crypt,
         output,
         16);
  if (res != 0)
  {
   return ERR_INTERNAL_ERROR;
  }
}
else
{
   res = mbedtls_cipher_update(&conn.aes_decrypt,
        input,
        ilen,
```

```
        output,
        olen);
if (res != 0)
{
 return ERR_INTERNAL_ERROR;
}

res = mbedtls_cipher_update_ad(&conn.aes_decrypt,
        ad,
        adlen,
        output,
        olen);
if (res != 0)
{
 return ERR_INTERNAL_ERROR;
}

res = mbedtls_cipher_finish(&conn.aes_decrypt,
        output,
        olen);
if (res != 0)
{
 return ERR_INTERNAL_ERROR;
}

res = mbedttls_cipher_write_tag(&conn.aes_decrypt,
         output,
         16);
if (res != 0)
```

```
  {
   return ERR_WRONG_TAG;
  }
}

return 0;
}
```