Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

МОНІТОРИНГ ТА ВДОСОКОНАЛЕННЯ АЛГОРИТМІВ БЕЗПЕКИ ІОТ

MONITORING AND IMPROVING IoT SECURITY ALGORITHMS

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Романюк Олександр Юрійович

(підпис)

Керівник:
к.т.н., доцент
Мельник Катерина Вікторівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 06 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. ТЕРЛЕЦЬКИЙ
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Романюку Олександру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Моніторинг та вдосконалення алгоритмів безпеки IOT*

Керівник роботи *Мельник Катерина Вікторівна к.т.н., доц.*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02, наказ на поновлення для проходження атестації від «31» жовтня 2024 року № 807/01-07
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 12.12.2024р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Огляд літератури по основам безпеки IoT
Аналіз алгоритмів безпеки IoT
Апаратна реалізація алгоритмів безпеки на базі мікроконтролерів
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Рисунки з зображеннями мікроконтролерів та периферійних пристроїв, електричних схем, інтерфейсів користувача та лістингів коду.
Використання технології: Arduino IDE, C/C++

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд літератури по основам безпеки IoT	*Мельник К.В., доцент*		
Аналіз алгоритмів безпеки IoT	*Мельник К.В., доцент*		
Апаратна реалізація алгоритмів безпеки на базі мікроконтролерів	*Мельник К.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури по основам безпеки IoT*	до 10.02.2025 р.	Виконано
2.	*Аналіз алгоритмів безпеки IoT*	до 02.03.2025 р.	Виконано
3.	*Апаратна реалізація алгоритмів безпеки на базі мікроконтролерів*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
	Представлення остаточного варіанту кваліфікаційної роботи керівникові	до 15.05.2025 р.	
8.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Романюк О.Ю.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Мельник К.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Романюк О.Ю. Моніторинг та вдосконалення алгоритмів безпеки IoT. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, одного додатку.

У першому розділі розглядаються теоретичні основи безпеки в системах (IoT). У ньому розглянуто основи організації IoT-систем, їх архітектурні моделі та ключові принципи функціонування. Аналізуються характерні загрози та вразливості, пов'язані з IoT-мережами, включаючи ризики несанкціонованого доступу, вразливості в протоколах та апаратному забезпеченні.

Другий розділ присвячений аналізу алгоритмів безпеки в системах IoT. У ньому розглядаються методи оцінки ефективності алгоритмів, які забезпечують захист даних та пристроїв IoT. Аналізуються сучасні алгоритми шифрування, а також протоколи безпеки, які використовуються для забезпечення конфіденційності, цілісності та автентифікації в IoT-мережах, з урахуванням їхньої адаптованості до обмежених ресурсів IoT-пристроїв.

Третій розділ присвячений практичним аспектам реалізації алгоритмів безпеки на основі мікроконтролерів. У ньому розглядається вибір апаратних компонентів, що відповідають специфіці систем IoT. Детально аналізуються алгоритми шифрування, їх ефективність та адаптованість до ресурсів обраного обладнання. Описано критерії вибору програмного забезпечення, яке використовується для розробки і тестування пристроїв. Завершується описом розробки кінцевого пристрою, його тестуванням і оцінкою відповідності вимогам безпеки IoT.

Ключові слова: Arduino Uno, ESP32, шифрування, IoT, мікроконтролер, безпека, RFID, Arduino IDE.

ANNOTATION

Romanyuk O. Monitoring and improving iot security algorithms. Manuscript.

Bachelor’s qualification work OP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three sections, conclusions, a list of sources used, one appendix.

The first section examines the theoretical foundations of security in systems (IoT). It examines the basics of organizing IoT systems, their architectural models and key principles of functioning. The characteristic threats and vulnerabilities associated with IoT networks are analyzed, including the risks of unauthorized access, vulnerabilities in protocols and hardware.

The second section is devoted to the analysis of security algorithms in IoT systems. It examines methods for assessing the effectiveness of algorithms that provide protection for IoT data and devices. Modern encryption algorithms and security protocols used to ensure confidentiality, integrity, and authentication in IoT networks are analyzed, taking into account their adaptability to the limited resources of IoT devices.

The third section is devoted to practical aspects of implementing security algorithms based on microcontrollers. It considers the selection of hardware components that meet the specifics of IoT systems. Encryption algorithms, their efficiency, and adaptability to the resources of the selected equipment are analyzed in detail. The criteria for selecting software used for developing and testing devices are described. It concludes with a description of the development of the final device, its testing, and assessment of compliance with IoT security requirements.

Keywords: Arduino Uno, ESP32, encryption, IoT, microcontroller, security, RFID, Arduino IDE.

ЗМІСТ

ВСТУП

Актуальність теми. Інтернет речей (IoT) стрімко інтегрується в різні сфери життя, об’єднуючи мільярди пристроїв у глобальну мережу. Це сприяє підвищенню ефективності та комфорту, але водночас створює нові виклики в сфері кібербезпеки. Багато IoT-пристроїв мають обмежені обчислювальні ресурси та енергетичні можливості, що ускладнює впровадження традиційних заходів безпеки. Крім того, відсутність стандартів безпеки та недостатня увага до оновлень програмного забезпечення роблять ці пристрої вразливими до кібератак.

Згідно з дослідженням [1], основними проблемами безпеки IoT є ненадійні паролі, незахищені мережеві служби та недостатні заходи конфіденційності. Інше дослідження підкреслює, що навіть прості датчики можуть створювати загрози безпеці, якщо не дотримуватися належних практик кіберзахисту [2].

Вразливості IoT-пристроїв активно експлуатуються зловмисниками для створення ботнетів, проведення DDoS-атак та несанкціонованого доступу до конфіденційних даних. Наприклад, ботнет Mirai використовував тисячі скомпрометованих IoT-пристроїв для масштабних атак на інтернет-ресурси. Це свідчить про необхідність розробки та впровадження ефективних алгоритмів безпеки, адаптованих до специфіки IoT.

Таким чином, забезпечення безпеки IoT є актуальною проблемою, що вимагає комплексного підходу, включаючи моніторинг, вдосконалення існуючих алгоритмів та розробку нових рішень для захисту від сучасних кіберзагроз.

Метою роботи є розробка та впровадження вдосконалених алгоритмів безпеки для систем Інтернету речей (IoT), які забезпечують ефективний захист даних і пристроїв від сучасних кіберзагроз, з урахуванням обмежень енергоспоживання та обчислювальних ресурсів, а також побудова системи моніторингу для оцінки та підвищення рівня захищеності IoT-систем.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити сучасні загрози та вразливості IoT-систем;

– зпровести аналіз існуючих алгоритмів безпеки, оцінити їх ефективність і недоліки;

– розробити вдосконалений алгоритм безпеки, який забезпечує енергоефективність та високий рівень захисту;

– реалізувати систему шифрування даних для IoT та протестувати її ефективність.

Об’єктом дослідження є системи Інтернету речей і їх безпека.

Предметом дослідження є алгоритми безпеки та методи моніторингу захищеності IoT-систем.

Методологія дослідження базується на аналізі існуючих рішень у сфері IoT-безпеки, використанні симуляційних платформ для тестування, а також розробці та впровадженні власних алгоритмів і систем моніторингу.

Робота має практичну цінність, оскільки запропоновані рішення спрямовані на підвищення рівня безпеки IoT-систем, що може бути корисним для їх розробників, адміністраторів мереж та користувачів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ БЕЗПЕКИ IOT

1.1 Основи організації систем IoT та їх архітектура

Інтернет речей (Internet of Things, IoT) – це концепція мережі, що об’єднує фізичні пристрої, оснащені сенсорами, програмним забезпеченням та іншими технологіями, які дозволяють їм взаємодіяти між собою та з зовнішнім середовищем через Інтернет. Ця технологія забезпечує збір, обмін та аналіз даних, сприяючи автоматизації та підвищенню ефективності різних процесів.

IoT є дуже зручним та корисним набором технологій, що значно спрощує як наше повсякденне життя, так і роботу організацій. Проте, на жаль, ідеальних технологій просто не існує. Не зважаючи на шалену популярність та зручність пристроїв IoT, вони мають свої недоліки. Мова йде про наявність вразливостей, які важко вчасно визначати та відсутність стандартизації. Пам’ятаємо, що IoT мережі – це комплекси де вкрай рідко з’являється людина, відповідно – нікому відслідковувати нетипову ситуацію і віруси [2].

До 2025 року загальна кількість встановлених розумних пристроїв досягне відмітки 30,9 мільярда (рис. 1.1).

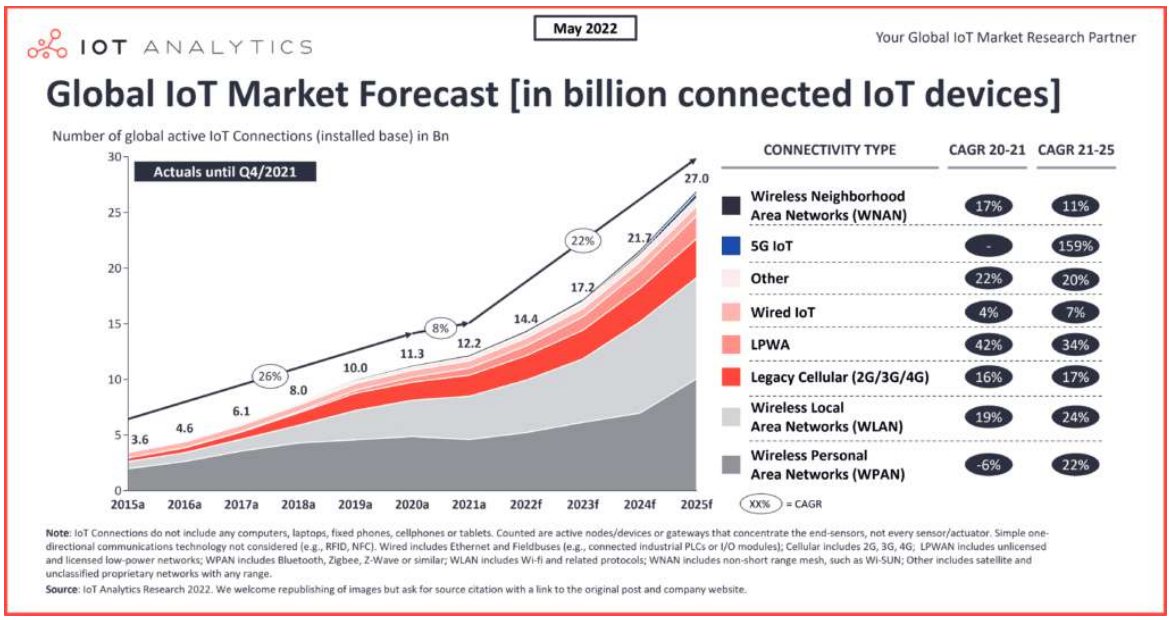

Рисунок 1.1 – Популярність IoT [2]

Архітектура IoT [3] поділяється на 4 різні рівні, тобто:

1) сенсорний рівень є першим рівнем архітектури Інтернету речей і відповідає за збір даних із різних джерел. Цей рівень містить датчики та виконавчі механізми , які розміщуються в навколишньому середовищі для збору інформації про температуру, вологість, світло, звук та інші фізичні параметри. Протоколи дротового або бездротового зв'язку з'єднують ці пристрої з мережевим рівнем;

2) мережевий рівень архітектури IoT відповідає за забезпечення зв'язку та підключення між пристроями в системі IoT. Він включає в себе протоколи та технології, які дозволяють пристроям підключатися та спілкуватися один з одним і з широким Інтернетом. Приклади мережевих технологій, які зазвичай використовуються в IoT, включають WiFi, Bluetooth, Zigbee та стільникові мережі, такі як технології 4G і 5G. Крім того, мережевий рівень може включати шлюзи та маршрутизатори, які діють як посередники між пристроями та широким Інтернетом, а також можуть містити функції безпеки, такі як шифрування та автентифікація для захисту від несанкціонованого доступу;

3) рівень обробки даних архітектури IoT відноситься до програмних і апаратних компонентів, які відповідають за збір, аналіз та інтерпретацію даних з пристроїв IoT. Цей рівень відповідає за отримання необроблених даних із пристроїв, їх обробку та надання доступу для подальшого аналізу чи дії. Рівень обробки даних включає різноманітні технології та інструменти, як-от системи керування даними, аналітичні платформи та алгоритми машинного навчання. . Ці інструменти використовуються для отримання значущої інформації з даних і прийняття рішень на основі цих даних. Прикладом технології, яка використовується на рівні обробки даних, є озеро даних, яке є централізованим сховищем для зберігання необроблених даних з пристроїв IoT;

4) прикладний рівень архітектури IoT є найвищим рівнем, який безпосередньо взаємодіє з кінцевим користувачем. Він відповідає за надання зручних інтерфейсів і функцій, які дозволяють користувачам отримувати доступ до пристроїв IoT і керувати ними. Цей рівень включає різноманітне програмне

забезпечення та програми, такі як мобільні програми, веб-портали та інші інтерфейси користувача, які призначені для взаємодії з основною інфраструктурою IoT. Він також включає служби проміжного програмного забезпечення, які дозволяють різним пристроям і системам IoT безперебійно спілкуватися та обмінюватися даними. Рівень додатків також включає аналітику та можливості обробки, які дозволяють аналізувати дані та перетворювати їх у значущі ідеї. Це може включати алгоритми машинного навчання, інструменти візуалізації даних та інші розширені аналітичні можливості.

На рисунку 1.2 зображено типову архітектуру системи IoT, показуючи взаємодію між різними рівнями та компонентами.

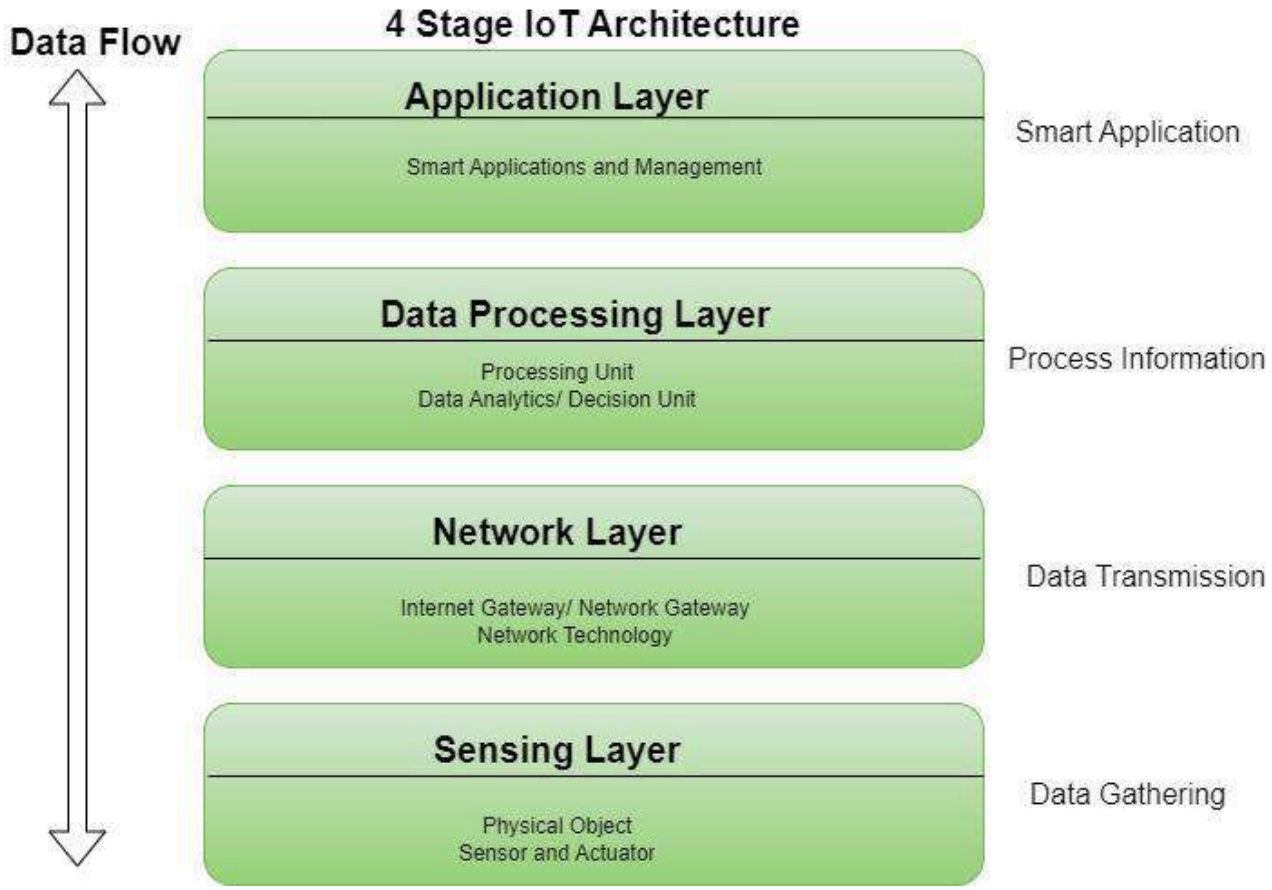

Рисунок 1.2 – Архітектура системи IoT [4]

Системи Інтернету речей (IoT) складаються з кількох ключових компонентів, які забезпечують їх ефективне функціонування. Розглянемо детальніше основні з них:

1) пристрої та сенсори – це фізичні об’єкти, оснащені сенсорами або виконавчими механізмами, які збирають дані з навколишнього середовища або виконують певні дії. Прикладами таких пристроїв є розумні лампи, термостати, фітнес-трекери та промислові датчики. Вони перетворюють фізичні явища, такі як температура, вологість або рух, на цифрові сигнали для подальшої обробки [5];

2) комунікаційні мережі, які забезпечують передачу зібраних даних від пристроїв до центрів обробки та назад. Для цього використовуються різні протоколи та технології зв’язку, такі як Wi-Fi, Bluetooth, Zigbee, LoRaWAN, мобільні мережі (3G, 4G, 5G) та супутникові канали. Вибір конкретної технології залежить від вимог до дальності передачі, енергоспоживання та обсягу даних [3];

3) шлюзи IoT, що виконують роль посередників між пристроями та хмарними сервісами. Вони агрегують дані з різних сенсорів, здійснюють попередню обробку та передають їх до хмари. Крім того, шлюзи можуть виконувати функції безпеки, фільтрації даних та управління пристроями [6];

4) платформи обробки та зберігання даних зазвичай розміщені в хмарі, відповідають за зберігання великих обсягів даних, їх обробку та аналіз. Вони використовують технології великих даних та машинного навчання для виявлення закономірностей, прогнозування та прийняття рішень на основі отриманої інформації [3];

5) інтерфейси користувача – це програмні додатки або веб-інтерфейси, які дозволяють користувачам взаємодіяти з IoT-системою. Через них користувачі можуть отримувати інформацію, налаштовувати параметри роботи пристроїв та контролювати їхній стан. Приклади включають мобільні додатки для керування розумним будинком або промислові панелі моніторингу [6].

На рисунку 1.3 показано взаємодію між основними компонентами системи IoT.

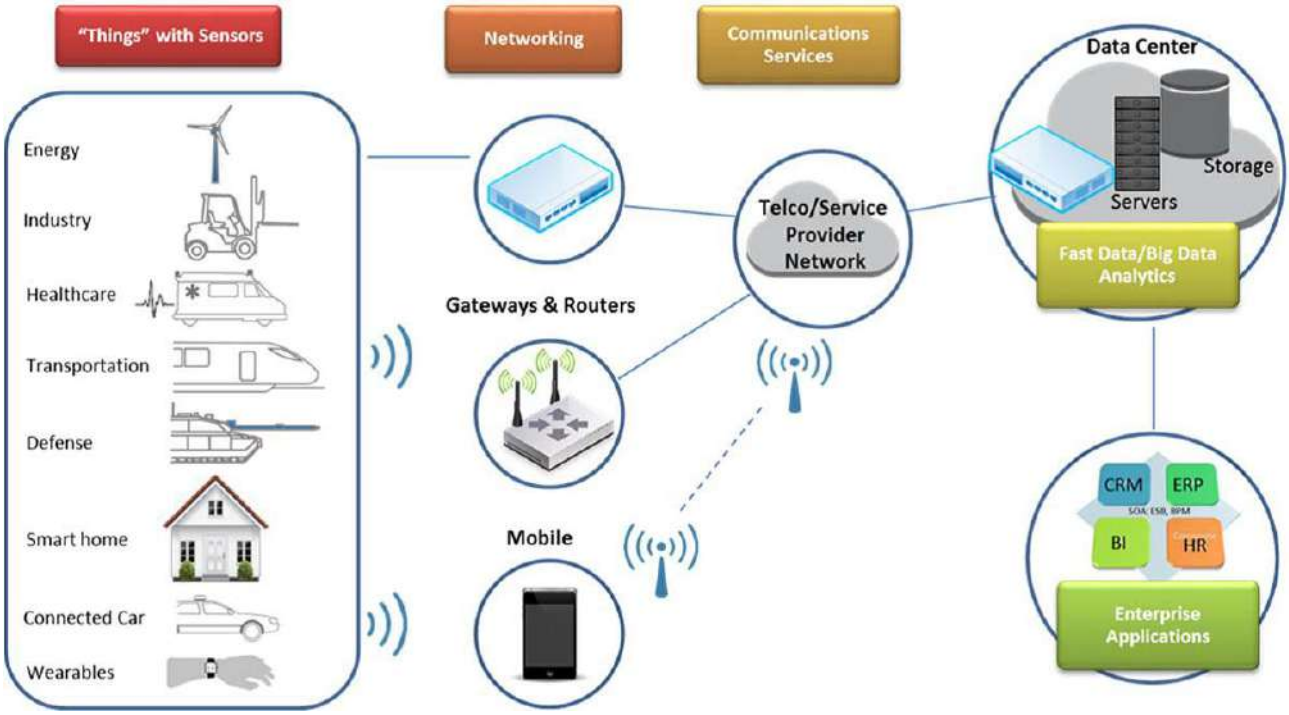

Рисунок 1.3 – Основні компоненти системи IoT [7]

Розуміння та правильна інтеграція цих компонентів є ключовими для створення ефективних та надійних IoT-систем, здатних забезпечити автоматизацію та оптимізацію процесів у різних сферах діяльності.

1.2 Загрози та вразливості в IoT-мережах

З розвитком технологій, кібербезпека пристроїв стає все більш критичним питанням. Особливо це стає очевидно в контексті Інтернету речей, оскільки, кількість пристроїв, які можуть отримувати та надсилати конфіденційні дані щоденно росте. Тож впровадження й оновлення заходів безпеки пристроїв має стати пріоритетом на найближчі роки для технології IoT.бездротової комунікації, що робить її ідеальним вибором для багатьох IoT-проєктів.

Захист пристроїв Інтернету речей є життєво важливим компонентом безпеки сучасної мережі в організаціях. Навіть такі, здавалося б, невинні пристрої, як, наприклад, прості датчики світла, можуть створювати загрози безпеки. IT-команди повинні включати ці ризики у свої протоколи кібербезпеки, та працювати над зведенням їх до мінімуму.

Без належних практик з безпеки IoT, компанії можуть зіткнутися з новими загрозами, що надходять з кіберпростору. Наприклад, зловмисники, які націлені на розумні пристрої, можуть отримати доступ до критичних ресурсів компанії далеко за межами їх початкової точки входу в мережу (через те, що вразливі розумні пристрої під'єднані до них). Це дасть їм змогу збирати конфіденційні дані або влаштовувати виснажливі для системи кібератаки.

Ризики IoT насправді легко не помітити, якщо не використовувати призначені для цього інструменти. Іноді офіцери з інформаційної безпеки нехтують інвентаризацією кінцевих точок, через це легко можна пропустити потенційно вразливий до атак пристрій, не приділивши йому достатньо уваги. Сьогодні існують програмні продукти для проведення інвентаризації та моніторингу всіх підключених IoT девайсів. Такий тип рішень з безпеки значно знижує ризики, аналізуючи всю поверхню потенційних атак [2].

Щоб запобігти атакам на пристрої IoT, перш за все треба знати які поширені види атак можуть використовувати кіберзлочинці для досягнення своїх цілей:

– DDoS-атака: відбувається коли ботнет – заражена мережа комп'ютерів – неперервно надсилає величезну кількість запитів до системи. Аномально висока активність може призвести до створення значних затримок у роботі системи або взагалі до її зупинки. Вдало скорегована та налаштована DDoS-атака може викликати системну помилку компонента безпеки, приховавши реальні шкідливі дії. Більш того, заражені пристрої IoT також можуть стати частиною ботнету та допомагати зловмисникам проводити ще більш руйнівні атаки зсередини локальної мережі, у якої зазвичай більший рівень довіри у систем інформаційної безпеки;

– експлойт програмного забезпечення: багато кіберзлочинців використовують вже відомі вразливості в програмній частині пристрою для проведення атаки. Розробники зазвичай закривають знайдені «діри» безпеки в оновленнях. Однак, далеко не завжди свіжі версії ПЗ вчасно завантажуються на пристрої. Саме це робить їх вразливими до атак з використанням експлойтів. Додатковою загрозою є те, що далеко не всі виробники приладів інформують

своїх користувачів про реальний технологічний стек ПЗ пристрою, мотивуючись ринковими стимулами (використання певної бібліотеки, яка є у відкритому доступі може бути конкурентною перевагою, якщо інші гравці ринку не здогадались її використати);

– MITM-атака (атака посередника): хакери можуть перехопити мережевий трафік (вставши посередині каналу передачі між пристроєм відправником і пристроєм отримувачем) і отримати облікові дані або конфіденційну інформацію, яку пристрої IoT передають через корпоративні мережі. Через те, що багато смарт пристроїв зазвичай навіть не зашифровано, зловмиснику буде дуже легко використати отримані дані для несанкціонованого доступу в систему;

– фізичне втручання: простого підключення кіберзлочинцем USB флешки зі шкідливим кодом, до зовнішнього пристрою IoT може бути достатньо, щоб розповсюдити зловмисне програмне забезпечення через мережу та шпигувати за комунікаціями що в ній проходять;

– брутфорс атаки: той факт, що в компаніях зазвичай не приділяється достатньо уваги парольній безпеці пристроїв IoT, робить їх вразливими до потенційних атак грубою силою або «Брутфорс». Часто паролі пристроїв IoT залишаються незмінними після встановлення просто використовуючи базовий пароль, що дозволяє зловмисникам дуже просто їх підбирати;

– викрадення прошивки: якщо оновлення мікропрограми пристрою не було криптографічно підписано або прошивка передається по не захищеному каналу зв’язку – це дає смогу зловмисникам перехопити її та завантажувати шкідливе ПЗ на пристрої під виглядом апдейтів. Також за допомогою викраденої прошивки у кіберзлочинців з’являється можливість отримати облікові дані пристрою. Використовуючи облікові дані, вони можуть отримати доступ до корпоративних мереж або інших систем, що зберігають конфіденційну інформацію. Таким чином атака на, здавалося б, невинний пристрій може перетворитися на повномасштабний витік даних [2].

З вищезазначених векторів атак на IoT можна зробити висновок, що основні компоненти систем Інтернету речей є досить вразливими до атак

зловмисників. Незалежно від масштабу та типу середовища, у яке вбудовується система IoT, безпека повинна розглядатися ще на етапі проєктування, щоб покращити її інтегрування. Особливим викликом для інженерів та офіцерів інформаційної безпеки є те, що через технологічні особливості IoT не дозволяються встановити агент для перевірки наявності заражень чи вразливостей.

Ось декілька основних рекомендацій, щоб запобігти кібератакам на пристрої та загалом зменшити ризики безпеки компанії:

1) управління поверхнею атаки, інвентаризація та моніторинг усіх пристроїв. Під час планування захисту IoT однією з головних задач має бути створення карти підключених пристроїв для їх інвентаризації. Команди безпеки повинні знати точну кількість пристроїв, що використовуються, а також ідентифікатори виробників, серійні номери, версії обладнання та прошивки. Моніторинг, аналіз та звітність в режимі реального часу є вкрай важливими для організацій, щоб мати можливість керувати ризиками Інтернету речей. Проте традиційні рішення безпеки кінцевих точок зазвичай використовують технологію так званих програмних агентів, які не підходять для пристроїв IoT. На щастя існують кращі сучасні підходи – безагентні рішення (наприклад DeviceTotal) моніторингу поверхні атаки. Вони забезпечують оцінку рівня ризику в режимі реального часу, неперервно аналізуючи поведінку і стан всіх підключених пристроїв Інтернету речей. Деякі рішення такого плану навіть дозволяють керувати поверхнею прекогнітивних атак, враховуючи ризики потенційних «атак нульового дня». Ці інструменти безпеки дають можливість організаціям використовувати всі переваги технології IoT, виправивши її основний недолік – недостатній рівень безпеки;

2) сегментація мережі. У разі успішної кібератаки зловмисник може отримати достсуп до всієї мережі організації. Сегментація запобігає цьому, обмежуючи поверхню атаки та мінімізуючи збитки. Сегментація мережі – це процес поділу внутрішньої мережі на кілька окремих підмереж. Хоча сегменти можуть час від часу спілкуватися між собою, зазвичай вони незалежні та

ізольовані один від одного. Цей метод дає змогу зосереджувати більше уваги на окремих частинах мережі, які місять найбільш критичні дані, для їх посиленого захисту;

3) встановлення надійних паролів для IoT. Багато пристроїв IoT постачаються зі слабкими попередньо встановленими паролями, які дуже легко підібрати. Як тільки IoT пристрій вперше реєструється у вашій мережі, для початку, найкращою методикою буде змінити його попередньо встановлений пароль на значно складніший. Новий пароль має бути стійким для підбору, унікальним для кожного захищеного пристрою та відповідати політикам керування паролями вашої команди з безпеки IT;

4) захист всіх пристроїв IoT на фізичному рівні. Фізичний захист пристроїв має дуже велике значення, оскільки девайси, що доступні ззовні можуть піддатися фізичному втручанню зловмисників з метою отримання несанкціонованого доступу або завантаження в систему шкідливого ПЗ. Тому слід забезпечити надійне місце дислокації пристрою, щоб до нього не було відкритого доступу;

5) своєчасні оновлення прошивок. Нові версії прошивок можуть мати виправлення наявних програмних вразливостей пристрою. Саме тому їх регулярне оновлення значно покращить загальну безпеку IoT. Проте оновлення також слід перевіряти на підробки, оскільки зловмисники можуть під виглядом оновлення завантажити на пристрій шкідливе програмне забезпечення. Інша сторона оновлень – це вразливості в офіційних оновленнях. Потрібно контролювати версії і тримати найновішу з безпечних версій прошивки, в цьому допоможуть автоматизовані системи аналізу прошивки пристроїв.

На рисунку 1.4 наведено 6 основних критеріїв, які забезпечують безпеку системи. Опис цих критеріїв:

– критерій конфіденційності – дані захищені від доступу неуповноважених осіб, забезпечуючи збереження їх секретності;

– критерій цілісності – дані залишаються надійними, тобто їх вміст не змінюється або не пошкоджується у процесі обробки, передачі чи зберігання;

– критерій доступності – дані або послуги доступні тоді, коли і де це необхідно користувачам чи системі;

– критерій безвідмовності – **с**истема забезпечує ведення надійного аудиторського шляху, який дозволяє підтверджувати виконання дій або виявляти їх відсутність;

– критерій достовірності – компоненти системи здатні підтвердити свою ідентичність, що дозволяє забезпечувати їхню автентичність і уникати шахрайства;

– критерій секретності – послуга чи система автоматично не отримує доступ до даних клієнтів без їхньої згоди.

Рисунок 1.4 – Вимоги безпеки IoT [7]

Виконання наведених вище рекомендацій допоможе вам безпечно користуватись пристроями IoT, використовуючи їх користь на повну та при цьому мінімізувати ризики, які вони можуть створювати. Але слід пам’ятати, що кібератаки постійно розвиваються та ускладнюються. Тому важливо бути в курсі нових подій в кіберпросторі та регулярно оновлювати заходи безпеки, використовуючи сучасні передові рішення для забезпечення моніторингу пристроїв та аналізу поверхні атаки.

1.3 Міжнародні стандарти безпеки для IoT

В наш цифровий час дуже швидкими темпами розвиваються різноманітні інформаційні технології, але на жаль разом з ними розвивається і кіберзлочинність.

Ключовим моментом захисту від кіберзлочинності є підготовка і виявлення вразливих місць, а також стійкість з точки зору взаємодії з загальними системами управління. Керувати інформаційною безпекою, а також визначити злочинців і притягнути їх до відповідальності допоможуть стандарти серії ISO/IEC 27000, розроблені спільним технічним комітетом Міжнародної організації з стандартизації (ISO) та Міжнародної електротехнічної комісії (IEC) – ISO/IEC JTC 1 [8].

Перший стандарт серії, ISO/IEC 27001 щодо систем управління інформаційною безпекою (ISMS), опубліковано вже більше 20 років тому. З того часу у серії видано понад 40 міжнародних стандартів, що охоплюють все: від створення спільного словника (ISO/IEC 27000), управління ризиками (ISO/IEC 27005), безпеки у хмарних технологіях (ISO/IEC 27017 і ISO/IEC 27018) до методів судової експертизи, що використовують для аналізу цифрових доказів та розслідування інцидентів ISO/ IEC 27042 та ISO/IEC 27043 відповідно).

Для промислових систем автоматизації та управління розроблено стандарт IEC/ISA 62443, який визначає процеси, техніки та вимоги до кібербезпеки в таких системах. Він охоплює аспекти створення та підтримки ефективної програми безпеки, проектування систем та вимоги до компонентів, забезпечуючи захист від потенційних загроз [9].

У сфері автомобільної промисловості діє стандарт ISO/SAE 21434, який пропонує заходи кібербезпеки для життєвого циклу розробки дорожніх транспортних засобів. Він допомагає виробникам інтегрувати заходи безпеки на всіх етапах розробки та експлуатації автомобілів, забезпечуючи захист від кіберзагроз [9].

Для споживчих IoT-пристроїв розроблено стандарт ETSI EN 303 645, який містить базові вимоги до безпеки, включаючи відсутність універсальних паролів за замовчуванням, управління вразливостями та регулярне оновлення програмного забезпечення. Впровадження цього стандарту сприяє підвищенню рівня безпеки споживчих пристроїв, підключених до інтернету [9].

Стандарт IoT Security Framework акцентує увагу на ключових аспектах, зокрема, конфіденційності, автентифікації та управлінні ризиками.

IoT Security Framework – це набір рекомендацій, розроблених організаціями, такими як IoT Security Foundation, що спрямовані на забезпечення надійної безпеки на кожному рівні IoT-екосистеми. Він включає наступні ключові аспекти:

– безпека пристроїв – це рекомендації щодо шифрування даних, автентифікації та оновлення програмного забезпечення;

– управління даними – забезпечення конфіденційності та цілісності переданої інформації;

– мережеві протоколи – використання безпечних протоколів, таких як TLS, DTLS, для шифрування мережевого трафіку;

– управління ідентифікацією та доступом – впровадження багатофакторної автентифікації та управління привілеями.

Дотримання цих міжнародних стандартів є критично важливим для забезпечення безпеки IoT-систем, оскільки вони надають структурований підхід до управління ризиками та захисту інформації в сучасному цифровому середовищі.

Проведений аналіз стандартизації в усьому світі показує, що стандартизація в галузі IoT усе ще розширюється. Фігурує велика кількість організацій, які створюють стандарти, простір застосування яких досить широкий. Очевидно, що подальший розвиток і широке застосування IoT буде потребувати більш глибшої співпраці. Оскільки інтелектуальні речі пов’язані між собою, то інтерфейси між кінцевими пристроями мають бути визначені на технічному та прикладному рівнях, щоб одержати всі переваги IoT. Однією з

найбільших проблем, від якої залежить кінцевий успіх IoT, є розроблення глобальних сумісних стандартів. Однак стандарти IoT сьогодні все ще широко відкриті – на рівні пристрою, протоколу та програмного забезпечення, оскільки не має глобальних валідованих рамок стан дартизації. А досягнення повного потенціалу IoT буде значно ускладнене за від сутності відповідних стандартів та їх невиконанням

На рисунку 1.5 зображено основні аспекти IoT Security Framework.

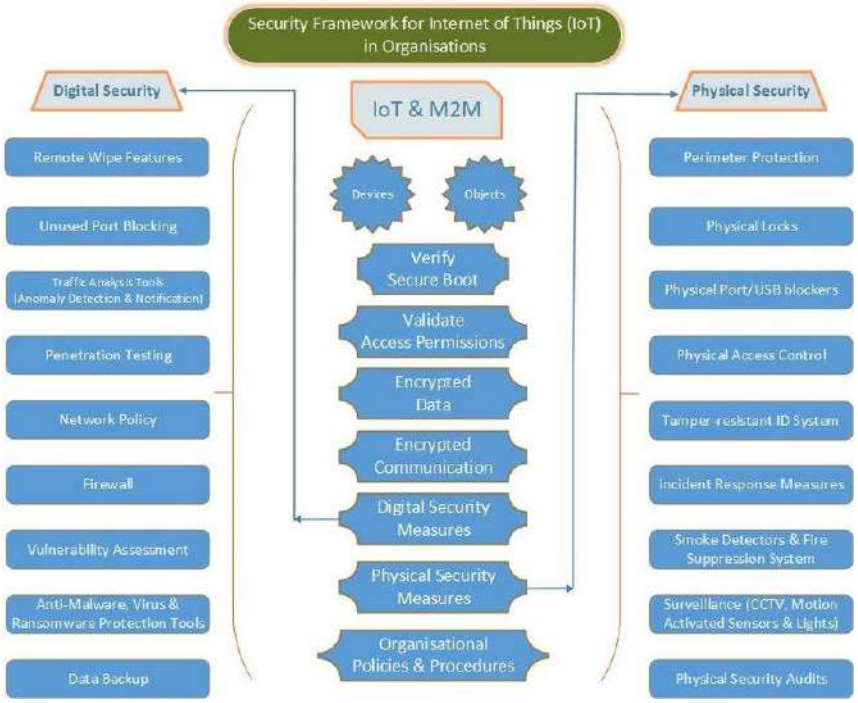

Рисунок 1.5 – Основні аспекти IoT Security Framework [10]

Робота зі стандартизації IoT прогресує у багатьох напрямах. Також варто за значити різноманітні поточні глобальні зусилля зі стандартизації і в Україні. Щоб IoT та життєво важливі мережі IoT були успішними в Україні, важливо визначити правильні варіанти використання разом із правильною політикою розгортання, зберігаючи водночас доступність технології для користувача.

РОЗДІЛ 2
АНАЛІЗ АЛГОРИТМІВ БЕЗПЕКИ IOT

2.1 Методи оцінки ефективності алгоритмів безпеки IoT

Ефективність алгоритмів безпеки в системах Інтернету речей (IoT) залежить від їх здатності протистояти загрозам, забезпечувати конфіденційність, цілісність та доступність даних, а також від їхньої продуктивності в умовах обмежених ресурсів пристроїв. Оцінка ефективності включає кілька ключових методів, що поєднують теоретичні моделі, емпіричне тестування та аналіз продуктивності. Розглянемо основні підходи до оцінки ефективності:

1) аналіз вразливостей спрямований на виявлення слабких місць у системі безпеки IoT, які можуть бути використані зловмисниками. Для цього використовуються автоматизовані інструменти, такі як Nessus (рис. 2.1), що дозволяють виявляти конфігураційні помилки та незахищені компоненти.

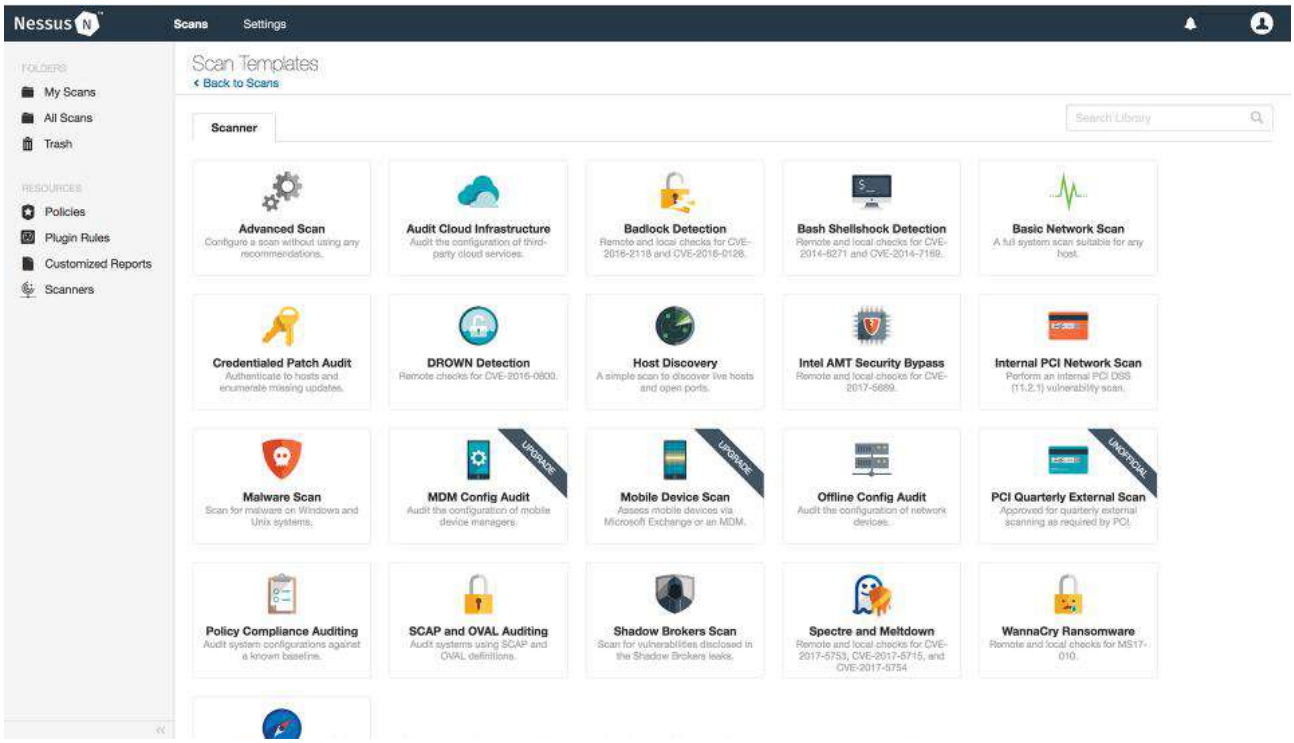

Рисунок 2.1 – Вікно програми Nessus [12]

Ризики оцінюються за допомогою метрик, наприклад, Common Vulnerability Scoring System (CVSS), яка дозволяє кількісно оцінити ризик за такими параметрами, як базовий вплив, тимчасові модифікації та фактори довкілля [11];

2) моделювання загроз є важливим етапом оцінки безпеки IoT-систем, який дозволяє передбачити потенційні загрози, ідентифікувати їх джерела та розробити ефективні стратегії захисту. Цей процес використовує аналітичні та візуальні інструменти для визначення слабких місць у системі та аналізу можливих сценаріїв атак. Основними підходами до моделювання загроз є [11]:

– дерева атак (Attack Trees) – це ієрархічна структура, яка описує цілі атакуючого та шляхи їх досягнення. Кожна гілка дерева представляє можливий метод реалізації атаки;

– графи атак (Attack Graphs) – це складніші моделі, які відображають всі можливі вразливості в IoT-системі та взаємозв’язки між ними. Вони дозволяють проводити симуляції атак для оцінки стійкості алгоритмів;

3) тестування на проникнення (пентестинг) – це практичний метод оцінки безпеки IoT-систем, що включає симуляцію реальних атак для виявлення вразливостей, перевірки стійкості до загроз та визначення ефективності алгоритмів захисту. Пентестинг є важливим етапом у забезпеченні кібербезпеки, оскільки дозволяє ідентифікувати слабкі місця до того, як вони будуть використані зловмисниками [11];

4) метрики безпеки – це кількісні або якісні показники, які використовуються для оцінки рівня захищеності IoT-систем від кіберзагроз. Вони дозволяють виміряти ефективність алгоритмів захисту, виявити слабкі місця та вдосконалити стратегії безпеки.

Для кількісної оцінки ефективності використовуються спеціалізовані метрики:

– рівень виявлення атак (Detection Rate) – частка правильно виявлених загроз;

– частота хибних спрацьовувань (False Positive Rate) – це кількість помилкових тривог;

– час реагування (Response Time) – швидкість, з якою алгоритм виявляє та нейтралізує загрозу.

2.2 Аналіз алгоритмів шифрування та протоколів IoT

Алгоритми шифрування є фундаментальними компонентами безпеки в інформаційних системах, зокрема в IoT, де вимоги до ефективності та стійкості до атак особливо високі. Є три популярні алгоритми.

Розширений стандарт шифрування (AES) був опублікований Національним інститутом стандартів і технологій (NIST) у 2001 році.

AES – це симетричний блоковий шифр, який має замінити DES як затверджений стандарт для широкого спектру програми.

Через популярність AES було зроблено ряд зусиль для покращення продуктивності за допомогою програмної та апаратної оптимізації. Зокрема, у 2008 році Intel представила нові інструкції щодо вдосконаленого стандарту шифрування (AES-NI) як апаратне розширення набору інструкцій x86 для підвищення швидкості шифрування та дешифрування. Інструкція AES-NI дозволяє процесорам x86 досягти продуктивності 0,64 циклу/байт для режиму автентифікованого шифрування відомого як AES-GCM.

У 2018 році Intel додала векторизовані інструкції, які називаються VAES*, до існуючих AES-NI для висококласних процесорів Inte17. Ці інструкції призначені для підштовхування продуктивність програмного забезпечення AES нижче, до нової теоретичної пропускної здатності 0,16 циклів/байт.

AES став найпоширенішим симетричним шифром. У порівнянні з шифрами з відкритим ключем, такі як RSA, структура AES і більшість симетричних шифрів є досить складними і не може бути описана так просто, як багато інших криптографічних алгоритмів [13].

На рисунку 2.2 показана загальна структура процесу шифрування AES.

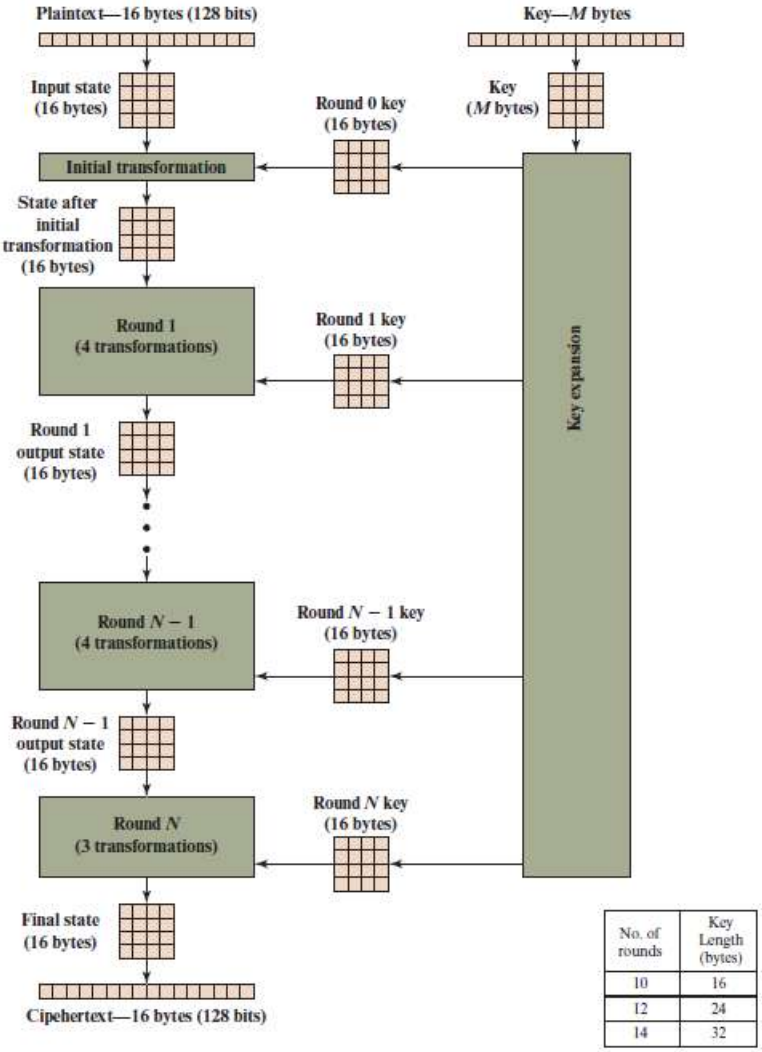

Рисунок 2.2 – Загальна структура процесу шифрування AES [13]

До переваг AES можна віднести:

– висока швидкість виконання (AES оптимізований для швидкої роботи на сучасному апаратному забезпеченні);

– стійкість до атак (захищений від відомих типів криптоаналітичних атак, таких як диференціальна криптоаналіз та лінійна криптоаналіз);

– гнучкість (підтримує три різні довжини ключів, що дозволяє балансувати між швидкістю та рівнем безпеки);

– енергоефективність (відносно низьке споживання енергії, що робить його ідеальним для використання у пристроях з обмеженими ресурсами, таких як IoT).

До недоліків AES належать:

– проблеми з управлінням ключами (AES є симетричним алгоритмом, тому потребує безпечного розподілу ключів між сторонами, що є складним у великих або розподілених системах);

– вразливість до атак на каналі передачі даних (алгоритм не забезпечує захист від атак на каналі, таких як маніпуляція або повторення повідомлень);

– ресурсоємність при високих вимогах (при роботі з довгими ключами (256 біт) збільшується навантаження на обчислювальні ресурси).

Алгоритм AES є ідеальним вибором для більшості додатків, включаючи IoT, завдяки його високій швидкості, безпеці та ефективності. Однак для успішного впровадження AES важливо забезпечити надійне управління ключами та врахувати обмеження пристроїв, на яких він використовується.

Криптоалгоритм RSA запропонували в 1978 р. три автори: Р. Райвест (Rivest), А. Шамір (Shamir) і А. Адлеман (Adleman). Він став першим алгоритмом з відкритим ключем, який може працювати як в режимі шифрування даних, так і в режимі електронного цифрового підпису. Безпека алгоритму RSA побудована на принципі складності факторизації. Алгоритм використовує два ключі – відкритий і секретний, разом вони утворюють пари ключів Відкритий ключ використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним йому секретним ключем. Криптоалгоритм RSA визнаний стійким при достатній довжині ключів. Сьогодні довжина ключа – 1024 біта вважається прийнятним варіантом.

В асиметричній криптосистемі RSA кількість використовуваних ключів пов’язана з кількістю абонентів лінійною залежністю (у системі з N користувачів використовуються 2N ключів), а не квадратичною, як в симетричних системах. Слід зазначити, що швидкодія RSA істотно нижча швидкодії DES, а програмна і апаратна реалізація криптоалгоритму RSA набагато складніша, ніж DES. Тому криптосистема RSA, як правило, використовується при передачі невеликого об’єму повідомлень та ЕЦП [14].

На рисунку 2.3 зображено схему роботи алгоритму RSA.

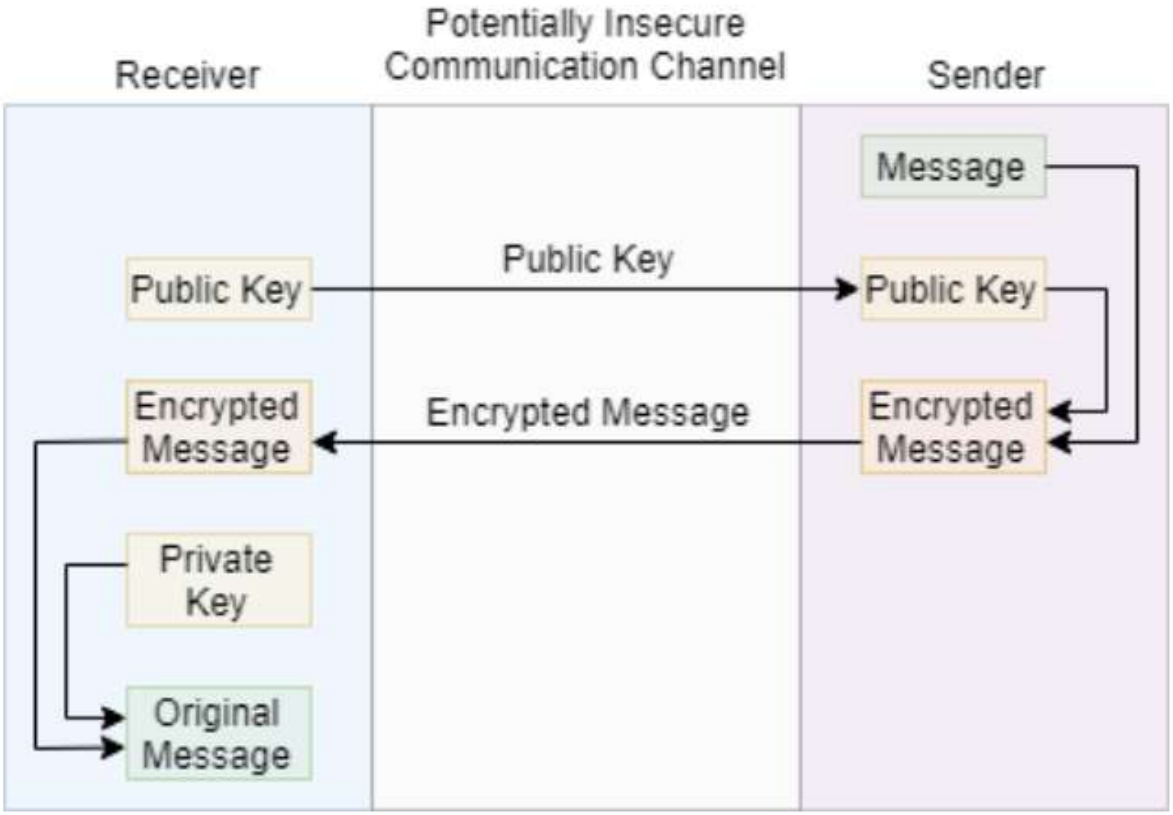

Рисунок 2.3 – Схема роботи алгоритму RSA [14]

Алгоритм RSA має кілька важливих переваг, які роблять його популярним вибором у багатьох сферах, особливо у випадках, коли потрібна безпека обміну даними. Його основною перевагою є можливість забезпечення шифрування та цифрового підпису без необхідності попереднього обміну секретними ключами між сторонами, що спрощує інтеграцію в розподілених системах. RSA базується на асиметричній криптографії, де використовується пара ключів (відкритий і закритий), що дозволяє підвищити рівень захисту. Алгоритм має високу стійкість до атак при достатній довжині ключа (2048 біт і більше), завдяки чому широко застосовується у сертифікатах безпеки, VPN та електронній комерції.

Однак RSA має й недоліки. Одним із найбільших є його низька швидкість у порівнянні із симетричними алгоритмами, такими як AES, що робить RSA менш придатним для шифрування великих обсягів даних. Натомість його часто використовують для шифрування невеликих повідомлень або передачі ключів симетричних алгоритмів. Іншим обмеженням є високі вимоги до

обчислювальних ресурсів, що може бути проблемою для пристроїв із низькою продуктивністю, таких як IoT-пристрої. Крім того, ефективність RSA значно залежить від правильної реалізації; будь-які помилки можуть зробити систему вразливою до атак, таких як атаки на підпис або математичні уразливості.

Алгоритм RSA є одним із найпоширеніших рішень для забезпечення безпеки в цифрових системах завдяки його стійкості до криптографічних атак і можливості застосування для шифрування, аутентифікації та цифрових підписів. Його асиметрична природа спрощує обмін ключами між учасниками, що є критично важливим у розподілених системах і мережах. Однак низька швидкість шифрування великих обсягів даних та значні вимоги до обчислювальних ресурсів обмежують застосування RSA у високонавантажених або ресурсом обмежених середовищах, таких як IoT.

Таким чином, RSA найкраще підходить для завдань, що потребують захисту невеликих даних, таких як передача симетричних ключів або створення цифрових підписів, тоді як для масштабних задач ефективніше використовувати симетричні алгоритми, наприклад AES. Це підкреслює важливість вибору криптографічного методу, враховуючи вимоги до продуктивності, безпеки та обчислювальних ресурсів.

Алгоритм ECC (Elliptic Curve Cryptography) в основному залежить від алгебраїчної структури еліптичних кривих. ECC включає в себе три етапи експлуатації, тобто ключове узгодження, шифрування та алгоритми цифрового підпису. Першим кроком є ключовий алгоритм розподілу, який використовується для спільного використання секретного ключа, другий крок – це алгоритм шифрування, який забезпечує конфіденційне спілкування, а останній – це алгоритм цифрового підпису, який використовується для автентифікації підписувача, тобто відправника, та перевірки цілісності повідомлення.

У процесі шифрування ECC еліптична крива представляє набір точок, які задовільняють математичне рівняння ($y2 = x3 + ax + b$) (рис. 2.4).

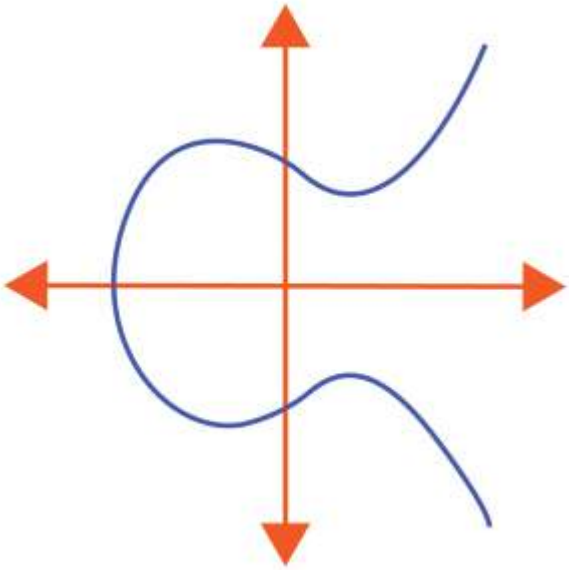

Рисунок 2.4 – Еліптична крива шифрування ECC [15]

ECC вважається кращим для створення більш швидких, і менших ефективних ключів. ECC пропонує еквівалентну кількість безпеки для набагато менших розмірів ключа, а отже, зменшує витрати на обробку та комунікацію. ECC вважається найбільш підходящим для сенсорних мереж, що забезпечує гарний компроміс між ключовими розмірами та безпекою [15].

ECC (Elliptic Curve Cryptography) має значні переваги, які роблять його популярним вибором для сучасних криптографічних систем, особливо в середовищах із обмеженими ресурсами, таких як IoT. Однією з ключових переваг є ефективність: ECC забезпечує такий самий рівень безпеки, як RSA, але при значно меншому розмірі ключа. Наприклад, ключ ECC розміром 256 біт еквівалентний RSA із ключем довжиною 3072 біт. Це зменшує потребу в обчислювальних ресурсах, знижує споживання енергії та пришвидшує операції, що є критичним для мобільних пристроїв, сенсорів та інших IoT-систем.

Ще однією перевагою ECC є висока стійкість до сучасних атак, включаючи атаки на квантових комп’ютерах, хоча ця перевага зберігається лише для специфічно обраних кривих. ECC використовується для шифрування, генерації цифрових підписів і обміну ключами, що робить його універсальним інструментом у сфері криптографії.

Однак ECC має і свої недоліки. Основним є складність його реалізації. Алгоритм вимагає ретельного підбору параметрів кривих, адже неправильний вибір може знизити безпеку системи. Реалізація ECC також потребує більш глибоких знань і спеціалізованих навичок порівняно з RSA чи AES. Крім того, хоча ECC ефективний для сучасних пристроїв, вразливості у фізичному доступі до обладнання або каналах передачі даних (наприклад, атаки на побічні канали) можуть бути проблемою, якщо алгоритм реалізований неналежним чином.

Таким чином, ECC є ідеальним вибором для ресурсом обмежених пристроїв і застосувань, де потрібна висока ефективність і безпека. Однак складність реалізації вимагає додаткової уваги до деталей, щоб уникнути потенційних вразливостей.

Таблиця 2.1 містить дані, які дозволяють провести порівняльний аналіз розглянутих алгоритмів.

Таблиця 2.1 – Порівняльна таблиця алгоритмів

Параметр	AES (Advanced Encryption Standard)	RSA (Rivest–Shamir–Adleman)	ECC (Elliptic Curve Cryptography)
Тип алгоритму	симетричний блочний	асиметричний	асиметричний
Призначення	шифрування даних	шифрування, цифрові підписи	шифрування, обмін ключами, цифрові підписи
Довжина ключа	128, 192, 256 біт	2048, 3072, 4096 біт	160-512 біт (256 біт еквівалентно 3072 біт RSA)
Швидкість	висока	низька	висока
Ресурсомісткість	низька	висока	низька
Стійкість до атак	висока (захищений від атак грубої сили)	висока (залежить від довжини ключа)	висока (ефективний для IoT)
Сфера застосування	захист даних, IoT, VPN, хмарні сервіси	передача ключів, цифрові сертифікати, електронна комерція	IoT, мобільні пристрої, блокчейн, цифрові підписи

Протоколи для Інтернету речей (IoT) забезпечують ефективну та безпечну передачу даних у мережах, що складаються з великої кількості пристроїв із

обмеженими ресурсами. Розглянемо два популярні протоколи, які активно використовуються в IoT-системах.

DTLS (Datagram Transport Layer Security) є модифікованою версією протоколу TLS (Transport Layer Security), яка адаптована для роботи з ненадійним транспортним протоколом UDP. Він забезпечує ті самі функції безпеки, що й TLS, зокрема шифрування, аутентифікацію та перевірку цілісності даних, але враховує специфіку передачі у ненадійних мережах [16].

На рисунку 2.5 представлена схема функціонування протоколу DTLS, яка демонструє ключові етапи, включаючи шифрування, автентифікацію, повторну передачу пакетів та перевірку цілісності.

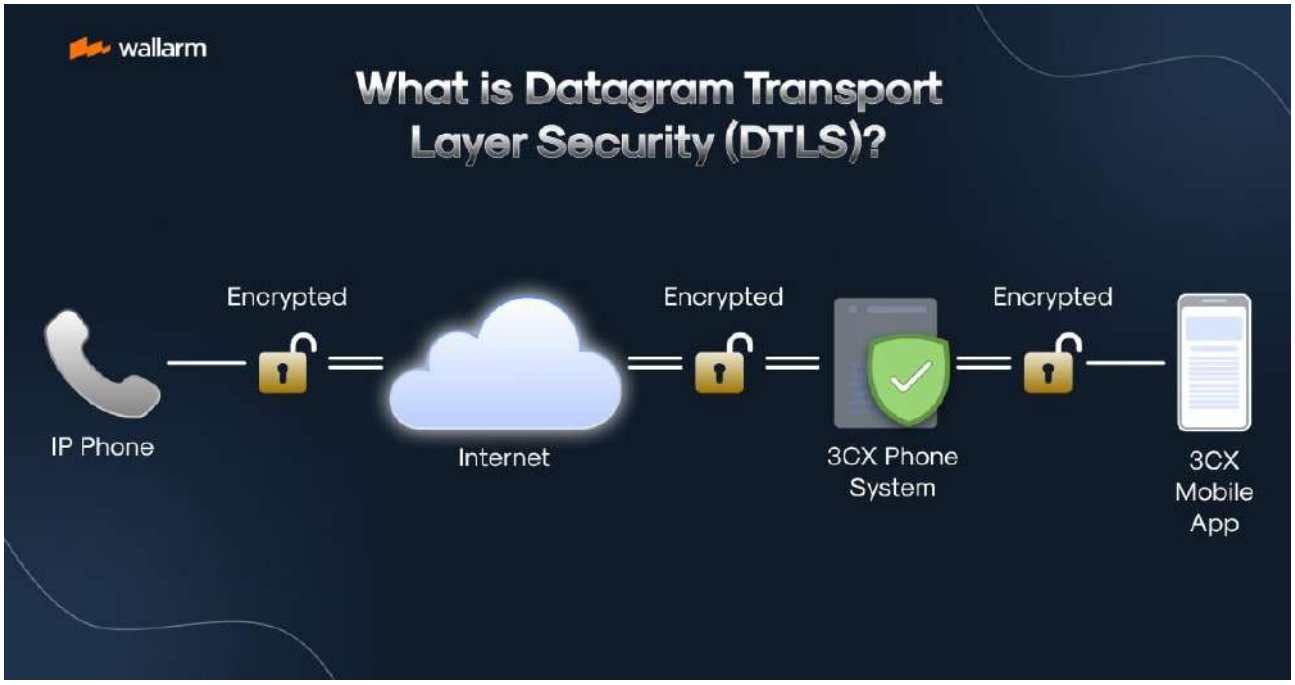

Рисунок 2.5 – Схема функціонування протоколу DTLS [17]

Протокол DTLS має ряд переваг, які роблять його ефективним вибором для безпеки в ненадійних мережах. Основною перевагою є здатність забезпечувати захист даних за допомогою шифрування, автентифікації та перевірки цілісності навіть при передачі через UDP, який не гарантує надійності доставки. Це робить DTLS ідеальним для застосувань, де важливі низька затримка і швидка обробка даних, наприклад, для відеоконференцій або пристроїв IoT. Протокол також

підтримує повторну передачу пакетів, що дозволяє зберігати цілісність даних навіть у разі втрат. Оскільки DTLS є адаптацією TLS, його легко інтегрувати в системи, які вже використовують TLS, що спрощує перехід на більш захищені методи передачі даних.

Проте DTLS має свої недоліки. Його налаштування є складнішим у порівнянні з більш простими протоколами, що може створювати труднощі для розробників, особливо в системах із великою кількістю пристроїв. Крім того, використання DTLS може вимагати більше ресурсів на обчислення, що може бути проблемою для пристроїв із дуже обмеженими можливостями. Хоча DTLS знижує затримки порівняно з TCP, у деяких випадках це може вплинути на продуктивність, особливо якщо мережа має високий рівень втрат пакетів. Таким чином, протокол потребує ретельного налаштування для досягнення оптимальної роботи.

MQTT (Message Queuing Telemetry Transport) – це протокол обміну повідомленнями, призначений для ефективного зв’язку між віддаленими пристроями з обмеженою пропускною здатністю мережі та малим кодом. Він використовує транспортну модель обміну повідомленнями публікації та підписки та широко використовується в таких галузях, як автомобілебудування, виробництво, телекомунікації, нафтогаз і багато інших.

MQTT – це заснований на стандартах протокол, який забезпечує між машинний зв’язок і зазвичай використовується в екосистемі Інтернету речей для передачі даних між пристроями Інтернету речей з низьким енергоспоживанням і обмеженими ресурсами та хмарою. Зокрема, MQTT використовує невеликий код і добре підходить для пристроїв, які мають обмежені можливості обробки та/або доступну пам’ять [18].

MQTT (рис. 2.6) є оптимальним вибором для сенсорних мереж, розумних будинків, систем моніторингу стану навколишнього середовища, промислових IoT-рішень та будь-яких додатків, де пристрої мають обмежені енергетичні ресурси і потребують ефективної передачі даних.

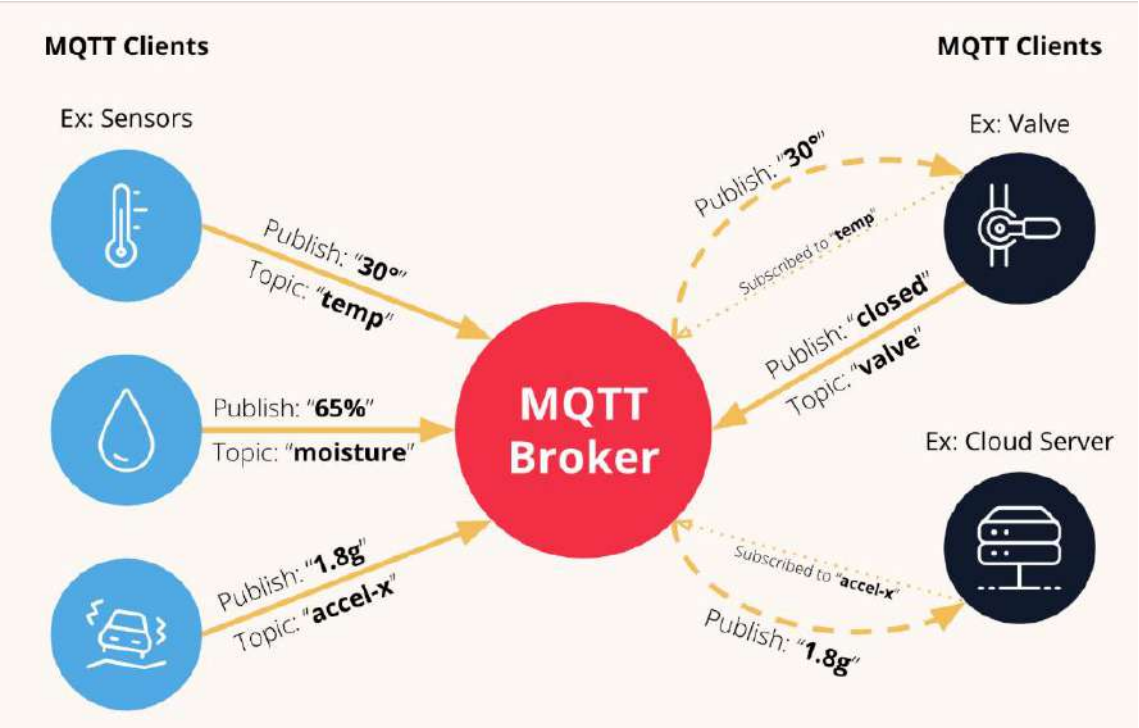

Рисунок 2.6 – Схема роботи MQTT [18]

Однією з головних переваг MQTT є його енергоефективність, що робить його придатним для пристроїв із обмеженим енергоспоживанням. Протокол забезпечує ефективний обмін повідомленнями навіть у мережах із низькою пропускною здатністю. Використання числових ідентифікаторів замість текстових топіків додатково знижує обсяг трафіку, що критично для сенсорних мереж. MQTT підтримує роботу з пристроями в автономному режимі, дозволяючи створювати децентралізовані системи. Простота інтеграції протоколу в існуючі IoT-інфраструктури робить його зручним для розробників.

Незважаючи на переваги, MQTT має і свої обмеження. Залежність від центрального брокера створює потенційний ризик єдиної точки відмови, що може впливати на надійність системи. Крім того, MQTT не включає вбудовані механізми шифрування або автентифікації, через що для забезпечення безпеки необхідно використовувати додаткові засоби, такі як DTLS або VPN. Ця залежність від зовнішніх рішень може ускладнити реалізацію безпечної системи. Також протокол менш придатний для великих і високонавантажених систем через обмеження його архітектури.

РОЗДІЛ 3
АПАРАТНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ БЕЗПЕКИ НА БАЗІ МІКРОКОНТРОЛЕРІВ

3.1 Вибір апаратних компонентів

ESP32 – це Wi-Fi мікроконтролер (рис. 3.1) китайського виробника Espressif з двоядерним 32-розрядним процесором Tensilica Xtensa LX6 та 520 Кб пам’яті SRAM. Контролер має тактову частоту до 240 МГц залежно від режиму споживання енергії.

Мікроконтролери цієї серії широко використовуються в проєктах розумного будинку, промислової автоматики, побутовій електроніці, робототехніці, камерах для потокового відео, розпізнаванні мови й зображень та ін.

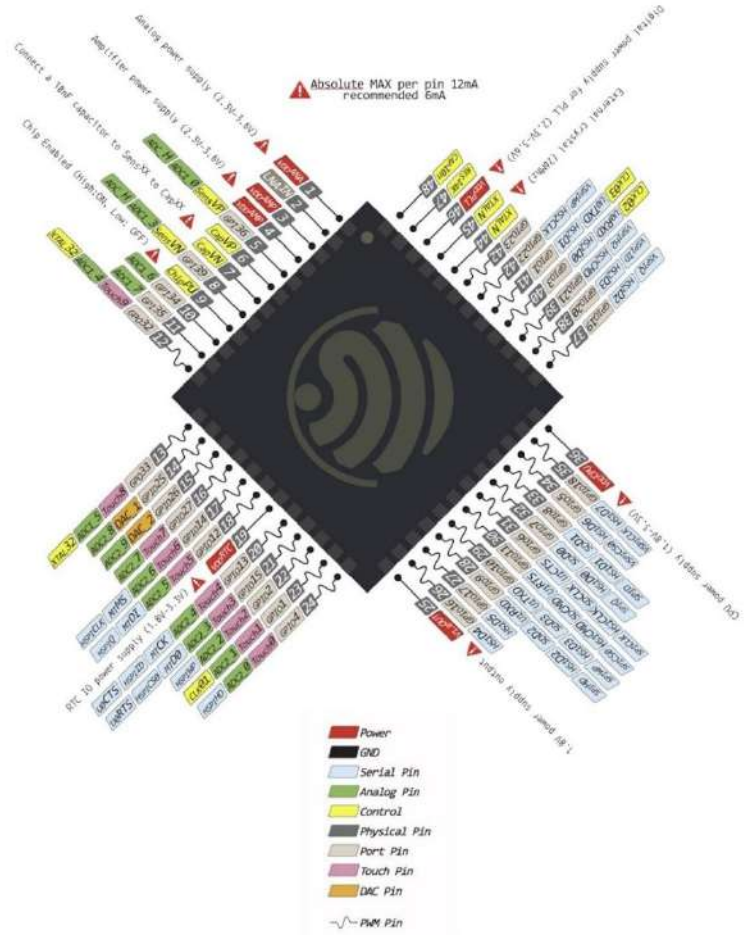

Рисунок 3.1 – Розпіновка ESP32 [19]

Arduino Uno – це пристрій на основі мікроконтролера ATmega328 (datasheet) До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШИМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньо-схемного програмування (ICSP) та кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою кабелю USB (рис. 3.2).

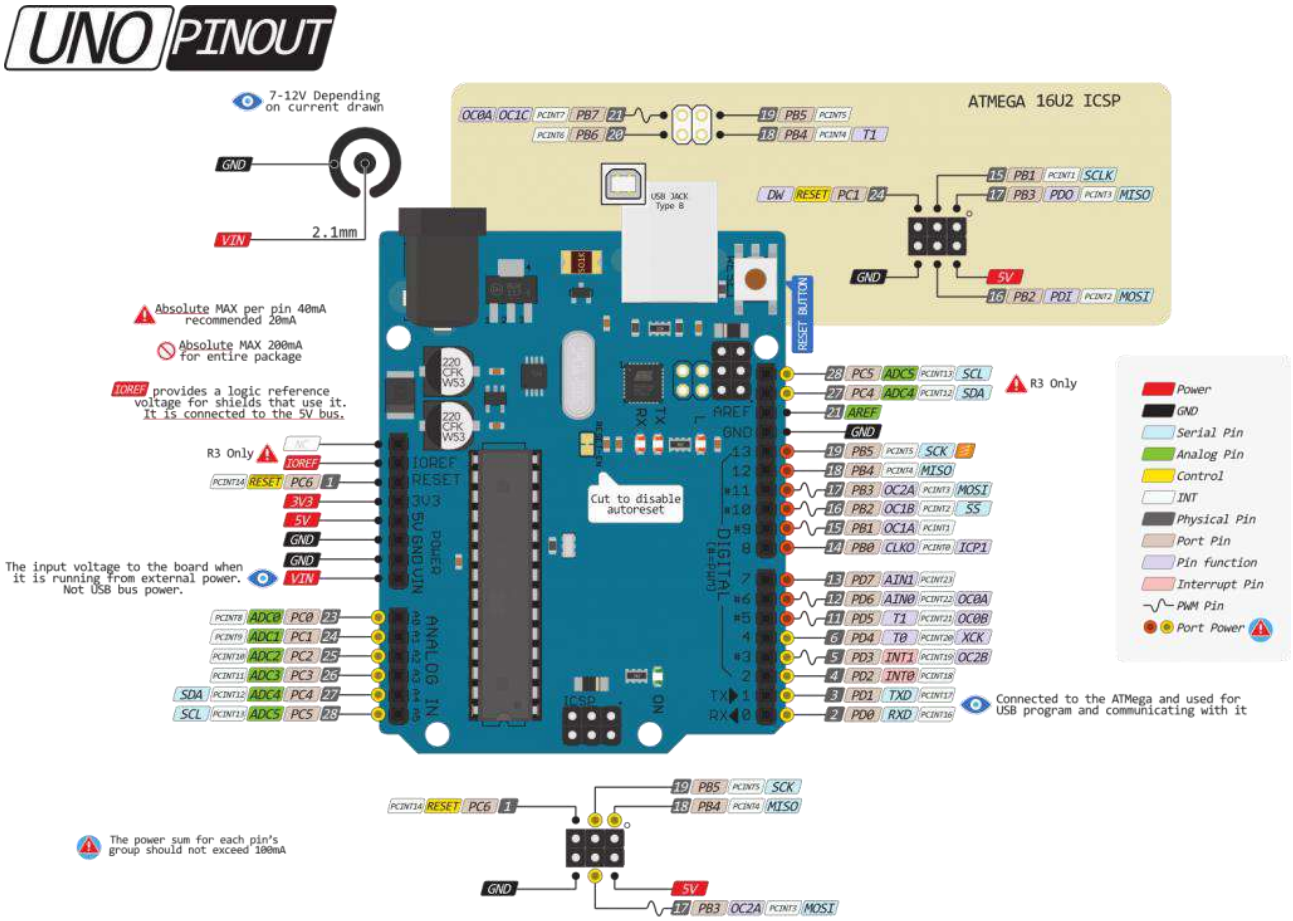

Рисунок 3.2 – Розпіновка Arduino Uno [20]

2,4-дюймовий TFT LCD з ILI9341 – це кольоровий рідкокристалічний дисплей, який часто використовується в проектах на базі Arduino, STM32, Raspberry Pi та інших мікроконтролерах. Він підтримує роздільну здатність

240x320 пікселів і забезпечує чудову якість зображення з широким кутом огляду (рис. 3.3).

Рисунок 3.3 – 2,4-дюймовий TFT LCD [21]

RFID-модуль 13.56 МГц з SPI-інтерфейсом (рис. 3.4). Даний модуль може бути використаний для різних радіоаматорських і комерційних застосувань, в тому числі для контролю доступу, автоматичної ідентифікації, робототехніки, відстеження речей, платіжних систем і т.д.

Рисунок 3.4 – RFID-модуль [22]

Енкодер EC11 – це електромеханічний пристрій для отримання інформації про напрямок обертання осі різних пристроїв, кута повороту та швидкість обертання. Вихід – стандартний двох імпульсний з відставанням одного із сигналів в залежності від напрямку обертання. Також в енкодер вбудована тактильна кнопка, що реагує на натискання вздовж осі (рис. 3.5).

Рисунок 3.5 – Енкодер EC11 [23]

Резистор на 580 Ом – це пасивний електронний компонент, що обмежує потік електричного струму в колі. Номінал 580 Ом означає, що резистор має опір 580 Ом із певним допуском, залежно від його класу точності.

Конденсатори 22 нФ (нанофарад) – це пасивні електронні компоненти, які зберігають електричний заряд і використовуються для фільтрації, розв’язки, згладжування сигналів або створення осциляторів.

3.2 Алгоритми шифрування

Організація алгоритмів шифрування дозволяє нам легко побудувати комбінований алгоритм шифрування, який є принаймні таким же сильним, як і найсильніший у каскаді, має довший ключ, може бути більш стійким до деяких атак і створювати зашифрований текст із вищою ентропією. У будь-якому випадку, не завадить мати додатковий рівень безпеки (або декілька).

На рисунку 3.6 зображено блок-схему алгоритму симетричного шифрування.

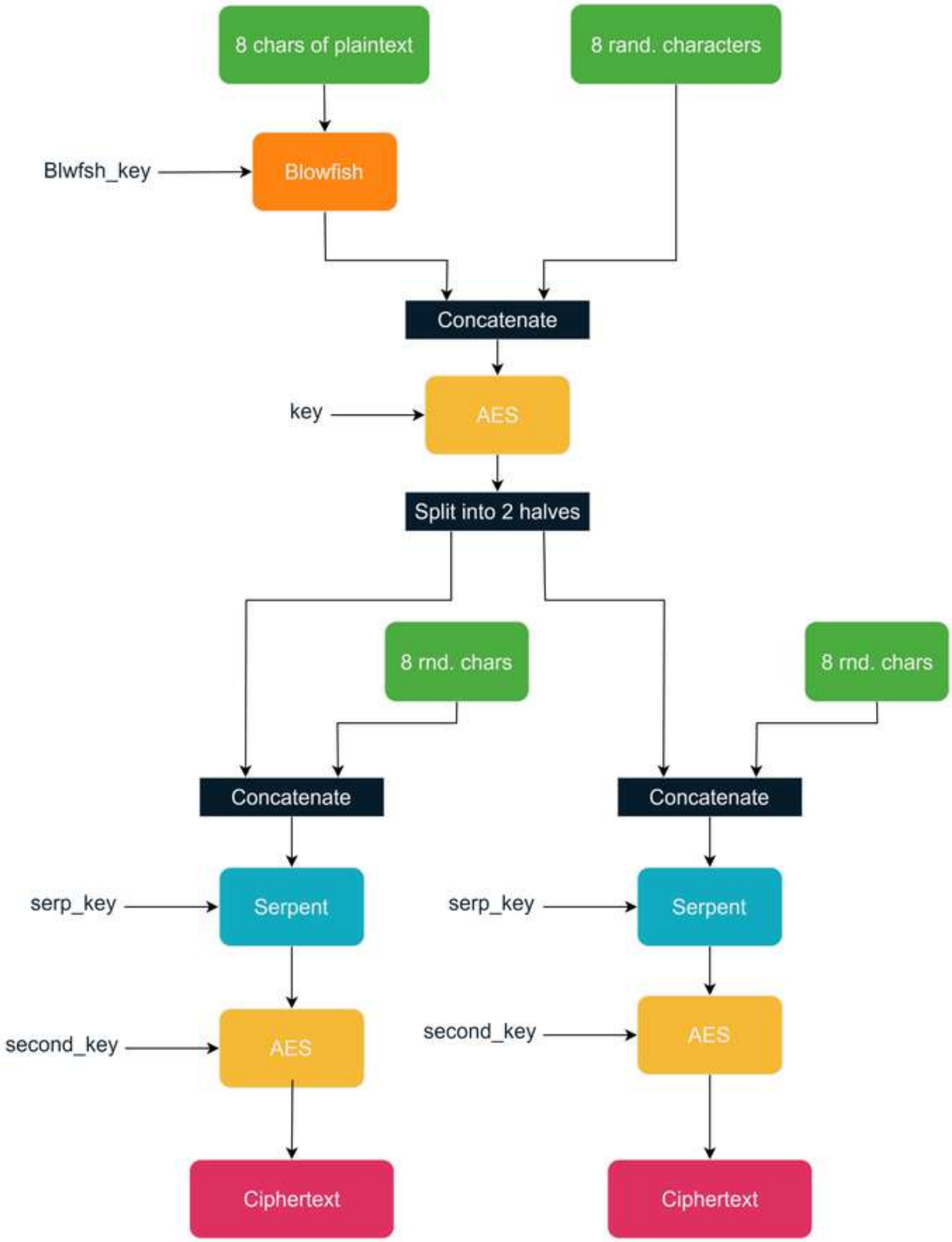

Рисунок 3.6 – Блок-схема алгоритму симетричного шифрування

3.3 Вибір програмного забезпечення

ThingSpeak – це програмне забезпечення з відкритим вихідним кодом, написане мовою Ruby, яке дозволяє користувачам спілкуватися з пристроями з підтримкою інтернету. Він полегшує доступ до даних, пошук і реєстрацію даних, надаючи API як для пристроїв, так і для веб-сайтів соціальних мереж (рис. 3.7).

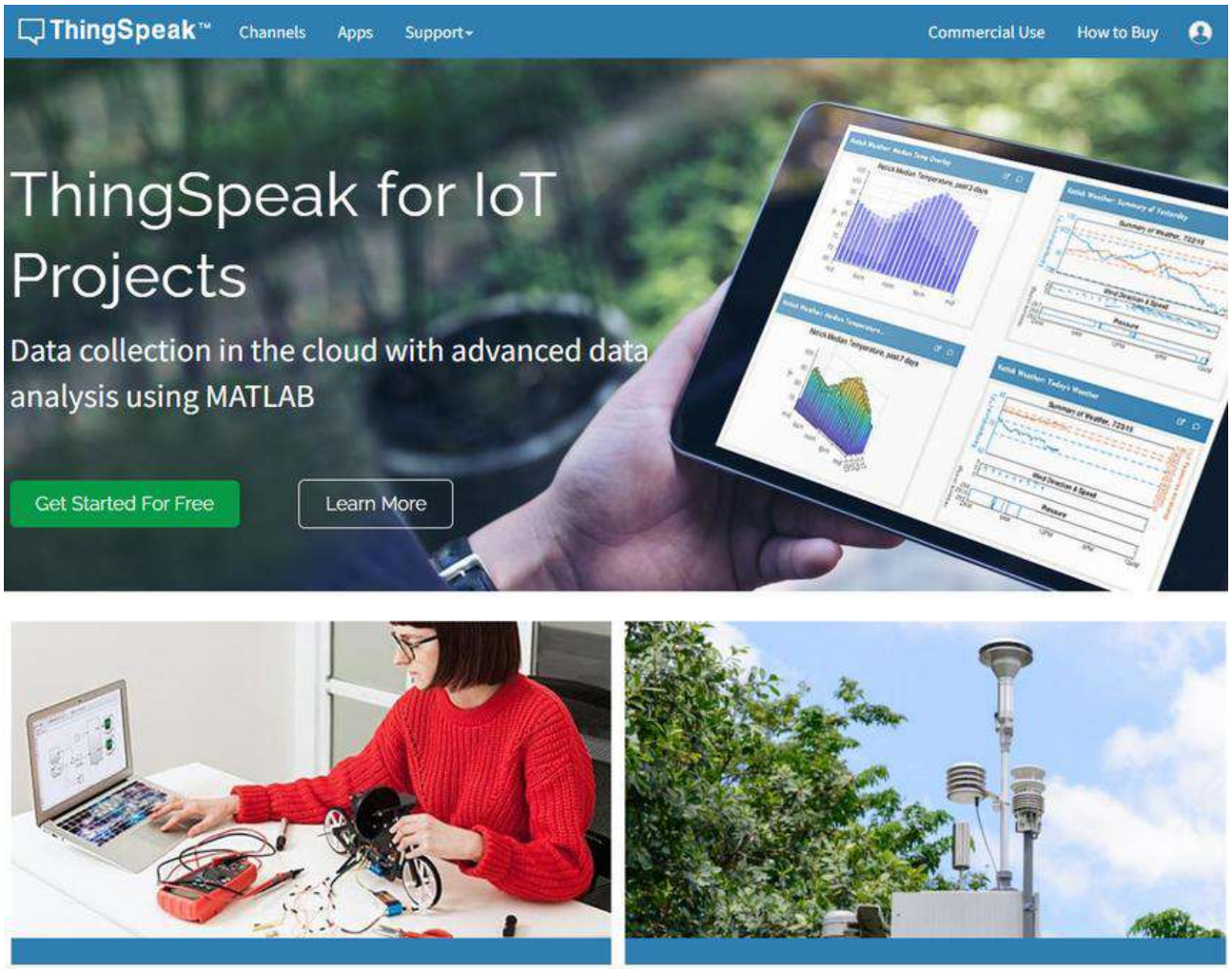

Рисунок 3.7 – ПЗ ThingSpeak

Щоб зберігати нотатки в хмарі необхідно створити обліковий запис ThingSpeak. Безкоштовний обліковий запис ThingSpeak дозволяє надсилати дані до нього 3 000 000 разів на рік.

Щоб створити безкоштовний обліковий запис ThingSpeak:

1) перейдемо на сайт thingspeak.com і натисніть кнопку «Почати безкоштовно»;

2) на наступній сторінці натиснемо на гіперпосилання «Створити»;

3) у формі, яка повинна з’явитися після цього. Введіть адресу електронної пошти, ім’я та прізвище. І натисніть кнопку «Продовжити»;

4) відкриємо свою електронну пошту та знайдіть лист від «service@account.mathworks.com»;

5) відкриємо цей лист і натисніть на ньому кнопку «Підтвердити електронну адресу»;

6) тепер повернемося до вкладки і натисніть кнопку «Продовжити»;

7) встановимо пароль для свого облікового запису та натисніть кнопку «Продовжити»;

8) натиснемо кнопку «ОК».

Після реєстрації необхідно створити новий канал (рис. 3.8):

1) виберемо «Канали» – «Мій канал»;
2) на сторінці, що відкриється, натиснемо кнопку «Новий канал»;
3) необхідно поставити прапорець праворуч від «Поле 2»;
4) назвемо канал і поля будь як;
5) перейдемо вниз і натисніть кнопку «Зберегти канал».

Рисунок 3.8 – Створення нового каналу

Для роботи з каналом потрібен його ID, Write API Key і Read API Key.

Щоб отримати ідентифікатор каналу, написати ключ API та прочитати ключ API, відкрийте канал і перейдіть на вкладку «Ключі API» (рис. 3.9).

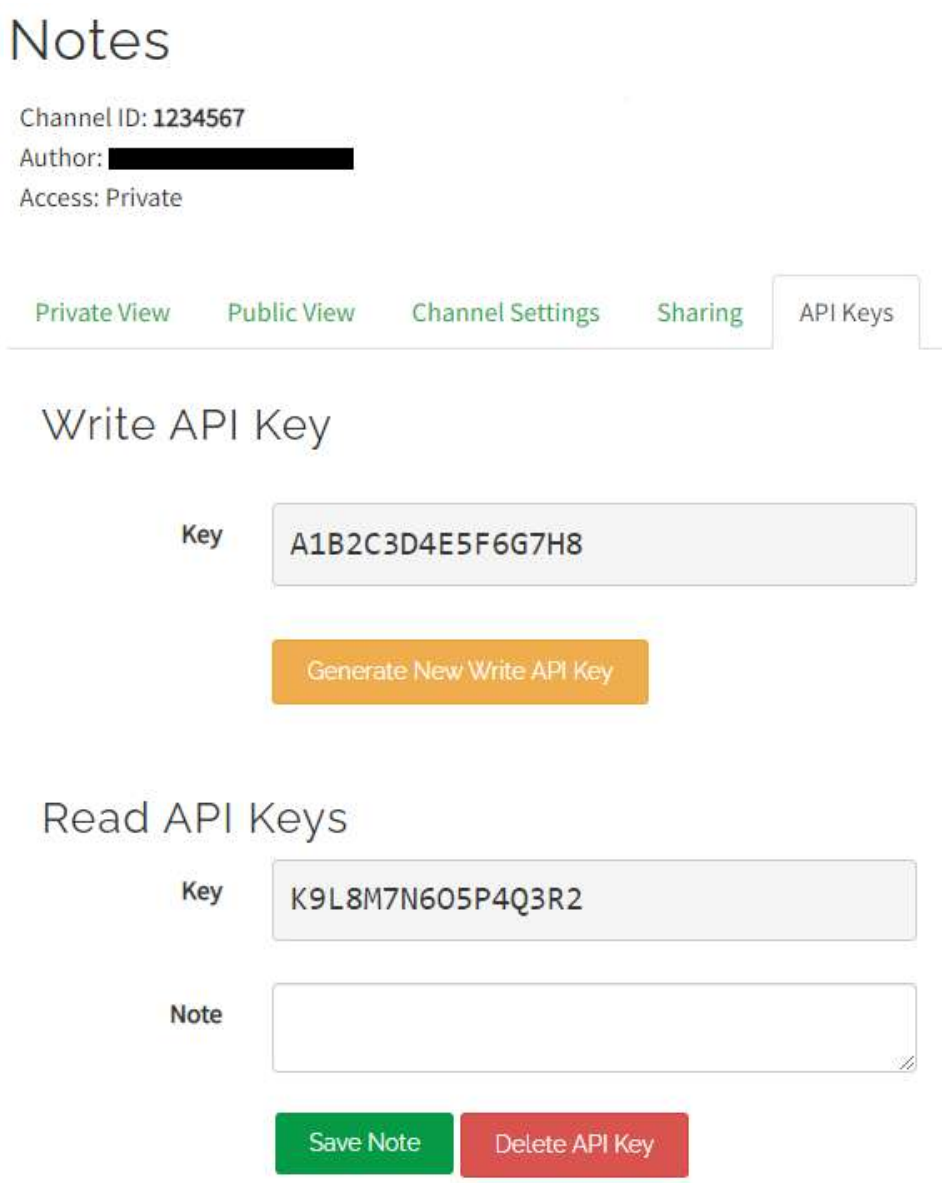

Рисунок 3.9 – Ідентифікатор каналу та ключі API

3.4 Завантаження програмного забезпечення для мікроконтролерів Arduino та ESP32

Відкриємо файл «Firmware_for_ESP32.ino», а потім замінимо значення ssid, password, myChannelNumber, myWriteAPIKey, myReadAPIKey, hmackey, Blwfsh_key, key, serp_key, second_key і TDESkey на свої.

Завантажуємо програму з папки на ESP32 (рис. 3.10).

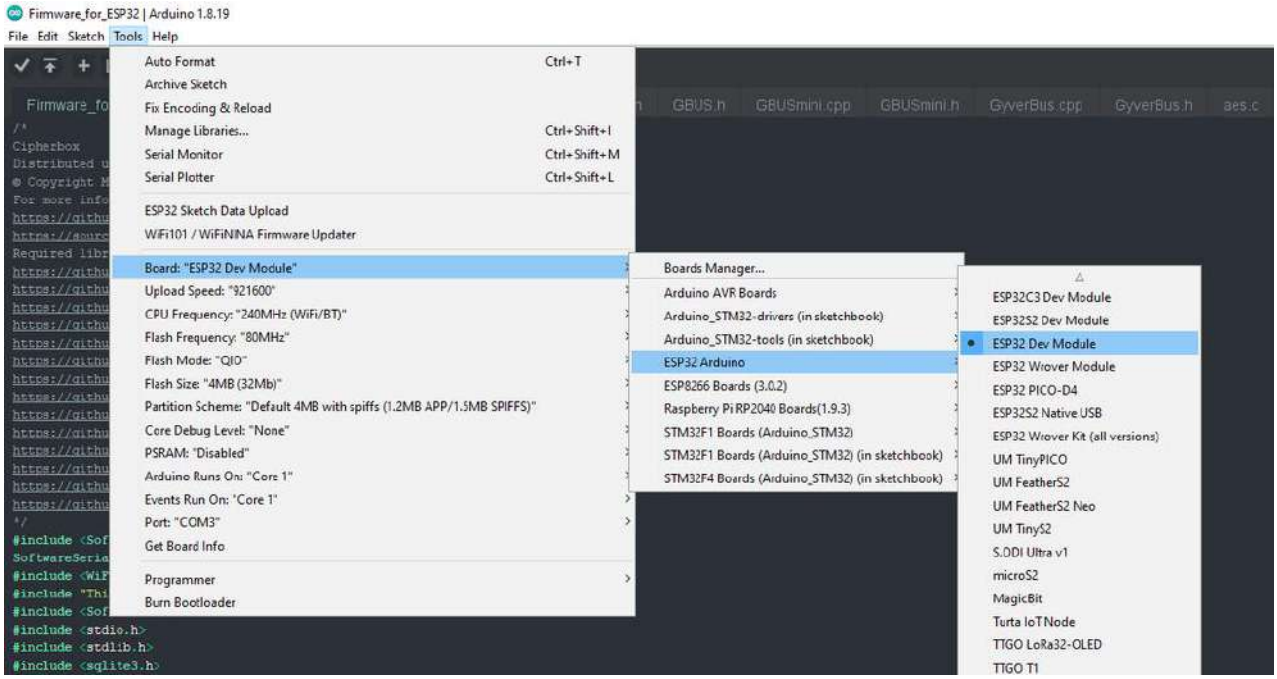

Рисунок 3.10 – Процес завантаження програми на ESP32

Завантажуємо програму з папки на Arduino Uno (рис. 3.11).

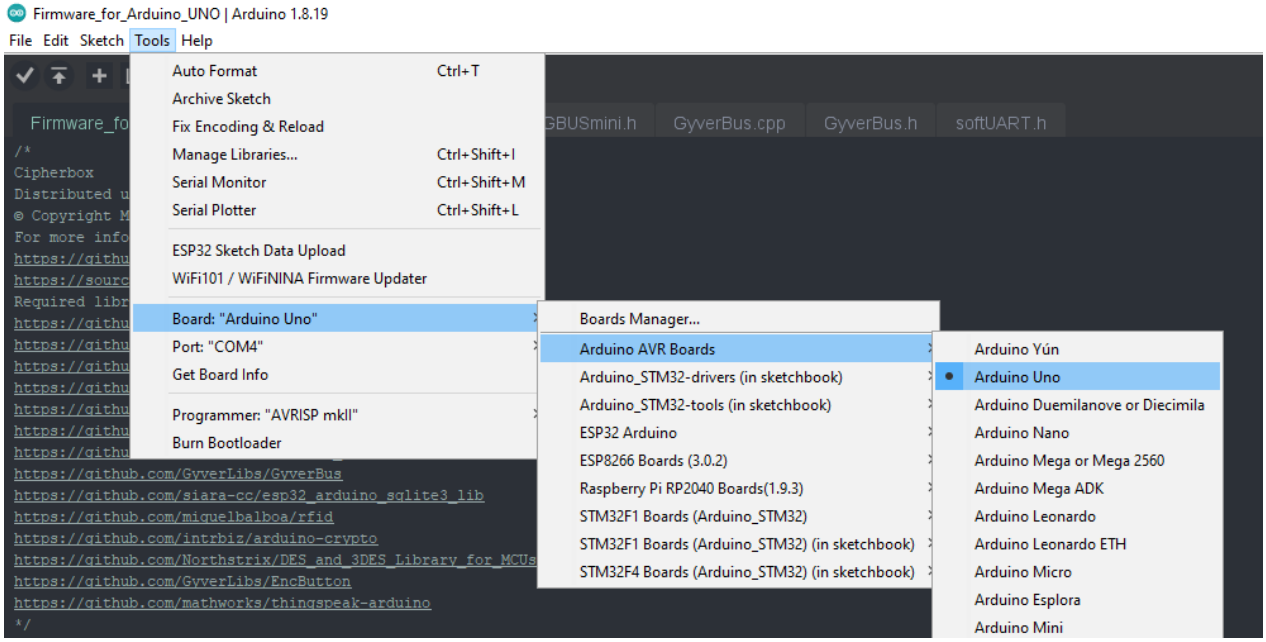

Рисунок 3.11 – Процес завантаження програми на Arduino Uno

3.5 Розробка та тестування пристрою

Необхідно скласти пристрій, який реалізує алгоритми безпеки для IoT, за принциповою схемою (рис. 3.12).

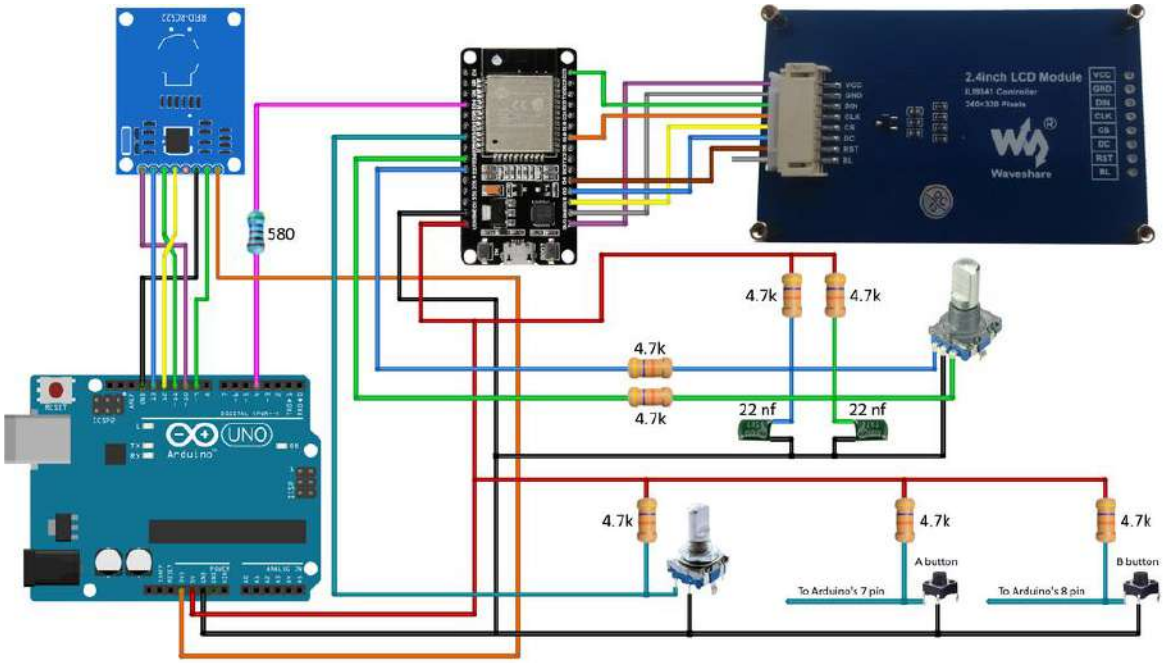

Рисунок 3.12 – Принципова схема пристрою

В результаті ми отримаємо наш пристрій (рис. 3.13).

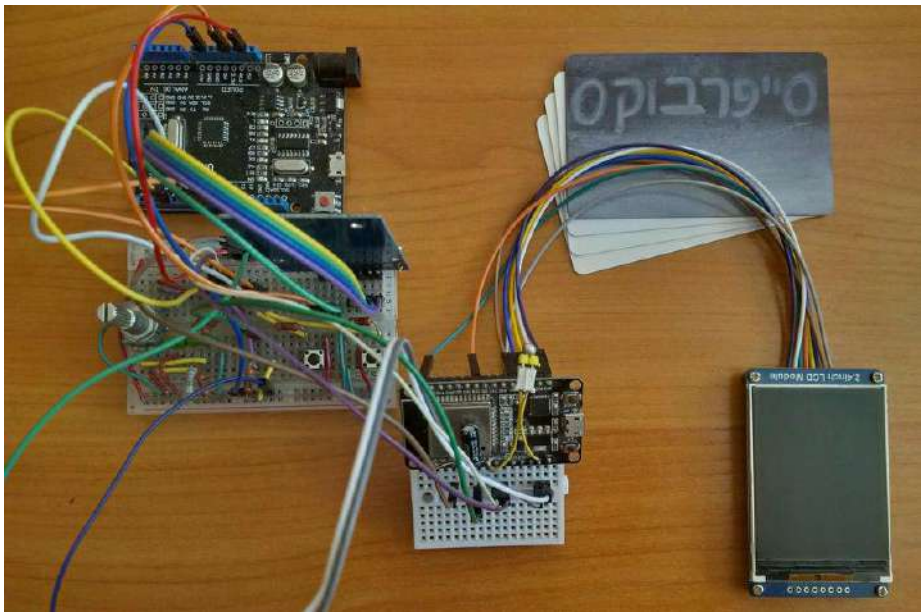

Рисунок 3.13 – Готовий пристрій

Увімкнемо живлення пристрою та встановимо чотири карти RFID (рис. 3.14).

Рисунок 3.14 – Початкове вікно

Під час першого введення нового імені користувача ви автоматично реєструєтеся і входите в систему. Усі наступні рази здійснюється лише вхід. Ви можете обрати будь-яке ім’я користувача та пароль, але важливо зберігати їх у таємниці, оскільки ім’я користувача визначає базу даних для зберігання ваших нотаток, а пароль використовується для отримання ключа доступу.

Якщо хтось дізнається ваше ім’я користувача, він може видалити всі ваші дані, зокрема логіни, кредитні картки та нотатки. У разі отримання доступу до RFID-карт, які використовуються для розблокування пристрою, та знання імені користувача і пароля, зловмисник зможе розшифрувати всі ваші дані.

Якщо вам не потрібні функції зберігання і ви плануєте використовувати лише шифрування, ви можете увійти під будь-яким іменем користувача або залишити це поле порожнім. Головне – завжди використовувати один і той самий пароль, інакше розшифрувати ваші зашифровані тексти буде неможливо.

Для переходу між полями натискайте кнопку енкодера – активне поле позначається написом унизу. Після входу в систему і відкриття головного меню:

– повернемо регулятор вправо, щоб переміщатися вниз по меню;

– повернемо регулятор вліво, щоб переміщатися вгору;

– натиснемо кнопку «A», щоб відкрити вибраний пункт меню;

– щоб повернутися з підменю до головного меню, натиснемо кнопку «B» (рис. 3.15).

Рисунок 3.15 – Головне меню

Для того щоб додати новий логін необхідно виконати наступне (рис. 3.16):

1) виберемо в головному меню рядок «Логіни»;

2) натиснемо кнопку «A»;

3) виберемо рядок «Додати логін»;

4) натиснемо кнопку «А»;

5) введемо назву;

6) натиснемо кнопку кодера чотири рази;

7) введемо ім’я користувача;

8) натиснемо кнопку кодера чотири рази;

9) введемо пароль;

10) натиснемо кнопку кодера чотири рази;

11) увійдемо в обліковий запис;

12) натиснемо кнопку кодера чотири рази;

13) натиснемо будь-яку кнопку, щоб повернутися до головного меню.

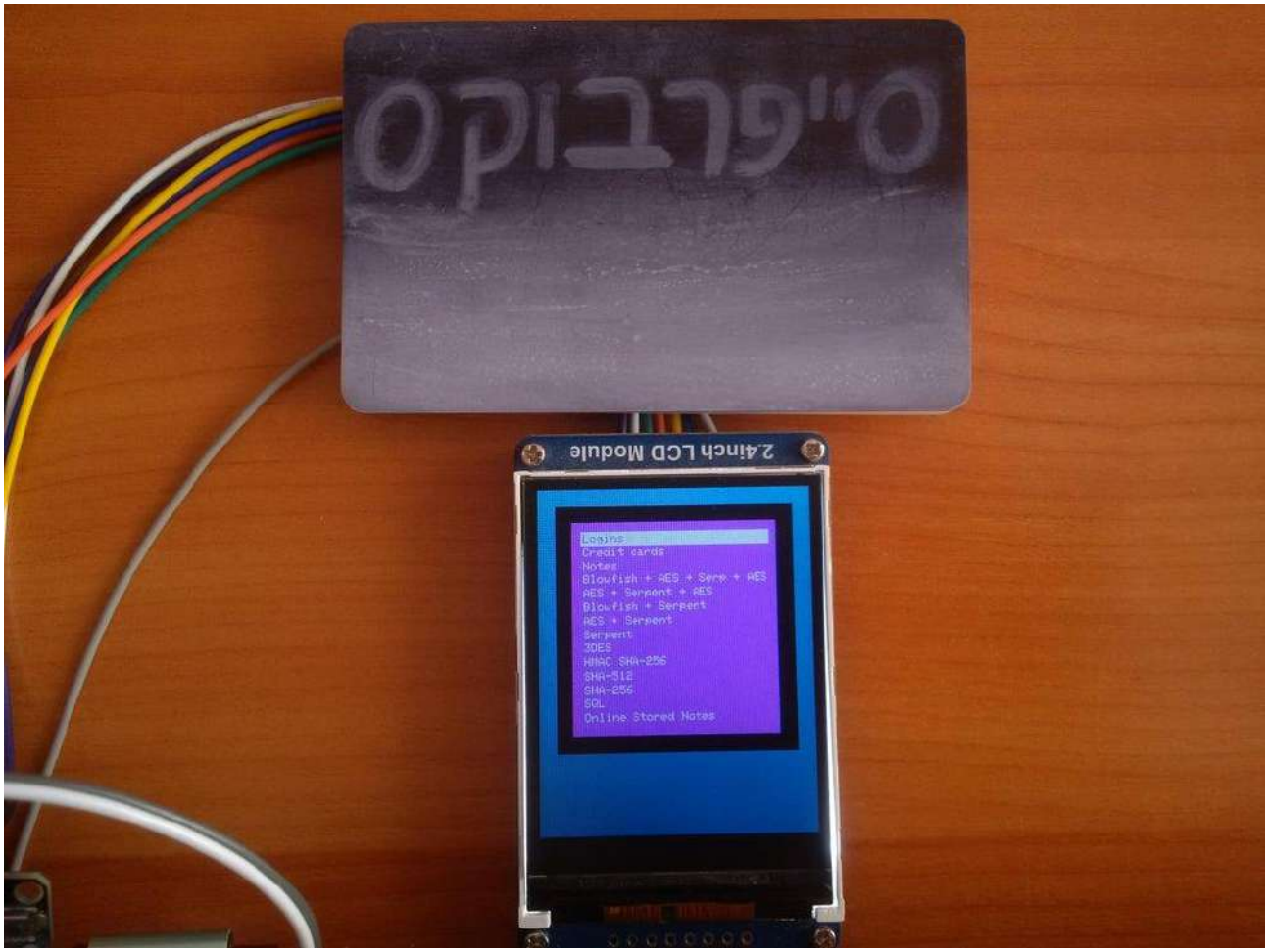

Рисунок 3.16 – Додавання нового логіну

Після успішної реєстрації можна зашифрувати текст за допомогою 6 алгоритмів шифрування. Для цього необхідно виконати такі дії:

1) відкриємо Serial Monitor, тому що зашифрований текст буде надруковано в ньому;

2) виберемо рядок із назвою алгоритму шифрування, який ви бажаєте використати;

3) натиснемо кнопку «A»;

4) виберемо джерело вхідного сигналу;

5) натиснемо кнопку «A»;

6) введемо текст, який потрібно зашифрувати;

7) натиснемо кнопку кодера чотири рази або натисніть кнопку «Надіслати» на моніторі послідовного порту залежно від вибраного джерела введення, щоб зашифрувати текст (рис. 3.17).

Рисунок 3.17 – Процес шифрування тексту

Для того, щоб розшифрувати текст, необхідно виконати наступне:

1) відкриємо Serial Monitor, тому що вам потрібно буде вставити в нього зашифрований текст;

2) виберемо рядок із назвою алгоритму шифрування, який ви використовували для шифрування тексту;

3) натиснемо кнопку «A»;

4) виберемо рядок «Розшифрувати рядок»;

5) натиснемо кнопку «A»;

6) вставимо зашифрований текст у серійний монітор;

7) натиснемо кнопку «Надіслати» на моніторі послідовного порту (рис. 3.18).

Рисунок 3.18 – Процес розшифрування тексту

Розроблений пристрій є потужним інструментом для захисту даних, поєднуючи кілька рівнів шифрування та механізми автентифікації для забезпечення високого рівня безпеки.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи була розроблена система захисту даних IoT на базі мікроконтролерів ESP32 та Arduino, яка дозволила забезпечити високий рівень безпеки інформації під час її передачі та обробки в IoT-системах. Використання сучасних алгоритмів шифрування, таких як AES, оптимізованих для обмежених ресурсів мікроконтролерів, гарантувало конфіденційність переданих даних. Для уникнення несанкціонованого доступу до мережі були реалізовані надійні механізми аутентифікації на основі протоколів TLS.

У процесі виконання дослідження було досягнуто значних результатів, спрямованих на підвищення безпеки IoT-систем. Аналіз сучасних загроз та вразливостей дозволив визначити ключові ризики, зокрема атаки на конфіденційність, доступність і цілісність даних, а також виявити основні проблеми, пов’язані з обмеженими ресурсами IoT-пристроїв та недостатньою стандартизацією протоколів безпеки.

Проведений аналіз існуючих алгоритмів безпеки продемонстрував їхні переваги й недоліки, зокрема недостатню енергоефективність та обмежену стійкість до складних атак.

На основі отриманих результатів було розроблено вдосконалений алгоритм безпеки, який забезпечує баланс між енергоефективністю та рівнем захисту. Оптимізація процесів шифрування та впровадження адаптивних методів аутентифікації дозволили зменшити енергоспоживання пристроїв, забезпечивши при цьому високий рівень захищеності.

Реалізація системи шифрування даних на основі розробленого алгоритму продемонструвала ефективність запропонованого підходу. Тестування підтвердило здатність системи протистояти основним типам атак, зокрема MITM, забезпечуючи при цьому швидкість роботи та низьке споживання ресурсів. Отримані результати є вагомим внеском у розвиток безпечних IoT-технологій, що можуть бути адаптовані до широкого спектра застосувань.

Таким чином, розроблена система є надійним рішенням для захисту IoT-пристроїв, побудованих на базі мікроконтролерів ESP32 та Arduino, і може бути використана в різноманітних застосуваннях, включаючи «розумний дім», промисловий IoT та інші сценарії з високими вимогами до безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Опірський І., Головчак Р., Мойсійчук І., Бальянда Т., Гаранюк С. Проблеми та загрози безпеці IoT пристроїв. *Кібербезпека: освіта, наука, техніка*. 2021. Т. 3, № 11. С. 31-42.

2. Поширені атаки на IoT та захист від них. *Corewin*. URL: https://corewin.ua/blog/attacks-on-iot-how-protect (дата звернення: 10.02.2025).

3. Що таке IoT (Інтернет речей)? *Guru*. URL: https://www.guru99.com/uk/iot-tutorial.html (дата звернення: 10.02.2025).

4. Architecture of Internet of Things (IoT). *Geeksforgeeks.* URL: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/ (дата звернення: 10.02.2025).

5. Інтернет речей: основні складові та їхня роль у сучасному світі. *Cyberset.* URL: https://cyberset.com.ua/hi-tech/iot/ (дата звернення: 28.02.2025).

6. Фундаментальні елементи IoT: розгортання ваших проектів IoT з досвідом. *Cyberset*. URL: https://cyberset.com.ua/hi-tech/iot/ (дата звернення: 28.02.2025).

7. Основні компоненти системи IoT. .*Telesphera.* URL: https://www.telesphera.net/assets/templates/telesphera/img/upload/Blog/30.09.2021.png (дата звернення: 18.03.2025).

8. Міжнародна стандартизація: Стандарти для інформаційної безпеки. *Viconsult.* URL: https://www.viconsult.com/ua/novyny/mizhnarodna-standartyzatsiia-standarty-dlia-informatsiinoi-bezpeky/ (дата звернення: 18.03.2025).

9. Information security standards. *Wikipedia.* URL https://en.wikipedia.org/wiki/Information_security_standards (дата звернення: 30.03.2025).

10. Основні аспекти IoT Security Framework. *Researchgate.* URL: https://www.researchgate.net/publication/327332700/figure/fig1/AS%3A666553041293312%401535930037217/Visual-illustration-of-Proposed-IoT-Security-Framework-IoTSFW.png (дата звернення: 31.03.2025).

11. Модель безпеки інформаційної системи на базі технологій IoT. *Repository*. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/3f887a4c-32be-4f31-9249-74ab0f249e93/content (дата звернення: 31.03.2025).

12. Nessus Professional. *Softonline.* URL: https://softonline.com.ua/ua/catalog/tenable-network-security/nessus-professional/ (дата звернення: 14.04.2025).

13. Cryptography and Network Seurity. *Mrce*. URL: https://mrce.in/ebooks/Cryptography20Network%20Security208t%2Ed.pdf (дата звернення: 14.04.2025).

14. Алгоритм RSA. *Dou.* URL: https://dou.ua/forums/topic/43026/ (дата звернення: 14.04.2025).

15. Шифрування: типи і алгоритми. Що це, чим відрізняються і де використовуються? *Hostpro*. URL: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms/ (дата звернення: 30.04.2025).

16. Datagram Transport Layer Security. *Wikipedia.* URL: https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security (дата звернення: 30.04.2025).

17. Схема функціонування протоколу DTLS. *Wallarm.* URL: https://www.wallarm.com/what/what-is-datagram-transport-layer-security-dtls (дата звернення: 30.04.2025).

18. What is MQTT? *Wallarm.* URL: https://www.wallarm.com/what/what-is-datagram-transport-layer-security-dtls (дата звернення: 05.11.2024).

19. Мікроконтролер ESP32. *Itmaster.* URL: https://itmaster.biz.ua/directory/microcontrollers/esp32.html (дата звернення: 30.04.2025).

20. Arduino Uno. *Arduino.* URL: https://doc.arduino.ua/ru/hardware/Uno (дата звернення: 10.11.2024).

21. TFT LCD з ILI9341. *Radiostore.* URL: https://radiostore.ua/products/modul-lcd-nokia-5110-2 (дата звернення: 30.04.2025).

22. RFID-модуль. *Arduino* URL: https://arduino.ua/ (дата звернення: 10.11.2024).

23. Енкодер EC11. *Mini-tech.* URL: https://www.mini tech.com.ua/ua/encoder -s- knopkoy-ec11 (дата звернення: 30.04.2025).

ДОДАТКИ

Додаток А

Лістинг коду для мікроконтролера Arduino

```
/*
Cipherbox
Distributed under the MIT License
© Copyright Maxim Bortnikov 2022
For more information please visit
https://github.com/Northstrix/Cipherbox
https://sourceforge.net/projects/mcu-cipherbox
https://osdn.net/projects/cipherbox
Required libraries:
https://github.com/zhouyangchao/AES
https://github.com/peterferrie/serpent
https://github.com/ddokkaebi/Blowfish
https://github.com/ulwanski/sha512
https://github.com/adafruit/Adafruit-GFX-Library
https://github.com/adafruit/Adafruit_ILI9341
https://github.com/adafruit/Adafruit_BusIO
https://github.com/GyverLibs/GyverBus
https://github.com/siara-cc/esp32_arduino_sqlite3_lib
https://github.com/miguelbalboa/rfid
https://github.com/intrbiz/arduino-crypto
https://github.com/Northstrix/DES_and_3DES_Library_for_MCUs
https://github.com/GyverLibs/EncButton
https://github.com/mathworks/thingspeak-arduino
*/
#include <SPI.h>
#include <MFRC522.h>
#include <SoftwareSerial.h>
#include <EncButton2.h>
EncButton2 < EB_BTN > a_button(INPUT, 7);
EncButton2 < EB_BTN > b_button(INPUT, 8);
SoftwareSerial mySerial(5, 4);
#include "GBUS.h";
GBUS bus( & mySerial, 6, 2);
```

```
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

struct myStruct {
  char x;
  bool d;
};

void setup() {
  //Serial.begin(115200);
  SPI.begin();
  mfrc522.PCD_Init();
  //Serial.println("Approximate four cards to the reader...");
  mySerial.begin(9600);
}

void loop() {
  a_button.tick();
  if (a_button.press()) {
    myStruct data;
    data.d = true;
    data.x = 1;
    bus.sendData(3, data);
  }

  b_button.tick();
  if (b_button.press()) {
    myStruct data;
    data.d = true;
    data.x = 2;
    bus.sendData(3, data);
  }

  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
```

```
  }
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  for (int i = 0; i < 4; i++) {
    //Serial.println(mfrc522.uid.uidByte[i]);
    myStruct data;
    data.d = false;
    data.x = (char) int(mfrc522.uid.uidByte[i]);
    bus.sendData(3, data);
    delay(12);
    if (i == 3) {
      delay(700);
    }
  }
  //Serial.println();
}
```