Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

(тема)

Методики тестування продуктивності сучасних легковагових шифрів

Виконав: студент 2 курсу, групи БІКСм-19-1 Кліщ І.Р.
(прізвище, ініціали)

Спеціальність 125 Кібербезпека
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва освітньої програми)

Керівник проф. Руженцев В.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Халімов Г.З.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Безпеки інформаційних технологій
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)
Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові *Кліщу Ігорю Романовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методики тестування продуктивності сучасних легковагових шифрів* затверджена наказом по університету від " " 2020 р. №____
2. Термін подання студентом роботи (проекту) 12. 12.2020
3. Вихідні дані до роботи (проекту) Теоретичні дані про легковагову криптографію
4. Перелік питань, що потрібно опрацювати в роботі (зміст пояснювальної записки)
1. Актуальний стан та перспективи розвитку сучасних систем, що використовують легковагове шифрування
2. Сучасні системи в яких використовуються легковагові алгоритми
3. Методи оцінки ефективності легковагових алгоритмів
4. Сучасні блокові легковагові алгоритми
5. Сучасні потокові легковагові алгоритми
6. Методика порівняння властивостей сучасних і перспективних легковагових алгоритмів
7. Застосування фреймворку FELICS для аналізу та порівняння легковагових алгоритмів
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *Презентаційний матеріал у вигляді слайдів*
.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської атестаційної роботи	Термін виконання етапів роботи	Примітка
1	*Отримання завдання*	*02.09.20*	
2	*Пошук літератури*	*04.09.20-28.09.20*	
3	*Аналіз властивостей легковагових алгоритмів*	*28.09.20-15.10.20*	
4	*Методика аналізу та порівняння алгоритмів*	*15.10.20-11.11.20*	
5	*Аналіз результатів тестів*	*11.11.20-25.11.20*	
6	*Обґрунтування вибору легковагового алгоритма з найкращим рівнем безпеки і швидкодії*	*25.11.20-04.12.20*	
7	*Оформлення пояснювальної записки*	*04.12.20-16.12.20*	

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи (проекту) _____ __проф. Руженцев В.І..____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 68 с., 7 табл., 15 рис., 1 дод., 33 джерела.

LW-ШИФРИ, ЛЕГКОВАГОВІСТЬ, RFID, FELICS.

Об’єкт дослідження – легковагова криптографія.

Предмет дослідження – методики тестування продуктивності сучасних легковагових шифрів

Методи дослідження – теоретичні дослідження, тестування.

Мета магістерської атестаційної роботи – визначення методик тестування та проведення тестів над сучасними легковаговими шифрами.

У магістерської атестаційній роботі проведено аналіз актуальності питання використання легковагової криптографії, основні проблеми та сучасний стан технологій, наведено приклади використання та напрямки розвитку. Також в другому розділі атестаційної роботи розглянуто всі відомі сучасні легковагові шифри і визначено їх переваги з недоліками. Було розглянуто як блочні так і потокові шифри й визначено найефективніші. В третьому розділі атестаційної роботи було обрано метод тестування та проведено тести над 2ма алгоритмами. Було проаналізовано результати і зроблено висновки.

ABSTRACT

Explanatory note contains: 68 pp., 7 tables, 15 figures, 1 appendix, 33 sources.

LW-CIPHERS, LIGHT WEIGHT, RFID, FELICS.

The object of research is lightweight cryptography.

The subject of research - methods of testing the performance of modern lightweight ciphers

Research methods - theoretical research, testing.

The purpose of the master's certification work is to determine the methods of testing and conducting tests on modern light ciphers.

In the master's attestation work described the analysis of urgency of a question of light cryptography uses, the basic problems and a modern condition of technologies is carried out, examples of use and directions of development are resulted. Also in the second section of the attestation work all known modern lightweight ciphers are considered and their advantages and disadvantages was considered. Both block and stream ciphers were considered and the most effective ones were identified. In the third section of the certification work, the testing method was chosen and tests were performed on 2 algorithms.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

LW – lightweight

SSH – secure shell

HTTPS – hypertext transfer protocol secure

VOIP – voice over internet protocol

ЕОМ – електронна обчислювальна машина

IOT – internet of things

RFID radio frequency identification

GE – gate equivalent

CMOS – complemental metal oxide semiconductor

СКУД – система контроля та управління доступом

ПЛІС - програмована логічна інтегральна схема

FELICS – fair evaluation of lightweight cryptographic systems

ВСТУП

Однією з визначальних тенденцій IT-ландшафту цього століття буде широке розгортання крихітних обчислювальних пристроїв. Ці пристрої не тільки будуть регулярно функціонувати в споживчих товарах, але вони становитимуть невід'ємну частину всеосяжного - і невидимого інтернету речей. Вже визнано, що такі пристрої приносять цілий ряд дуже конкретних ризиків для безпеки. Проте водночас криптографічні рішення, і особливо криптографічні примітиви, які ми маємо на сьогодні є незадовільними для середовищ з надзвичайно обмеженими ресурсами.

Робота присвячена огляду алгоритмів легковагової криптографії, стійкість яких знижується незначно, на відміну від обсягу необхідних ресурсів. Даний повний огляд літератури в області легковагової криптографії. Наданий аналіз застосування легковагових блокових і потокових шифрів. Зроблено висновок про обнадійливих перспективи.

Об'єкт дослідження – «легковагова» криптографія.

Предмет дослідження – методики тестування продуктивності сучасних легковагових шифрів

Методи дослідження – теоретичні дослідження, тестування.

Мета магістерської атестаційної роботи – визначення методів тестування та проведення тестів над сучасними легковаговими шифрами.

Для досягнення поставленої мети потрібно рішення наступних завдань:

– скласти критерії оцінки, та обрати фреймворк на якому будуть проводитися тести алгоритмів;
– порівняти апаратні реалізації алгоритмів
– провести тести ефективності
– порівняти результати тестів
– визначити найефективніші алгоритми
– оформити пояснювальну записку згідно з методичними вказівками [1], та вимогами ДСТУ 3008:2015 [2].та згідно з положеннями [3-7].

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМ ВИКОРИСТАННЯ «ЛЕГКОВАГОВИХ» ШИФРІВ

1.1 Постановка проблеми застосування систем з «легковаговим» шифруванням

Криптографія — це наука що вивчає математичні методи перетворення інформації для виконання необхідних задач.

Основними задачами сучасної криптографії є — забезпечення конфіденційності повідомлень, перевірка їх цілісності та ідентифікація учасників системи комунікації. Окрім цього також застосовується в таких технологіях як: SSH, HTTPS, цифровий підпис, криптовалюта, електронні контракти та захищений VOIP.

Криптографічна система — це алгоритм математичних перетворень вхідного тексту в шифротекст. Основним недоліком криптосистем є те, що обираючи криптосистему потрібно обирати компроміс між швидкодією, ціною реалізації, розміром системи та її надійністю. Чим складніший для криптоаналізу шифр тим довше він працює і більше потребує ресурсів. Тому характеристики криптосистеми залежать напряму від задач які ця система виконує. Зазвичай проводиться оцінка важливості інформації та вірогідність атаки і обирається така система цінність інформації якої менша ніж ціна атаки, або атака на котру займе більше часу і ресурсів ніж атакуючий готовий затратити.

У сучасних комп’ютерів достатньо продуктивності та ресурсів, щоб забезпечити надійне функціонування криптосистем без оптимізації на апаратному рівні. Але зараз за приходом IOT постає проблема у створенні криптосистем, що будуть займати якомога менше місця на платі, будуть недорогі у виробнитстві, достатньо захищеними та атакостійкими. Тому для таких задач були розроблені «легковагові» шифри та «легковагова» криптографія.

«Легковагова» криптографія – розділ криптографії, що має на меті розробку алгоритмів для застосування в пристроях, які не здатні забезпечити більшість існуючих шифрів достатніми ресурсами (пам’ять, електроживлення, розміри) для функціонування.

Більшість сучасних алгоритмів захисту інформації і, зокрема, шифрування, розраховані на застосування в ЕОМ в складі програмних комплексів без урахування оптимізації на апаратному рівні забезпечення. Цей факт робить неможливим застосування більшості існуючих криптографічних алгоритмів в пристроях з обмеженою обчислювальною потужністю, малим обсягом і малим енергоспоживанням. Методи криптографічного захисту даних в системах з низькою вартістю стали основою «легковагової» криптографії.

Особливої актуальності «легка» криптографія набуває в світлі розвитку ідеї IOT («інтернету речей»), який являє собою бездротову мережу, що самоконфігурується між об’єктами різного класу, прикладом можуть бути побутові прилади, транспортні засоби, інтелектуальні датчики і мітки радіочастотної ідентифікації (RFID).

Найчастіше, розробники легковажних алгоритмів змушені вибирати між трьома, часом взаємовиключними, вимогами до алгоритмів: безпекою, вартістю і продуктивністю. На практиці не складає труднощів оптимізувати будь-які два параметра: безпеку і вартість, безпеку і продуктивність або вартість і продуктивність, однак дуже важко оптимізувати всі три одночасно. У зв’язку з цим існує досить багато реалізацій легковажних алгоритмів криптографії: як програмних, так і апаратних. У них різні, а іноді й протилежні характеристики.

Стандартними підходами до вирішення проблеми створення ефективних методів і засобів «легковагової» криптографії є:

- використання класичних криптографічних алгоритмів, якщо це можливо;
- модифікація класичних алгоритмів з адаптацією до апаратних особливостей і обмежень систем з низькою вартістю;
- розробка нових спеціалізованих рішень в методологічному, алгоритмічній і програмно-апаратному плані;

Кожен з цих підходів має свої недоліки. До цих пір більшість рішень в цій галузі знання відноситься до третього підходу, і показують непогані результати. При цьому, однак, слід пам'ятати, що при адаптації криптографічного алгоритму до особливостей апаратного базису в умовах обмеженості ресурсів, можуть бути небажані наслідки. Вони можуть виражатися в появі додаткових слабкостей алгоритму або в ослабленні їх загальної стійкості.

Перш ніж говорити про приклади реалізації схем «легковагової» криптографії, ми повинні сформулювати критерії пошуку таких криптографічних алгоритмів.

Основні критерії до «легковагових» шифрів, по-перше, це вічний пошук балансу між надійністю, продуктивністю і ціною.

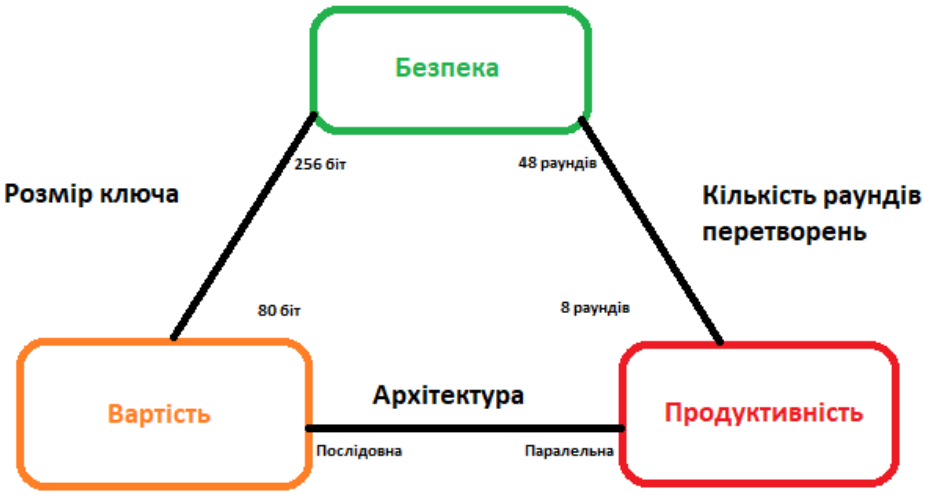

Рисунок 1.1 - Схема взаємодії надійності, продуктивності і ціни [8]

Для блочних шифрів розмір ключа визначає співвідношення надійності та вартості, число раундів шифрування – надійність та продуктивність, а особливості апаратної конструкції – продуктивність і ціна. Як правило, будь-які дві з трьох цілей розробки можуть бути легко досягнуті, в той час як задоволення всіх трьох вимог – вкрай складне завдання. Наприклад, можна забезпечити

прийнятне співвідношення між надійністю і продуктивністю, однак для реалізації подібного алгоритму буде потрібно велика площа на схемі, що призводить до підвищення вартості. З іншого боку, можна створити надійну і дешеву систему, але з обмеженою продуктивністю. По-друге, це займає площу мікросхеми. По-третє, важливо енергоспоживання схеми і, відповідно, вид самої схеми (пасивна або активна), в залежності від якої будуть накладатися додаткові вимоги на схему.

Чим же «легковагові» алгоритми шифрування відрізняються від універсальних? Ось основні підходи, що дозволяють криптографії створити невимогливі до ресурсів і при цьому відносно стійкі алгоритми шифрування:

- зменшення (до розумних меж) розмірів основних параметрів алгоритму: блоку шифрованих даних, ключа шифрування і внутрішнього стану алгоритму;
- спроби компенсації вимушеної втрати стійкості алгоритмів за рахунок проектування на основі добре вивчених, широко застосовуваних операцій, які здійснюють елементарні лінійні та нелінійні перетворення. Такі операції можна уявити як деталі якогось конструктора, з яких криптографи «збирають» алгоритм, що володіє потрібними якостями;
- зменшення обсягів даних, що використовуються в конкретних операціях. Наприклад, в алгоритмах шифрування часто застосовуються таблиці замін; щоб зберігати таблицю, яка замінює 8-бітові фрагменти даних, необхідно 256 байт, але таку таблицю можна скласти з комбінації двох 4-бітових таблиць, що вимагають всього 32 байта в сумі (даний підхід обрали автори алгоритму Curupira);
- використання «дешевих» з точки зору ресурсоємності, але ефективних перетворень, таких як керовані бітові перестановки (в яких вибирається конкретний варіант перестановки в залежності від значення керуючого біта; цим бітом може бути, наприклад, певний біт ключа), реєстри зсуву і ін .;
- застосування перетворень, щодо яких можливі варіанти реалізації в

залежності від ресурсів конкретного шифратора (наприклад, зменшення вимог до пам’яті, але також зменшення швидкості шифрування, або навпаки).

Слід зазначити, що «полегшені» алгоритми шифрування створюються або для систем з низьким або середнім рівнем безпеки, або для систем, де буде врахована специфіка використовуваних алгоритмів і буде знайдено рішення, що дозволяє зробити реалізацію алгоритму максимально безпечною для його рівня стійкості.

Одним з основних понять, що використовуються при розгляді «легковажних» алгоритмів криптографії, є GE - Gate equivalent (еквівалентний логічний елемент). Ця величина являє собою одиницю виміру, яка дозволяє визначити виробничу складність технології незалежно від складності цифрових електронних схем.

Для поточної CMOS технології еквівалентним логічним елементом є НЕ-І з двома входами. У таблиці 1.1 представлені короткі відомості за складністю реалізації різних логічних елементів.

Таблиця 1.1 – GE-вимоги до площі стандартних логічних елементів[8]

Елемент	Технологічний процес, нм	Площа, нм^2	GE
НЕ	0,18	6,451	0,67
І-НЕ	0,18	9,677	1
АБО-НЕ	0,18	9,677	1
І	0,18	12,902	1,33
АБО	0,18	12,902	2,33
Мультиплекс	0,18	22,579	2,67
ВИКЛЮЧНЕ АБО-НЕ	0,18	25,805	4,67

Таким чином, при розробці алгоритму, який претендує на «легковаговість», необхідно дотримуватися обмеження на кількість еквівалентних логічних

елементів, з використанням яких може бути здійснена апаратна реалізація алгоритму. Спочатку вважалося, що максимальною межею буде від 2000 до 3000 GE, проте на даний момент, з розвитком «легковагової» криптографії, даний поріг був знижений до 1000 GE.

Зумовлюючи вимоги до реалізації схем на основі «легковагової» криптографії, можна перейти до опису самих алгоритмів. Як і у всіх випадках, існують алгоритми, які прийнято вважати за еталон. У криптографії таким алгоритмом є AES. Стандартом є алгоритм 1997 року, чия пропускна здатність доходить до 70 Гбіт/с (2004). Однак, дане рішення абсолютно не підходить для «легковагової» криптографії, так як апаратна реалізація потребує більше 250 000 GE. У той же час існує більш компактна реалізація AES, датована 2006 роком, її показник становить 3100-3400 GE. На жаль, при сучасних вимогах до «легковагової» криптографії даний показник також є перевищенням всіх допустимих норм.

Таким чином, основним критерієм пошуку «легковагових» алгоритмів є відповідність вимогу до апаратної реалізації алгоритму. Варто зауважити, що на даний момент поріг в 1000 GE вдалося подолати лише невеликій кількості алгоритмів.

1.2 Використання «легковагових» шифрів в RFID-системах

RFID-системи є однією зі сфер, що швидко розвиваються. У сфері комп’ютеризації невеликих пристроїв. RFID (англ. Radio Frequency IDentification, радіочастотна ідентифікація) – це метод автоматичної ідентифікації об’єктів, в якому за допомогою радіосигналів зчитуються або записуються дані, що зберігаються в так званих RFID-мітках. Будь яка RFID-система складається з пристрою, що зчитує і RFID-мітки (іноді також застосовується термін RFID-теґ). Більшість RFID-міток складається з двох частин: інтегральної схеми для зберігання і обробки інформації і антени для прийому і передачі сигналу.

Фізичні принципи (принаймні, для більшості частотних діапазонів)

нагадують роботу трансформатора або системи пов’язаних контурів. Як показано на рисунку (1.2), якщо взяти дві котушки і розмістити їх не дуже далеко один від одного, то вони будуть надавати один на одного взаємний вплив

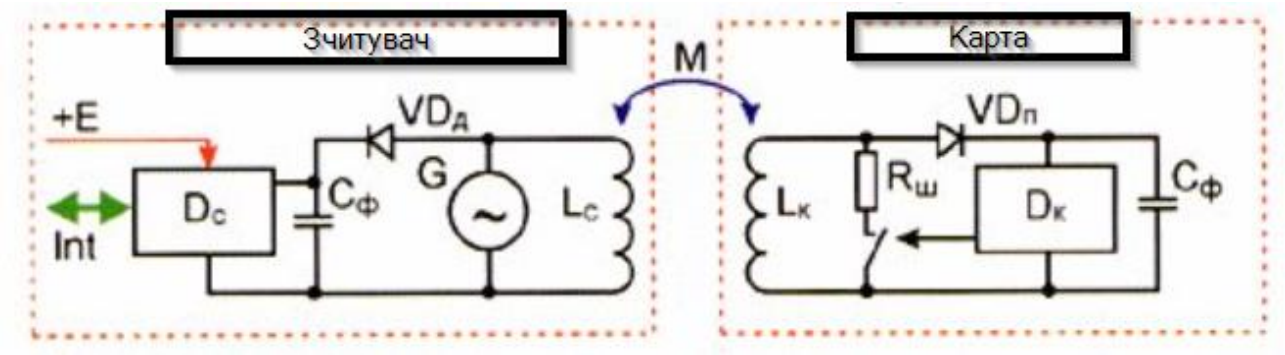

Рисунок 1.2 – Принцип роботи пари «зчитувач-ідентифікатор»

Зчитувач містить генератор високої частоти G, який живить антену зчитувача Lc. За рахунок наявності електромагнітного зв’язку M між антеною зчитувача і антеною ідентифікатора (карти) LK в останній наводиться змінна напруга, величина якої залежить від конструкції і відстані між картою і зчитувачем. Наведена напруга використовується для живлення мікросхеми карти DK через випрямляч, утворений діодом VDп і фільтрує конденсатором Сф. Мікросхема карти DK модулює напругу в антені IK шляхом її шунтування резистором Кш. За рахунок зв’язку антен модуляція з’являється в антені зчитувача Lc, детектується діодом VDд і надходить на мікросхему зчитуючи Dc, яка дешифриує код карти і передає його на контроллер через інтерфейс Int. За таким принципом працювали перші пасивні R / O (Read Only - тільки для читання) Proximity-карти і зчитувачі. Потім були створені ідентифікатори, здатні не тільки передавати інформацію зчитувача, а й отримувати її для цілей програмування (записи інформації в енергонезалежну пам’ять).

За типом джерела живлення RFID-мітки діляться на дві категорії: пасивні, активні та напівпасивні. Пасивні RFID-мітки не мають вбудованого джерела енергії. Електричний струм, індукований в антені електромагнітним сигналом від зчитувача, забезпечує достатню потужність для функціонування кремнієвого

чіпа, розміщеного в мітці, і передачі сигналу у відповідь. Активні RFID-мітки володіють власним джерелом живлення і не залежать від енергії зчитувача, внаслідок чого вони читаються на далекій відстані, мають великі розміри і можуть бути оснащені додатковим електронікою. Однак, такі мітки найбільш дорогі, а у батарей обмежений час роботи. Напівпасивну RFID-мітки дуже схожі на пасивні мітки, але оснащені батареєю, яка забезпечує чіп енергоживленням. На поточний момент RFID-технології використовуються в найрізноманітніших сферах: від сільського господарства до транспорту.

Технологія радіочастотної ідентифікації з'явилася на світ близько 20 років тому і весь цей період формувалася темпами, що випереджають комп'ютерні технології. Особливо інтенсивно RFID удосконалювалася в останні 5-7 років. Пояснити це можна двома факторами: по-перше, розвиток мікроелектроніки дозволяє реалізувати багато ідей, раніше недоступних з технологічних причин, а по-друге, з'явилися стандарти застосування яких забезпечило сумісність технічних рішень від різних виробників.

Область застосування «легковагових» алгоритмів криптографії в RFID-системах обмежена перевагами і недоліками даного сімейства алгоритмів. Практика показала, що однією з областей, в якій «легковагові» алгоритми отримали величезну популярність, є системи доступу. Це можна пояснити двома факторами:

- по-перше, простотою реалізації самої технології у СКУД (досить використовувати ідентифікатори read-only (тільки для читання) з невеликою - в три або чотири байти - довжиною коду);
- по-друге, зручністю в порівнянні з будь-якими іншими типами ідентифікаторів: контактних, з магнітною смугою, Wiegand.

А оскільки, за даними IdTechEx, в 2015 році було виготовлено 2 мільярди активних RFID-міток і близько трильйона пасивних, активно з'являються проблеми захисту RFID-міток, для чого і можуть використовуватися алгоритми «легковагової» криптографії. Окремо варто відзначити перспективність використання «легковагових» алгоритмів при бездротовій взаємодії між

побутовими приладами, транспортними засобами, пожежними і охоронними системами.

Враховуючи широку різноманітність використання RFID технології можна зробити висновок, що дані технології незабаром стануть повсюдно поширеними. Однак не варто забувати про проблеми безпеки. Особливо гостро це питання стоїть при застосуванні RFID-технологій у військовій або фінансовій сферах. Через жорсткі цінові обмежень, система захисту повинна бути не тільки надійною і продуктивною, але й дешевою в реалізації.

Основною проблемою в забезпеченні безпеки RFID-міток, є збереження конфіденційності інформації, записаної в мітці. Це викликано тим, що технологія RFID дозволяє зчитувати інформацію з відстані кількох метрів. У зв'язку з цим, правозахисники часто виступають проти масового поширення RFID-міток, обґрунтовуючи це можливістю вторгнення в приватне життя. Можливий наступний сценарій несанкціонованого використання RFID-мітки: зловмисник, використовуючи пристрій, що зчитує, здатний отримати ідентифікатори жертви і використовувати отриману інформацію проти неї (дістати несанкціонований доступ до бази даних і отримати інформацію про переміщення або інформацію про покупки). Інакше кажучи, технологія RFID-міток крім безлічі переваг має і ряд недоліків, не дозволяє впровадити її повсюди.

Шифрування в пасивних радіомітках представляє особливий випадок навіть серед пристроїв з вкрай обмеженими обчислювальними ресурсами. Пасивні радіопозначки не мають власного джерела живлення, тому вони активуються за допомогою наведеного сигналу зчитувачем. Таким чином, шифрування в чіпах має бути найменш ресурсоємним радіопозначка повинна до загасання наведеного сигналу встигнути зашифрувати дані і передати свою відповідь назад до зчитувача. Через серйозних обмежень на внутрішні обчислювальні ресурси RFID-міток стає неможливо використовувати існуючі криптографічні алгоритми, що створює актуальність розробки «легковагових» алгоритмів, які відповідають даним обмеженням. Один із способів захисту транспортних засобів – установка автомобільної сигналізації, спирається на

технологію аутентифікації через незахищений канал, тоді як діалогові коди, особливо у дешевих моделей, передаються у відкритому вигляді. Застосування «легковагових» алгоритмів дозволить підвищити надійність захисту при мінімальних витратах ресурсів і електроспоживання. Великого поширення набули бездротові пожежні і охоронні сповіщувачі. Вони зручні в монтажі та експлуатації, мають гнучкі параметри налаштування, в тому числі мають можливості динамічної маршрутизації. Однак при отриманні зловмисником доступу до системи управління сповіщувачами, він має можливість створити помилкові спрацьовування або зовсім відключити захист, що гостро ставить питання про захист інформації, що передається в таких системах. В якості одного з варіантів вирішення даної проблеми можуть виступити «легковагові» алгоритми.

Таким чином, мінімальні вимоги до обчислювальної потужності і низьке енергоспоживання при порівняно високій стійкості - це поєднання дозволяє забезпечити конфіденційність інформації не тільки при роботі з персональними даними, що зберігаються на RFID-мітці, а й при бездротовому взаємодії між різними пристроями, що містять в собі інформацію, компрометація якої небажана. Загальна комп'ютеризація призводить до того, що більшість речей, побутових приладів, що оточують нас, отримують доступ до бездротових мереж, при цьому, в зв'язку з економічними витратами, рідкісний виробник турбується про безпеку переданої ними інформації. Застосування «легковагових» алгоритмів якщо і не знищить загрозу повністю, то, як мінімум, зменшить ймовірність її реалізації.

1.3 Використання легковагової криптографії у хмарових сервісах

Хмарні обчислення отримали значний розвиток в останні роки. Ця технологія має свої переваги, вона надає доступ до призначених для користувача даних в будь-який час в будь-якому місці. Також, як правило, хмарні ресурси в значній міру перевершують ресурси, якими володіють звичайні користувачі, тому можуть вирішувати багаточисленні та складні обчислення за коротший час.

Але як і будь-яка система хмарні обчислення мають недоліки зі сторони захисту інформації. Розглянемо основні ризики порушення інформаційної безпеки в хмарних обчисленнях:

- доступ до даних з боку провайдера (зловмисний інсайдер);
- публічне розголошення даних (доступ необмеженого кола осіб);
- винос / виїмка даних або носіїв з датацентру провайдера (органи, співробітники);
- помилки ізоляції середовища (доступ одного клієнта хмари до даних інших клієнтів);
- недостатнє знищення даних провайдером при відході клієнта або стирання даних.

Для забезпечення надійного захисту інформації необхідно використання стійких криптографічних алгоритмів.

З іншого боку, хмарна модель повинна підтримувати високу доступність сервісів, а також задовольняти властивості «швидкої еластичності» (rapid elasticity), а саме: обчислювальні можливості повинні надаватися швидко і гнучко «змінюваний обсяг», в ряді випадків - автоматично, для оперативного підвищення масштабованості (scale out) і швидкого звільнення для зменшення масштабів споживання (scale in).

Щоб забезпечити постійний доступ користувача до його ресурсів в хмарі, необхідно підтримувати безпечні технології передачі і зберігання даних незалежно від можливостей клієнта в даний момент. Зокрема, необхідно забезпечити конфіденційність даних при зберіганні і передачі за допомогою шифрування. Клієнту дозволено звертатися до хмари за допомогою практично будь-якого пристрою, але воно може бути не в змозі підтримувати шифрування деякими алгоритмами зважаючи на складність і енергоємності їх реалізації. У такому випадку можна використовувати алгоритми «легковагової» криптографії, стійкість яких знижується незначно, на відміну від обсягу необхідних ресурсів.

Таким чином, наступаюча ера обчислювальної техніки буде відрізнятися наявністю

багатьох смарт-пристроїв, які в зв’язку з жорстким обмеженням вартості, характерним для масового впровадження, будуть мати обмежені ресурси, такі як пам’ять, обчислювальна потужність і час роботи від батареї. В цьому випадку необхідно тлумачити закон Мура по-іншому: замість того щоб подвоювати продуктивність, ми бачимо зменшення в два рази ціни на обчислювальну потужність кожні 18 місяців. Так як багато пристроїв мають дуже жорсткі обмеження ресурсів, наприклад RFID, з плином часу закон Мура буде більше впливати на них. Багато пристроїв будуть обробляти таку важливу
інформацію, як спостереження за здоров’ям або біометричні дані, тому попит на криптографічні компоненти, які можуть бути ефективно реалізовані, сильно зростає.

1.4 Постановка задач досліджень

В магістерській атестаційній роботі необхідно визначити ефективність сучасних «лековагових» алгоритмів перед теперішніми задачами з захисту інформації та визначити найефективніші методом порівняння за чіткими критеріями.

Для цього наведено аналіз блокових і поточних шифрів розроблених різними країнами в різний час і визначено їх недоліки та переваги.

В результаті визначено, що до основних досліджуваних характеристик відносяться:

– Енергоефективність;
– Розмір фізичної реалізації;
– Вартість;
– Криптостійкість до відомих атак;

Таким чином, метою магістерської атестаційної роботи є визначення найефективнішого «легковагового» алгоритму шифрування.

Для досягнення поставленої мети потрібно рішення наступних завдань:

– скласти критерії оцінки, та обрати фреймворк на якому будуть

проводитися тести алгоритмів;

– порівняти апаратні реалізації алгоритмів
– провести тести ефективності
– порівняти результати тестів
– визначити найефективніші алгоритми
– оформити пояснювальну записку згідно з методичними вказівками [1], та вимогами ДСТУ 3008:2015 [2].

1.5 Висновки до 1-го розділу

В першому розділі проаналізовано основні критерії до «легковагови» шифрів та надано пояснення до них і терміну «легковаговий». Указано сучасні галузі де використовується ця технологія, та наведено потенційні напрямки розвитку.

Найчастіше, розробники легковажних алгоритмів змушені вибирати між трьома, часом взаємовиключними, вимогами до алгоритмів: безпекою, вартістю і продуктивністю. На практиці не складає труднощів оптимізувати будь-які два параметра: безпека і вартість, безпеку і продуктивність або вартість і продуктивність, однак дуже важко оптимізувати всі три одночасно. У зв'язку з цим існує досить багато реалізацій легковажних алгоритмів криптографії: як програмних, так і апаратних. У них різні, а іноді й протилежні характеристики.

Основною технологією, що використовує «легковагову» криптографію є RFID-мітки. Технологія радіочастотної ідентифікації з'явилася на світ близько 20 років тому і весь цей період формувалася темпами, що випереджають комп'ютерні технології. Особливо інтенсивно RFID удосконалювалася в останні 5-7 років.Область застосування «легковагових» алгоритмів криптографії в RFID-системах обмежена перевагами і недоліками даного сімейства алгоритмів.

Враховуючи широку різноманітність використання RFID технології можна зробити висновок, що дані технології незабаром стануть повсюдно поширеними. Однак не варто забувати про проблеми безпеки. Особливо гостро це питання

стоїть при застосуванні RFID-технологій у військовій або фінансовій сферах.

Таким чином, наступаюча ера обчислювальної техніки буде відрізнятися наявністю багатьох смарт-пристроїв, які в зв'язку з жорстким обмеженням вартості, характерним для масового впровадження, будуть мати обмежені ресурси, такі як пам'ять, обчислювальна потужність і час роботи від батареї. В цьому випадку необхідно тлумачити закон Мура по-іншому: замість того щоб подвоювати продуктивність, ми бачимо зменшення в два рази ціни на обчислювальну потужність кожні 18 місяців. Так як багато пристроїв мають дуже жорсткі обмеження ресурсів, наприклад RFID, з плином часу закон Мура буде більше впливати на них. Багато пристроїв будуть обробляти таку важливу інформацію, як спостереження за здоров'ям або біометричні дані, тому попит на криптографічні компоненти, які можуть бути ефективно реалізовані, сильно зростає.

РОЗДІЛ 2
АНАЛІЗ ЕФЕКТИВНОСТІ «ЛЕГКОВАГОВИХ» ШИФРІВ

2.1 Визначення сучасних вимог до «легковагових» шифрів

Як вже було зазначено, головною особливістю сучасного етапу розвитку Інтернету є зростаюча кількість найрізноманітніших інтелектуальних пристроїв, що мають доступ в Інтернет. В силу умов їх функціонування, а також жорстких цінових обмежень, типового для масового виробництва, ці пристрої характеризуються значними обмеженнями на використовувані ресурси пам'яті, обчислювальну потужність, джерела живлення і т.д. Звідси виходять обмеження на використовувані технології і технологічні рішення «легковагової» криптографії. Так, наприклад, жорсткі обмеження накладаються на енерговитратність реалізації криптографічних алгоритмів для пасивних інтелектуальних пристроїв таких, як радіочастотна мітка або безконтактні смарт-карти. Відповідно до стандарту ISO / IEC 8 пасивні RFID-мітки повинні мати рівень енергоспоживання не більше 15 μW для того, щоб гарантувати роботу пристрою в радіусі до 1 м. Останнє, в свою чергу, обмежує можливості, наприклад, в розпаралелювання обчислень з метою збільшення швидкодії алгоритму. Таким чином, типовими обмеженнями, які зустрічаються в «легковаговій» криптографії, є: для апаратної реалізації — розмір мікросхеми, споживана енергія, час затрачений на виконання програми; для програмної реалізації - розмір програмного коду, використання оперативної пам'яті, час витрачений на виконання програми. Можуть з'являтися й інші обмеження. Так в залежності від конкретних умов застосування розроблюваного пристрою важливою може виявитися така характеристика, як ширина смуги робочих частот каналу зв'язку.

Кожен проектувальник в області «легковагової» криптографії повинен прагнути знайти баланс між безпекою, ціною і продуктивністю. Зазвичай легко

оптимізувати будь-які дві з трьох цілей розробки, дуже важко оптимізувати всі три параметра одночасно. Наприклад, безпечна і високопродуктивна апаратна реалізація може бути досягнута на конвеєрній архітектурі, стійкої до витоку інформації з побічних каналів, що веде до збільшення розміра мікросхеми і відповідно зростанню її вартості. З іншого боку, можна спроектувати безпечне, недороге обладнання, яке має однак, обмежену продуктивність. Нарешті, ефективність реалізації того чи іншого перетворення на програмному або апаратному рівні оцінюється по-різному. Для порівняння програмних реалізацій прийнято розглядати вимоги до пам’яті та час роботи, яка вимірюється в тактах процесора. для аппаратної реалізації критерієм ефективності являється насамперед розмір мікросхеми і час роботи в тактах процесора, хоча для багатьох важливим фактором є енергоспоживання пристрою. Більшість вимог, пред’явлених до алгоритмів, призначеним до використання в низькоресурсних умовах, були закріплені в рамках міжнародного стандарту ISO / IEC FDIS 29192 - Information technology — Security.

Всі алгоритми шифрування використовують у своїй основі криптографічні примітиви. Криптографічні примітиви (cryptographic primitives) є основним криптографічним інструментарієм, який забезпечує виконання тих чи інших криптографічних сервісів. Зазвичай їх поділяють на примітиви з секретним ключем (симетричні примітиви), примітиви з відкритим ключем (асиметричні примітиви) і безключеві примітиви.

Примітиви з секретним ключем. Для таких примітивів криптографічний стійкість забезпечується таємністю ключа, який повинен залишатися невідомим для всіх учасників інформаційного процесу, які не мають відповідних повноважень. У цей клас входять такі примітиви, як симетричні шифри (Які, в свою чергу, поділяються на блокові шифри і потокові шифри), ключові хеш-функції, звані також кодами перевірки справжності повідомлення або імітовставка (Keyed hash function, Message Authentication Code - MAC) а також криптографічні генератори псевдовипадкових послідовностей.

Примітиви з відкритим ключем. Примітиви з цього класу використовують

пари ключів - (ключ шифрування, ключ розшифрування) або (Відкритий ключ, секретний ключ). Криптографічний стійкість забезпечується таємністю секретного ключа, який повинен залишатися невідомим для всіх учасників інформаційного процесу, які не мають відповідних повноважень. У цей клас входять такі примітиви, як протоколи вироблення та узгодження ключів, асиметричні шифри (шифри з відкритим ключем), схеми цифрового підпису та деякі інші.

Безключеві примітиви. У цей клас входять примітиви, які не використовують ключів. Такі примітиви використовуються в таких криптографічних сервісах, як аутентифікація і гарантування цілісності інформації. У цей клас входять такі примітиви, як односпрямовані підстановки (One-way permutation), хеш-функції, безключеві хеш-функції або коди виявлення модифікації інформації (unkeyed hash function, Modification Detection Code - MDC), генератори випадкових послідовностей та ін.

Далі будуть розглянуті результати «легковагової» реалізації основних криптографічних примітивів. Основний упор буде робитися на питання «легковаговості». Безумовно найважливішою характеристикою криптоалгоритму є його криптостійкість. Однак аналіз стійкості і «легких» варіантів класичних алгоритмів і спеціально розроблених легких алгоритмів принципово нічим не відрізняється від аналізу криптоалгоритмів загального вигляду. Він не має якоїсь особливої специфіки і проводиться звичайним для сучасного криптоаналізу чином - здійснюється перевірка стійкості алгоритму щодо відомих на сьогоднішній день методів криптоаналізу (лінійний криптоаналіз, кореляційний аналіз і т.д.). Відзначимо, що для засобів низькоресурсної криптографії дуже важлива їх стійкість по відношенню до аналізу по побічним каналам (Side Channel Attacks), що пояснюється умовами їх експлуатації. Треба сказати, що більшість алгоритмів (і полегшені варіанти класичних і спеціально розроблені «легковагові» алгоритми) демонструють практичну стійкість, тобто показано, що відомі методи криптоаналізу не дозволяють «зламати» алгоритм за час менше, ніж злом грубою силою, що

вимагає перебору всього ключового простору алгоритму. Рідкісним винятком є, наприклад, алгоритм ГОСТ 28147-89, для якого ряд дослідників отримали результати, що дозволяють зламувати його швидше повного перебору. Проте, алгоритм демонструє практичну стійкість - на його злом потрібен час порядку 2191 операцій шифрування при наявності 264 пар відкритий текст-шифротекст, що робить запропоновану атаку нездійсненною на практиці.

Всеосяжне порівняння різних реалізацій залежить від дуже багатьох параметрів, включаючи технологію, архітектуру і т.д. Отримання «абсолютно кращих» реалізацій навряд чи можливо хоча б в силу відсутності методів отримання нижніх оцінок схемової складності навіть для найпростіших перетворень. Однак, за оцінками деяких авторів, для ряду схем досягнута межа в області мінімізації по площі. Симетричні і асиметричні криптоалгоритми будуть розглянуті окремо, з огляду на те, що вони мають різні сфери застосування. Симетричні алгоритми, в основному, служать для шифрування, перевірки цілісності повідомлень, аутентифікації, в той час як асиметричні алгоритми використовуються в основному для управління ключами і забезпечення неспростовності. Асиметричні алгоритми вимагають значно більшого обсягу обчислень в порівнянні з симетричними як при апаратній так і при програмної реалізації. Розрив в продуктивності на пристроях з обмеженими ресурсами (наприклад, для 8-розрядних мікроконтролерів) величезний. Так оптимізований асиметричний алгоритм на еліптичних кривих (ECC) виконується від 100 до 1000 раз повільніше, ніж стандартний симетричний шифр і має на два-три порядки вище енергоспоживання. Блокові шифри найбільш активна і найбільш продуктивна діяльність з розробки низькоресурсних криптоалгоритмів відбувалася в області алгоритмів блочного шифрування. За останні 10 років було запропоновано безліч низькоресурсних рішень. При цьому розвиток цієї галузі низькоресурсної криптографії йшло за двома напрямками:

– ефективна реалізації відомих алгоритмів блочного шифрування (з, можливо, їх невеликою модифікацією в сторону «полегшення», але за умови збереження або незначного зниження їх кріптографічних

властивостей).

– розробка нових блокових шифрів, орієнтованих на оптимальну реалізацію на мікропрограмному або апаратному рівні. Деякі дослідники вважають, що вже є досить великий вибір «полегшених»

блокових криптоалгоритмів, придатних для практичного застосування. Зокрема, в міжнародний стандарт ISO / IEC 29192-2 (Block ciphers) включені два алгоритма:

– блоковий шифр PRESENT (розмір інформаційного блоку - 64-біт, розмір ключа — 80 або 128 біт);
– блоковий шифр CLEFIA (розмір інформаційного блоку - 128-біт, розмір ключа — 128, 192 або 256 біт). Як «точки відліку» для порівняння тих чи інших реалізацій блокових шифрів наведемо різні реалізації «еталонного» алгоритму блочного шифру AES. Реалізації AES. Апаратна реалізація. Найбільш швидкісна реалізація алгоритму AES демонструє швидкість до 70 Гбіт/с. Така реалізація використовує конвеєрну архітектуру процесора і вимагає більш ніж 250,000 GE. У той же час найбільш компактна реалізація цього алгоритму вимагає порядку 2,400 GE.

Програмна реалізація для стандартних процесорів алгоритму AES, забезпечує швидкість 7.6 тактів на байт на процесорі Intel Core 2 Q9550 або 6.9 тактів на байт на процесорі Intel Core i7. Програмно-апаратна реалізація. Прийняття стандарту AES викликало розробку додаткових команд для процесорів сімейства Intel Подібне розширення PadLock engine існує в мікропроцесорах від VIA Technologies. Метою даного розширення є прискорення додатків, що використовують шифрування за алгоритмом AES що забезпечує швидкість шифрування порядку 0.75 тактів на байт.

Таким чином, основними характеристиками реалізації криптографічного алгоритму є складність реалізації і швидкість роботи. Швидкість – дуже важлива характеристика в багатьох (але не всіх) застосуваннях - в свою чергу залежить не тільки від частоти роботи процесора, але і від розмірів мікросхеми (в разі

апаратної реалізації) оскільки криптографічні примітиви, як правило, дуже зручні для розпаралелювання. У свою чергу складність характеризується розміром мікросхеми в GE9 (в разі апаратної реалізації) або розміром програмного коду в байтах і розміром необхідної оперативної пам'яті (в разі програмної реалізації).

2.2 Порівння блочних «легковагових» шифрів

Блоковий шифр - різновид симетричного шифру, який за допомогою незмінного відображення (як правило) обробляє блоки інформації (найчастіше 64 або 128 біт). «Полегшений» блоковий шифр відрізняється, від блокового тим, що в ньому використовуються алгоритми, які вимагають менших обчислювальних потужностей. Нижче наведено список найпоширеніших на сьогоднішній день блокових «легковагових» криптографічних алгоритмів.

Далі розглянемо такі блокові шифри, як:

PRESENT є блоковим шифром і класичною SP-мережею (Substitution permutation network) з 64-бітними інформаційними блоками, 80-бітовим або 128-бітовим ключем і складається з 31 + 1 раунду шифрування. Кожен раунд виконується операція XOR з раундовим ключем K. Ключ має розрядність 64 біта і визначається функцією оновлення ключа. Після цього проводиться такт розсіювання перетворення, тобто блок пропускається через 16 однакових S-блоків, що мають розрядність 4, складених таким чином, щоб максимально підвищити стійкість алгоритму до лінійного і диференціального криптоаналізу. Потім в блоці переставляються біти. Трудомісткість алгоритму невелика. Для функції add_round_key () слід дотримуватися операції bj → bj XOR kj де j варьюється від 0 до 63. У s_box_layer виконються перетворення з 4-х біт в 4 біта. p_layer виконує переміщення біта і на позицію P (i). Ключ задається користувачем в особливому регістрі ключа K тобто містить 64 біта лівой частини значущих біт. Таким чином в раунді i, регістр ключа оновлюється по ходу алгоритму наступним чином, ключ зсувається на 61 позицію вліво, 4 біта лівої значущої частини

проходить через S-блок, а над молодшими правими значеннями циклічного лічільника і та бітами регістра ключа K виконується операція XOR. Даний алгоритм добре захищений від атак на основі пов’язаних ключів, слайд-атак і інших поширених методів атак на криптосистеми. На рисунку 2.1 показано схему данного алгоритму.

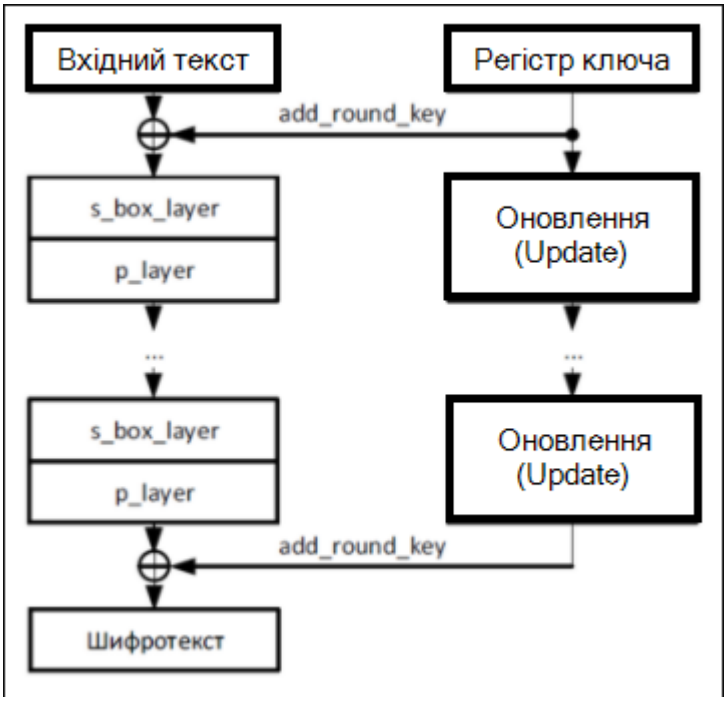

Рисунок 2.1 – Алгоритм PRESENT[9]

Наступний стандарт блочного шифрування ГОСТ 28147-89, відомий вже понад двадцять років і при відповідному виборі вузлів заміни вважався стійким алгоритмом. Простота конструкції і низькі апаратні вимоги до реалізації дозволяють досить ефективно використовувати алгоритм на платформах з обмеженими ресурсами. На рисунку 2.2 показано схему 1го раунду даного алгоритму.

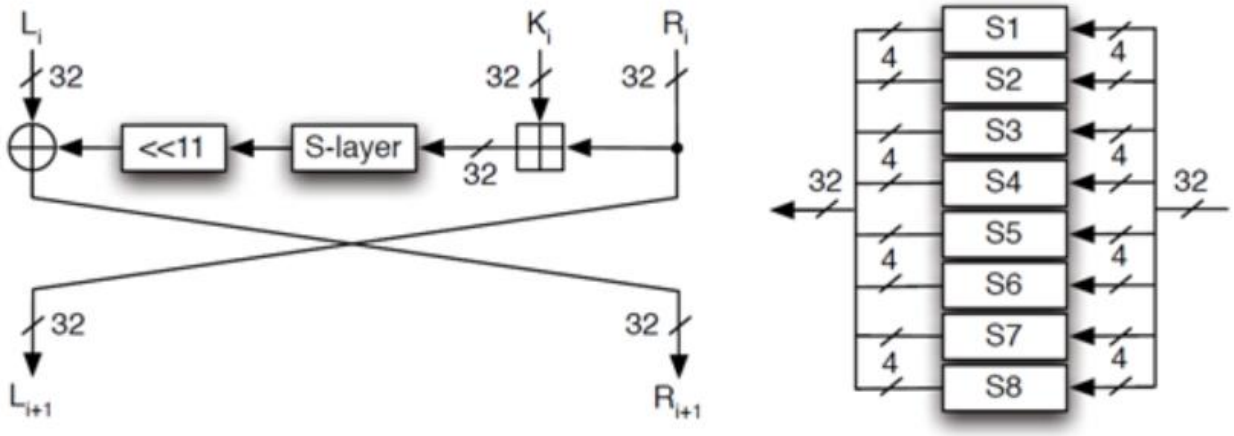

Рисунок 2.2 - Раунд ГОСТ 28147-89 (зліва) і докладна схема S-блоку[10]

ГОСТ має розмір блоку 64 біта і розмір ключа 256 біт. Його загальна структура - дві мережі Фейстеля з 32 раундами і внутрішньою функцією roundfunction F, що складається з складання, нелінійної заміни і 11-бітового циклічного зсуву вліво. Позначимо 64-бітове стан даних ГОСТ як STATEi = Li || Ri, де || позначa з'єднання, тоді $STATE_0$ являється відкритим текстом а $STATE_{32}$ – зашифрованим. Права половина Ri складається по модулю 32 з roundKey Mrd Ki, результат розбивається на вісім 4-бітових підпослідовностей, кожна з яких надходить на вхід свого вузла таблиці замін (в порядку зростання старшинства бітів), званого нижче S-блоком і, нарешті, циклічно зсувається на 11 біт вліво. Результат roundfunction F (Ki, Ri) складається з лівої половиною Li і зберігається в правій половині як результат Ri+1, в той час як Li+1 зберігається без будь-яких змін. Виходить $L_{i+1} = R_i$

$$R_{i+1} = L_i \oplus (S(K_i + R_i mod\ 2^{32}) \ll 11) \quad (2.1)$$

де $\oplus$ позначає побітове виключне АБО та $\ll$ a - циклічний зсув на a наліво. У заключному колі, половинки не змінюються, тобто

$$R_{32} = R_{31} \text{ та } L_{32} = L_{31} \oplus\oplus (S(K_{31} + R_{31}\ mod\ 2^{32}) \ll 11) \quad (2.2)$$

Це дає можливість використовувати той же самий алгоритми в зворотному порядку для розшифровки. В роботі описана реалізація даного шифру, придатна для легковагій реалізації. У криптографічному алгоритмі ГОСТ 28147-89 - використовується відображення

$$F: V_{64} * V_{256} \to V_{64}: F(P, K) = C \tag{2.3}$$

де P – відкритий текст, K – ключ,

$$K = (K_0, \dots, K_7) \in V_{256}, K_i \in V_{32}, i \in \overline{0,7} \tag{2.4}$$

C-шифрований текст. При заданому фіксованому ключі

K відображення

F здійснює біективне перетворення множини V^{64} і є добутком 32-х відображень:

$$F(X) = F_1 * F_2 * \dots * F_{32} \tag{2.5}$$

Тут і далі в множеннях перетворення застосовуються до аргументу послідовно зліва-направо, тобто

$$F(X) = F_1 * F_2 * \dots * F_{32}\ P(X) = F_{32}\big(\dots F_2\big(F_1(P)\big)\dots\big) \tag{2.6}$$

Відображення $F_i, i \in \overline{1,32}$ (i-та ітерація) залежить від ітераційного ключа M_i і визначається наступним чином(2.7): Нехай відображення

$$F^*(X, M) = F^*\left(\left(x^{(2)}, x^{(1)}\right), M\right) = \left(y^{(2)}, y^{(1)}\right) = Y,\ F^*: V_{64} * V_{32} \to V_{64} \tag{2.7}$$

задається наступними формулами:

$$y^{(1)} = x^{(2)}, y^{(2)} = L(\Pi(x^{(2)}[+]M)) \oplus x^{(1)} \quad (2.8)$$

Де [+] - додавання по модулю 2^{32}.

L- лінійне перетворення простору V_{32}, що представляє собою циклічний зсув вліво (в сторону старших координат) на 11 позицій

П - нелінійне перетворення простору V_{32}, реалізоване за допомогою восьми паралельно діючих підстановок на довічних векторах довжини 4:

$$\Pi = (\Pi_0, \dots, \Pi_7), \Pi_i: V_4 \to V_4, i = \overline{0,7} \quad (2.9)$$

Наступний алгоритм шифрування CLEFIA розроблений компанією Sony в якості безпечної альтернативи AES для сфери захисту авторських прав і DRM-систем (Digital rights management - програмні або програмно-апаратні засоби, які навмисно обмежують або ускладнюють різні дії з даними в електронній формі, наприклад захист творів від копіювання та інших дій, що забороняються авторами або іншими правовласниками на підставі авторського або суміжних прав після продажу кінцевому користувачу). Алгоритм складається з двох складових елементів: частини ключового розкладу і частини обробки даних. Алгоритм відповідає вимогам до шифру AES: Розмір блоку - 128 біт (16 байт), підтримувана довжина ключа - 128, 192 і 256 біт. Число раундів залежить від довжини ключа і становить відповідно 18, 22 або 26 раундів і 36, 44 і 52 використовуваних раундовий ключів. Схема шифрування алгоритму Clefia показана на рисунку 2.3 і 2.4, параметри і атаки на алгоритм наведені в таблиці 2.1.

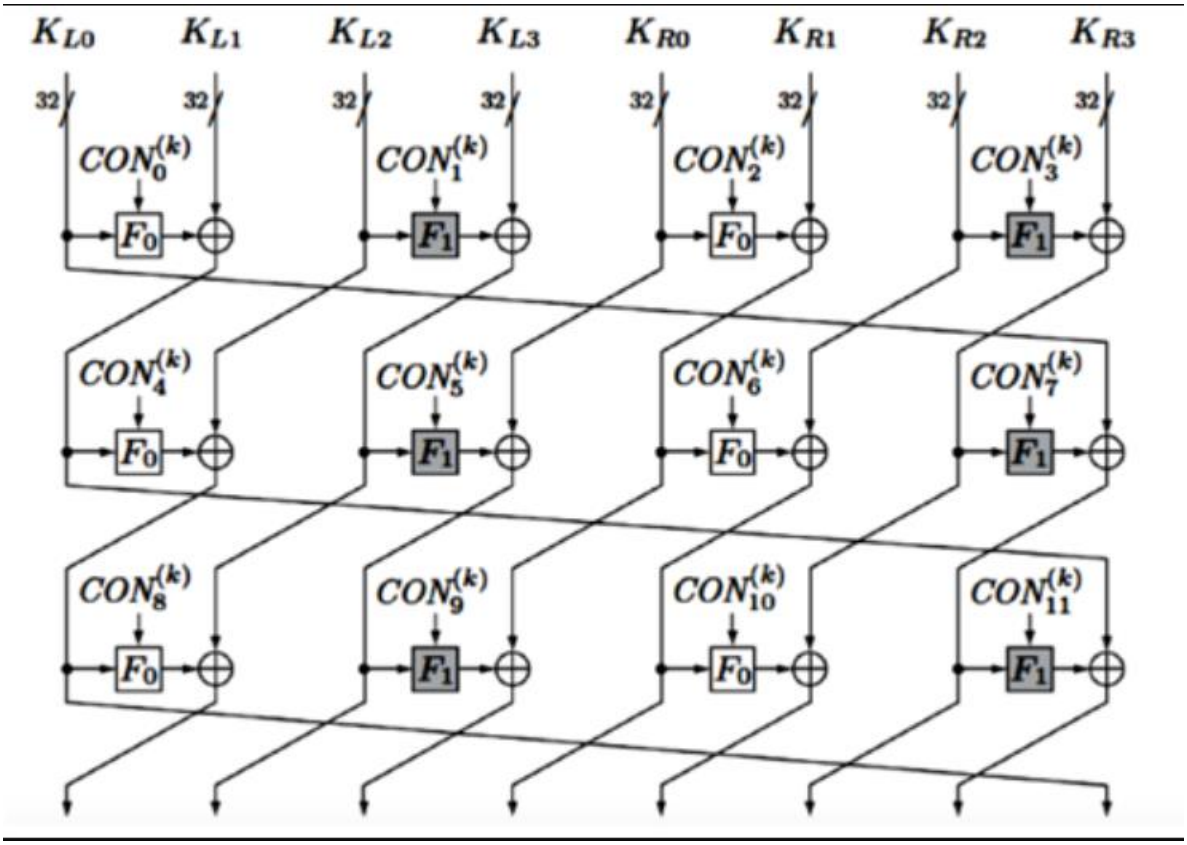

Рисунок 2.3 – Схема шифрування алгоритму Clefia [11]

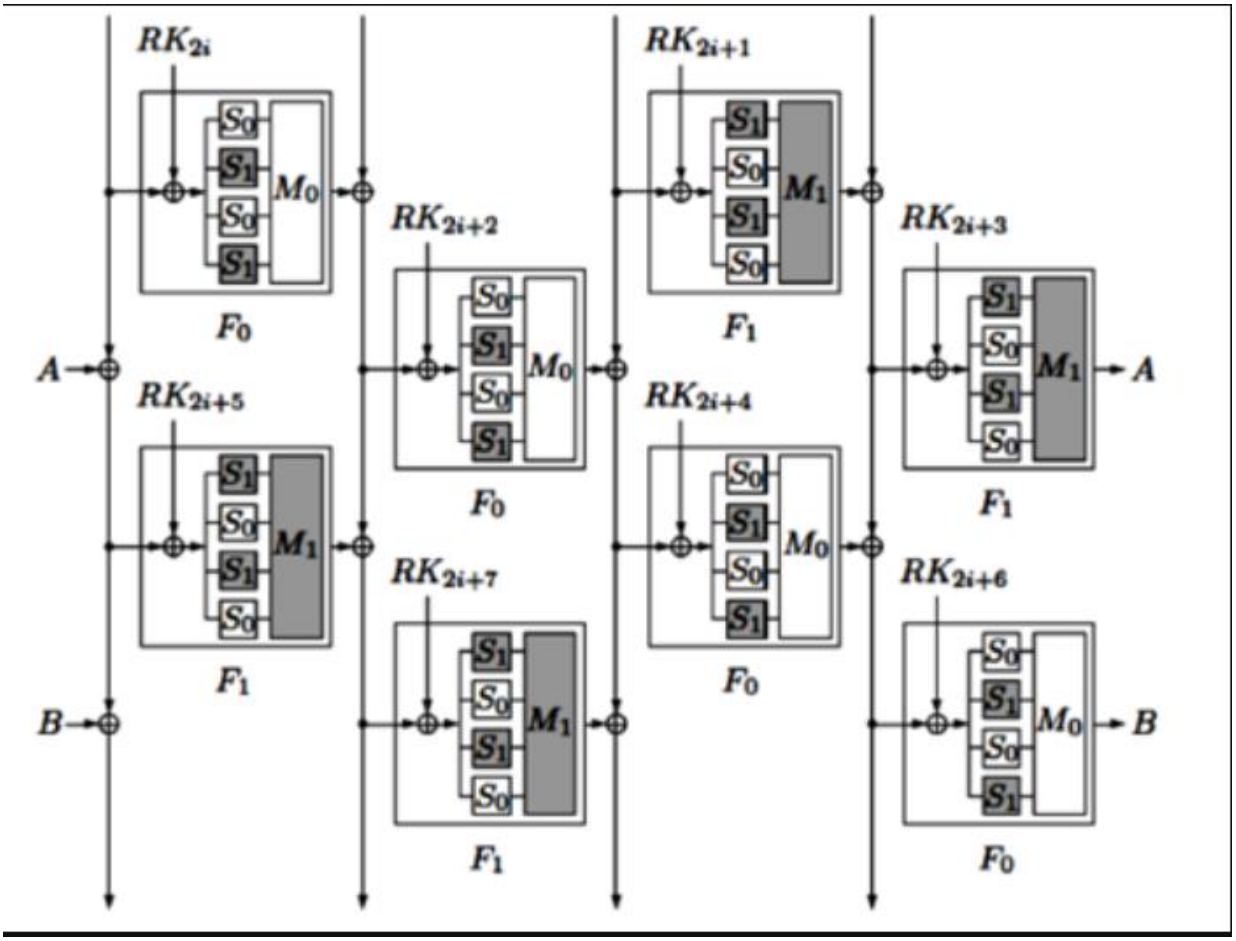

Рисунок 2.4 – Схема шифрування раундового ключа алгоритму Clefia [11]

Таблиця 2.1 – Параметри алгоритма Clefia на ключах різної довжини [9]

Ключ	Тактів на блок	Пропускна швидкість при 100Kbps	Кількість	Ефективність	Струм
128	36	355,6	4950	71,83	N/A
128	18	711,11	5979	118,93	N/A
192	22	581,8	8536	68,16	N/A
256	26	492,3	8482	58,04	N/A

Алгоритм використовує ключове «вибілювання» з додатковим підключами перед обробкою даних і після неї. Заявленими вимогами до алгоритму були:

– Безпека;

– Швидкість виконання;

– Простота реалізації на недорогому обладнанні.

Алгоритм став першим шифром, який застосовує технологію DSM (Diffusion Switching Mechanism) для збільшення захищеності від лінійного і диференціального криптоаналізу. Розробники також врахували існування нового типу атак - алгебраїчних і заявляють про стійкість до них шифру CLEFIA. Для захисту від ряду атак застосовуються також два типи таблиць підстановки.

Наступним розглянемо сімейство алгоритмів шифрування KATAN KATAN32, KATAN48, KATAN64. Число в назві алгоритму позначає розмір блоку шифрованих даних в бітах. Всі алгоритми використовують 80-бітовий ключ шифрування. Будь-який з алгоритмів KATAN завантажує шифруємий блок даних в два регістра зсуву, що утворюють внутрішній стан алгоритму (тобто якесь проміжне значення, залежне від блоку даних і ключа шифрування; саме воно і обробляється алгоритмом в процесі шифрування). Шифрування складається з 254 раундів, в кожному з яких використовуються нелінійні функції, що формують зворотний зв’язок регістрів:

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_2] * L_1[x_4]) \oplus (L_1[x_5] * IR) \oplus k_a \quad (2.10)$$

$$f_a(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_2] * L_2[y_4]) \oplus (L_2[y_5] * L_2[y_6]) \oplus k_b \quad (2.11)$$

Ключовою особливістю даного алгоритму є можливість його реалізації з використанням 802 GE в разі KATAN32, що дозволяє його використовувати в системах з обмеженими обчислювальними ресурсами. Структура одного раунду показана на рисунку 2.4, де L_1 і L_2- регістри зсуву.

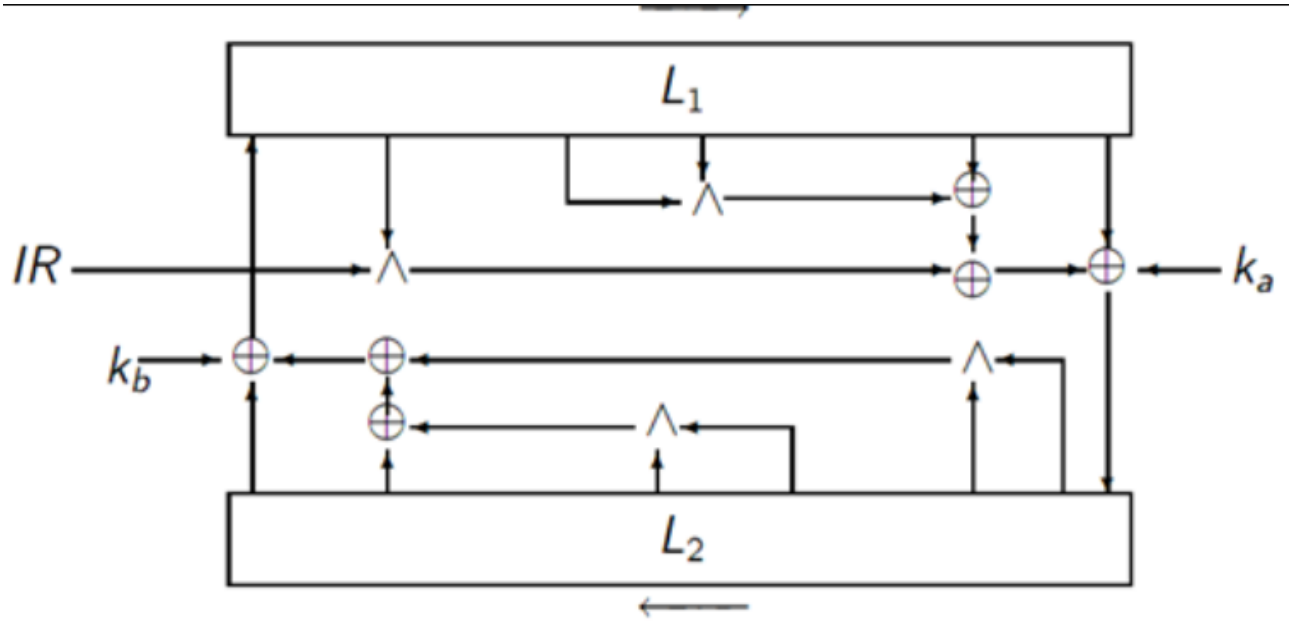

Рисунок 2.4 – Структура одного раунду алгоритму [12]

Після 254 раундів шифрування вміст регістрів є шифротекстом. Причому останній біт регістра L_2 є останнім бітом шифротекста. При розшифруванні шифротекст поміщається в регістри зсуву, причому останній біт відкритого тексту поміщається в останній біт регістра L_2 Так як шифр симетричний, то алгоритм розшифрування ідентичний алгоритму шифрування на тому ж ключі.

2.3 Порівняння потокових «легковагових» шифрів

Потоковий шифр - це симетричний шифр, в якому кожен символ відкритого тексту перетворюється в символ шифрованого тексту в залежності не тільки від

використовуваного ключа, а й від його розташування в потоці відкритого тексту. Поточний шифр реалізує інший підхід до симетричного шифрування, ніж блокові шифри. Нижче наведено список розглянутих «легковагових» криптографічних алгоритмів, заснованих на поточному шифруванні та основні механізми реалізації. Але перш ніж ми перейдемо до їх опису, необхідно ознайомитися з основними реалізаціями, які використовуються в апаратної частини таких алгоритмів: LFSR і FCSR.

LFSR - регістр зсуву бітових слів, у якого значення вхідного (всувають) біта одно лінійної булевої функції від значень інших бітів регістра до зсуву. Може бути організований як програмними, так і апаратними засобами. Застосовується для генерації псевдовипадкових послідовностей бітів, що знаходить застосування, зокрема, в криптографії. На рисунку 2.5 показано принцип роботи

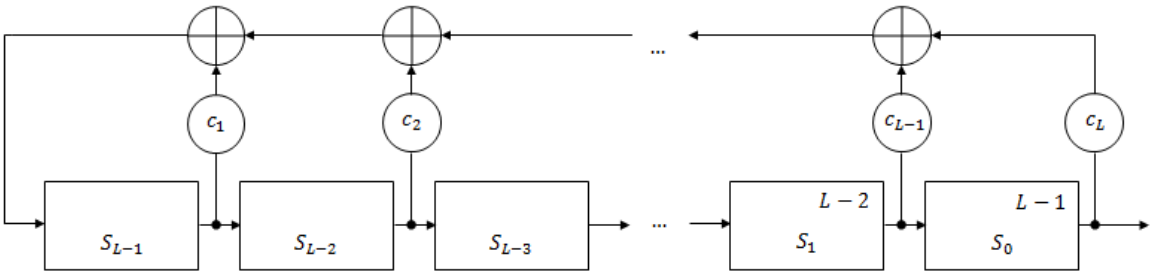

Рисунок 2.5 – Регістр зсуву з лінійним зворотнім зв’язком

Протягом кожного такту регістр зсуву з лінійним зворотним зв’язком виконує наступні операції:

- зчитується біт, розташований в осередку L - 1; цей біт є черговим бітом вихідний послідовності;
- функції зворотного зв’язку обчислює нове значення для осередку 0, використовуючи поточні значення осередків;
- вміст кожної i-й осередку переміщається в наступну комірку i + 1, де i = 0; 1; ...; L-2;
- в клітинку 0 записується біт, раніше обчислений функцією зворотного зв’язку.

Якщо функція зворотного зв’язку виконує логічну операцію «XOR» (виключне АБО), значення біт (осередків) можуть бути обчислені за наступними формулами:

$$S_L = (c_1 * S_{L-1}) \oplus (c_2 * S_{L-2}) \oplus \dots \oplus (c_L * S_0) \quad (2.12)$$

$$S_{L+1} = (c_1 * S_L) \oplus (c_2 * S_{L-1}) \oplus \dots \oplus (c_L * S_1) \quad (2.13)$$

$$S_{L+j-1} = (c_1 * S_{L+j-2}) \oplus (c_2 * S_{L+j-3}) \oplus \dots \oplus (c_L * S_{j-1}) \quad (2.14)$$

FCSR використовує ідею використання регістра зсуву зі зворотним зв’язком по перенесенню для створення поточного фільтра. Ця ідея була запропонована Клаппером і Гореські в 1994 році. Пізніше ними був розроблений алгоритм такого шифру. Один FCSR без підключення лінійного компонента не може бути використаний в якості поточного шифру, так як легко дешифрується. У 2002 році був запропонований самосинхронізуючийся потоковий шифр, заснований на спільному використанні FCSR і LFSR. Пізніше він був підданий атаці з вибором шифротекста. У 2005 році Арно і Бергер запропонували ідею спільного використання FCSR і лінійного фільтра для створення поточного шифру, який отримав назву F-FCSR (Filtered FCSR). Пізніше ними були запропоновані 4 алгоритму, що реалізують цю ідею: F-FCSR-SF1, F-FCSR-SF, F-FCSR-DF1 і F-FCSR-DF8. Перші два використовували статичні фільтри, останні - фільтри, які залежать від ключа. Пізніше була виявлена слабкість всіх цих алгоритмів перед різними видами атак. У 2005 Террі Бергер, Франсуа Арноль і Седрік Лараду запропонували два шифру на основі F-FCSR для участі в конкурсі eSTREAM: F-FCSR-H для апаратної реалізації і F-FCSR-8 для програмної. В результаті подальших випробувань у початкових версій F-FCSR-H і F-FCSR-8 були знайдені вразливості, які пізніше були виправлені в версіях F-FCSR-H v.2 і F-FCSR-16. Покращений варіант F-FCSR-H v.2 став фіналістом eSTREAM. Але після виявлення уразливості був виключений з eSTREAM Portfolio. На рисунку 2.6 показаний принцип роботи алгоритму

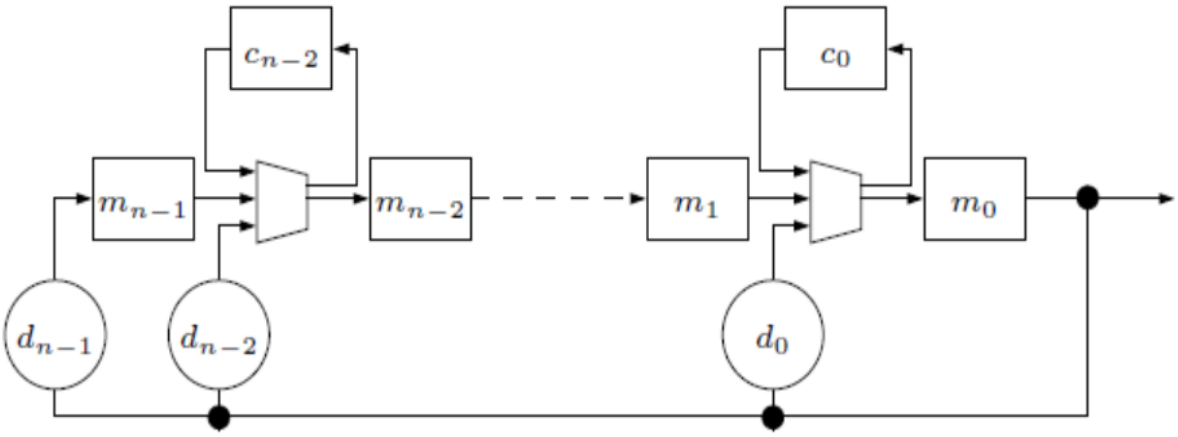

Рисунок 2.6 – Конфігурація Галуа для FCSR

FCSR реалізується в двох конфігураціях: Галуа і Фіббоначі. У F-FCSR використовується конфігурація Галуа, так як вона ефективніше. Вводяться такі позначення:

q - цілісність з'єднання (connection integer) - негативне непарне ціле число, яке задовольняє наступним умовам:

$$2^{|q|-1} = 1 \; mod \; q \tag{2.13}$$

- просте, 2T - період бітової послідовності p / q

$$T = (| q | - 1) / 2 \tag{2.14}$$

p - параметр, що залежить від ключа, так, що 0 <p <| q |

n - розмір головного регістра FCSR, | q | в двійковому запису має n + 1 знаків: 2n

d = (1 - q) / 2, в двійковому запису

Число одиниць в двійковому представленні числа (1 - q) / 2 порядку n / 2

$$<-q <2n + 1 \tag{2.15}$$

$$\sum_{i=0}^{n-1} d_i 2^i, \text{ di} = \{0, 1\}, \text{ dn-1} = 1 \tag{2.16}$$

M - n-розрядний головний регістр, значення його i-го розряду

$$m(t) = \sum_{i=0}^{l-1} m_i(t)\, 2^{|q|-1} \quad (2.17)$$

C - l-розрядний регістр зсуву, l + 1 - число одиниць в двійковому запису d,

$$c(t) = \sum_{i=0}^{l-1} c_i(t) 2^i \quad (2.18)$$

(m, c) - стан FCSR. Якщо (m, c) - стан FCSR в момент часу t0,

$$m = m(t_0)\; c = c(t_0), \text{то } (m_0(t_0 + i))_{i \in N} \quad (2.19)$$

двійкове подання p / q, де p = m + 2c.

Наступними розглянемо сімейство поточних шифрів Grain. Головною відмінністю від інших алгоритмів є надзвичайно мала апаратна реалізація. Початкова версія v0 була доопрацьована після деяких спостережень Berbain. Остаточний варіант Grain відомий як Grain v1. Як і інші шифри, Grain v1 досить сучасний для того щоб використовувати публічні, тобто загальнодоступні вектори ініціалізації (IV), хоча і 80-бітові. Визнаючи необхідність в 128-бітних ключах, Hell запропонував 128-бітові ключі для Grain, в якому використовуються 96-бітові вектори ініціалізації. Принцип той же, що і у 80-бітової версії, але особливістю є нелінійна частина шифру, а саме менший ступінь ніж в v0. Загальна ідея Grain полягає у використанні ключа розміру | k |; NFSR і LFSR аналогічно с розміром | k | і вихідний функції, яка використовує стан обох регістрів. LFSR забезпечує функціонування NFSR шляхом передачі даних. У NFSR надходить ключ, а LFSR - вектор ініціалізації, що доповнюється константою. Після цього ключ і вектор ініціалізації завантажуються, але перед генерацією гами стан визначається шляхом 2 · | k | оновлень, де відбувається зворотний зв’язок вихідний послідовності в LSRF. Після ініціалізації

починається генерування гами. Реалізації будь-якого генератора гами, відповідного вище описаного, можна зробити швидше шляхом додавання стану поновлення і вихідний функції кілька разів, причому паралельно. Однією особливістю шифрів Grain є те, що знову вироблені біти в регістрах не використовується протягом декількох тактів. Це означає, що немає необхідності у використанні рекурсії для реалізації паралельних функцій. Поточний шифр Grain був розроблений Мартіном Хеллом, Томасом Йоханссоном і Віллі Мейером. Їх основною метою було розробити алгоритм, який дуже легко реалізувати в апаратної частини і який досить малий по площі. Це біт-орієнтований синхронний потоковий шифр, який означає, що ключ генерується незалежно від вхідного тексту. Загалом, потоковий шифр складається з двох етапів. Перший етап - це етап ініціалізації внутрішнього стану за допомогою секретного ключа і вектора ініціалізації. Після цього стан повторно оновлюється і використовується для генерації гами. Основні елементи поточного шифру Grain, як було згадано вище, - це два 80-бітових регістрів зсуву, де один має лінійну зворотний зв'язок (LFSR), а інший нелінійну зворотний зв'язок (NFSR). Розмір ключа становить 80-біт, в утворенні якого використовується 64-біта вектора ініціалізації. На жаль, первісна версія Grain v0 мала вразливість у функції виведення, яку виявили під час першого етапу оцінки. У Grain v1 проблема з ветором ініціалізації була вирішена. Два багаточлена ступеня 80 f (x) і g (x) використовуються в якості опції зворотного зв'язку для регістрів зсуву зі зворотним зв'язком LFSR і NFSR. Функція виходу h (x) використовує в якості вхідної послідовності біти з обох регістрів зсуву зі зворотним зв'язком. Крім того, сім біт з NFSR XOR-яться і результат додається до функції h (x). Цей результат використовується в фазі ініціалізації в якості додаткового зворотного зв'язку LFSR і NFSR (сірі ліній на малюнку позначають цю зворотний зв'язок). Під час нормальної роботи це значення використовується в якості гами. Основну структуру алгоритму можна побачити на рисунку 2.7.

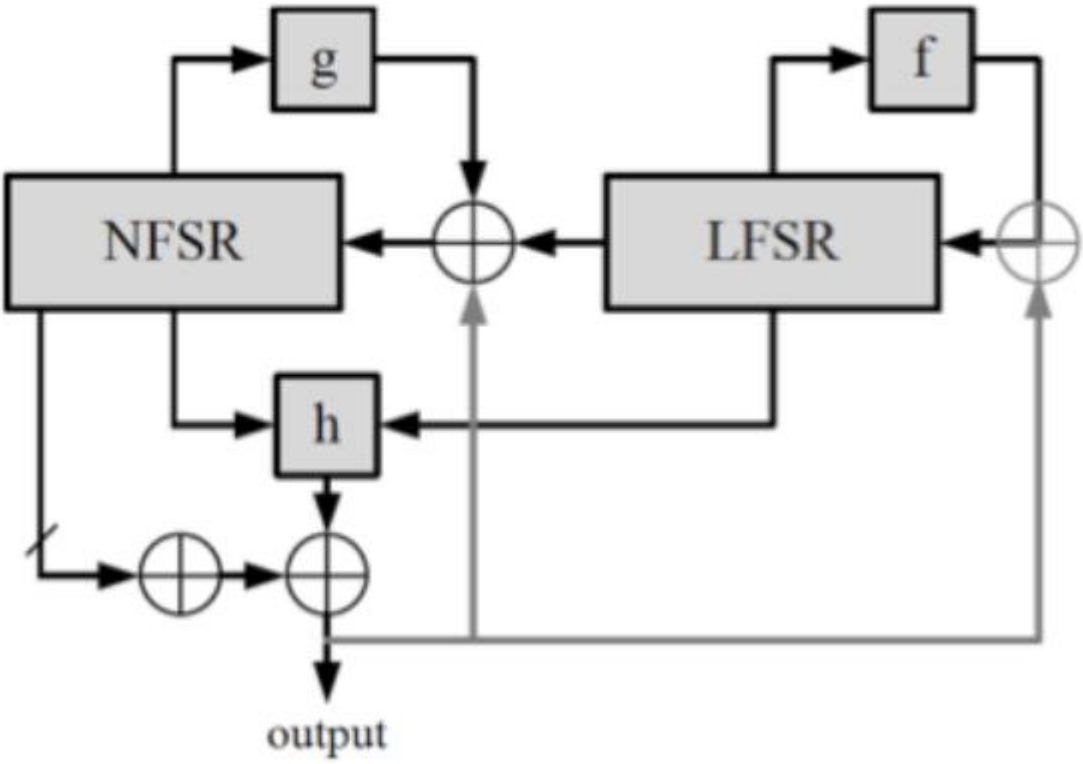

Рисунок 2.7 – Схема потокового шифру Grain v1

Апаратний модуль Grain v1 був реалізований з 16-бітовим AMBA APB інтерфейсом і технологічним процесом в 0,35 мкм на CMOS. Цей інтерфейс підходить до 16-бітної архітектури. Причиною для цієї реалізації на 16-бітах слова є підхід з низьким енергоспоживанням. Подробиці ітерацій проходу даних представлені на рисунку 2.8.

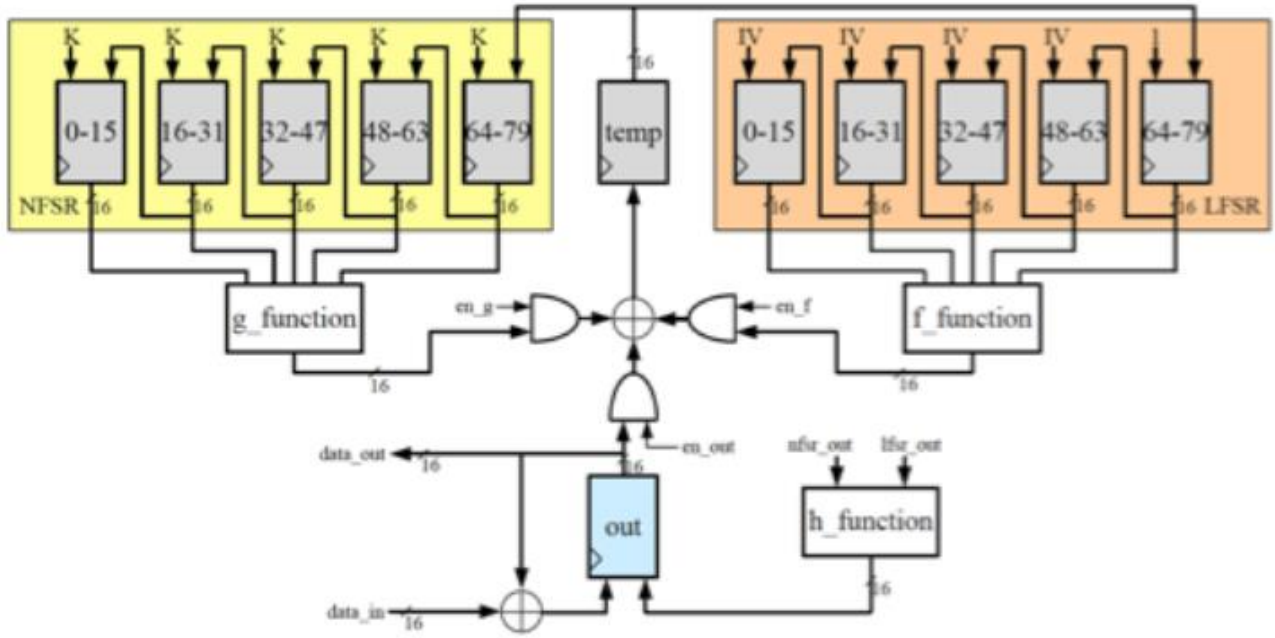

Рисунок 2.8 – Схема обробки даних поточного шифру Grain v1

Нижче наведено таблицю зі значеннями GE для основних елементів реалізації Grain v1

Таблиця 2.2 – Компоненти реалізації Grain v1

Компонент	Мінімальна площа (radix -1), GE	Мінімальна площа (radix -16), GE
NFSR + LFSR (160біт)	1275	1130
Тимчасовий регістр	0	85
Регістр виходу	50	150
Логіка з misc	315	1835
Контролер (FSM)	120	160
Сумарно	1760	3360

Видно, що регістри зсуву зворотного зв’язку NFSR і LFSR обробляють 16 біт за такт. Тільки один регістр діє в той же момент часу, що полегшує введення ключа і вектор ініціалізації, тому що ті ж самі 16 проводів підключені до всіх регістрах. Крім того, значно зменшується середнє споживання енергії. Це

відбувається за рахунок наявності тимчасового регістра, який зберігає проміжні результати. Крім того всі комбінаційні схеми, як функції зворотного зв'язку g, f і h повинні бути реалізовані в шістнадцятиричній системі числення. Входи цих функції - це вибрані біти з регістрів, які не показані детально на цьому малюнку. h функція також включає в себе XOR операції вихідний функції. Вихід модуля вже зареєстрований і замість гами і зашифрованого результату вхідних даних зберігається в регістрі. Замість того, щоб мультиплексор сам вибирав коректну зворотну функцію для тимчасового регістра, команда AND використовуються для включення і відключення відповідних входів. Виробництво 16-бітного результату шифрування вимагає 13 тактів після ініціалізації.

Наступним розглянемо Trivium. Розробниками поточного шифру Trivium є Крістоф де Каніер і Барт Пренель. Це апаратно-орієнтований синхронний потоковий шифр, який був розроблений для дослідження того, наскільки простим повинен бути шифр, щоб не жертвувати його безпекою. Можна генерувати до 2^{64} бітів гами з 80-бітного ключа і 80 бітів вектора ініціалізації. Стан складається з 288 біт які позначені як S0, S2, ..., s287. Протягом ініціалізації, яка не відображено на схемі, ключ і вектор не завантажуються в стан і в той же час оновлення фнкцій відбувається тисячу сто п'ятьдесят два раза без використання виходу для гами. В описі алгоритму, N використовується для визначення кількості вихідних бітів поточного шифру. Реалізація модуля Trivium має той же 16-бітний AMBA APB інтерфейс, як і Grain v1. Використання 16-бітного процесора також мотивовано енергозбереженням, яке є необхідним критерієм для «полегшених» шифрів. Рисунок 2.9 показує детальну інформацію про архітектуру Trivium.

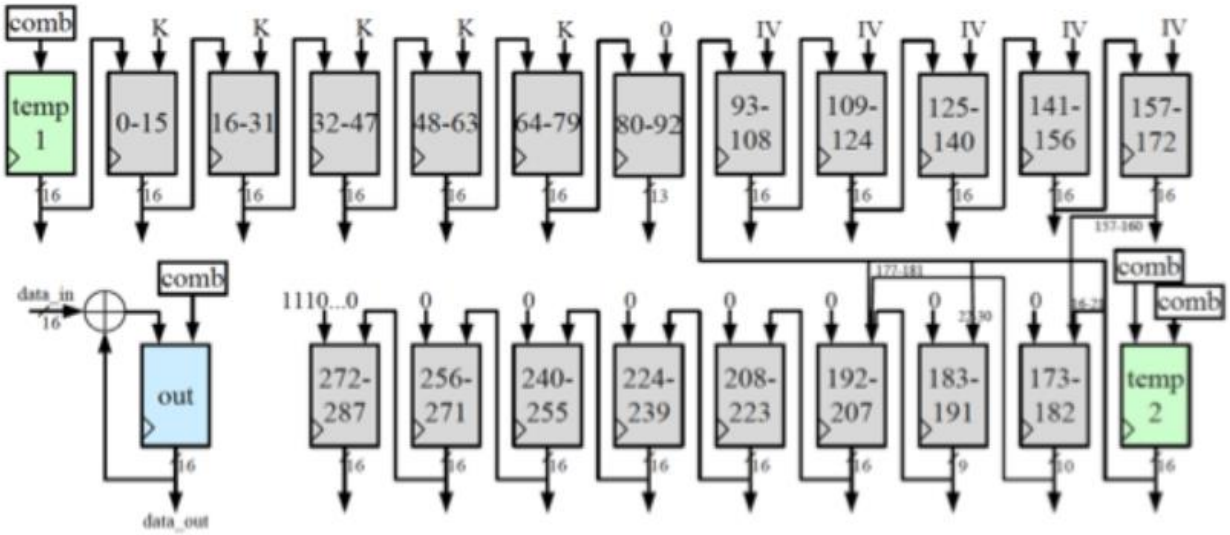

Рисунок 2.9 – Схема обробки даних поточного шифру Trivium

«Коробки» з вхідним полем comb є комбінаційними логічними елементами алгоритму, які використовуються для оновлення стану відповідно до специфікації. 288 біт для визначення стану поділяються в 16-розрядні регістри. Крім того необхідно два тимчасових регістра для зберігання проміжних результатів. Вихідний регістр знову використовується для безпосереднього застосування операції XOR гами з вхідним значенням. Під час ініціалізації ключ, вектор ініціалізації і константи завантажуються в регістри. Потім комбінаційні схеми використовуються для поновлення регістрів, де також використовуються тимчасові регістри для запобігання перезапису необхідних значень. Генерація 16-бітної гами вимагає 22 такту після фази ініціалізації.

Нижче наведено таблицю зі значеннями GE для основних елементів реалізації Trivium.

Таблиця 2.3 – Компоненти реалізації Trivium

Компонент	Мінімальна площа (radix - 1), GE	Мінімальна площа (radix -16), GE
Регістр стану (288біт)	1840	2040
Тимчасовий регістр	0	200
Регістр виходу	50	150
Логіка з misc	290	410
Контролер (FSM)	210	290
Сумарно	2390	3090

Останнім розглянутим потоковим шифром буде Hummingbird. На відміну від існуючих легких криптографічних примітивів, в яких присутній або блоковий шифр, або потоковий шифр, Hummingbird є поєднанням двох вищезгаданих шифрів з 16-бітовим розміром блоку, 256-бітовим розміром ключа, і 80-бітовим внутрішнім станом. Розмір ключа і внутрішнього стану Hummingbird забезпечує рівень безпеки, адекватний для багатьох додатків RFID. На рисунку 2.10 показана структура верхнього рівня шифрування Hummingbird в процесі шифрування.

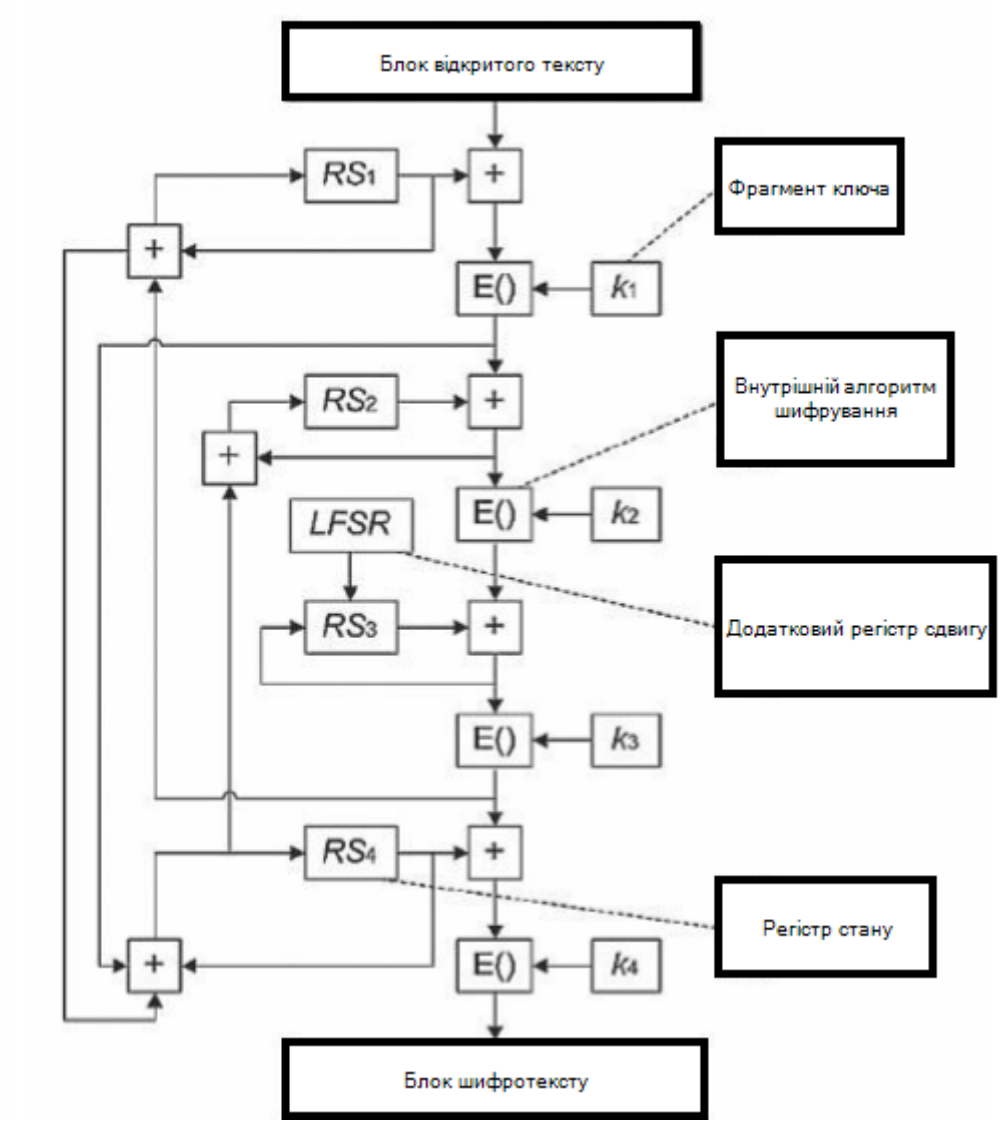

Рисунок 2.10 – Структура верхнього рівня шифрування Hummingbird

Загальна структура алгоритму шифрування Hummingbird складається з чотирьох 16-бітових блокових шифрів Ek1, Ek2, Ek3, Ek4, - чотирьох 16-бітових регістрів внутрішнього стану RS1, RS2, RS3 і RS4, і 16-ступеневої LFSR. 256-бітний секретний ключ K ділиться на чотири 64-бітних підключа k1, k2, k3 і k4, які використовуються в чотирьох блокових шифрів. 16-бітний блок відкритого тексту PTi шифрується по модулю 2^{16} додаванням PTi і містить перший регістр RS1 внутрішнього стану. Результат складання потім шифрується з першим блоковим шифром Ek1. Ця процедура повторюється таким же чином, ще три рази, і вихід Ek4 є зашифрованим текстом CTi. Крім того, стан чотирьох регістрів

внутрішнього стану також будуть оновлені непередбачуваним способом, заснованим на їх поточних станах, виходах перших трьох блокових шифрів, і стані LFSR. Процес розшифровки відбувається за схемою аналогічною шифрування.

На практиці в Hummingbird використовують чотири 16-бітних випадкові NONCEi які в першу чергу вибираються для ініціалізації чотирьох внутрішніх станів регістрів RSi (I = 1, 2, 3, 4), далі йдуть чотири послідовних шифрування повідомлення (RS1 + RS3) mod2 з Hummingbird, які працюють в режимі ініціалізації. Останній 16-бітний зашифрований текст використовується для ініціалізації LFSR. Крім того, 13тий біт LFSR завжди має значення, щоб запобігти нульовий регістр. LFSR також «ступив» з метою оновлення регістра внутрішнього стану RS2.

Чотири однакових 16-бітних блокових шифра використовуються в послідовному порядку в схемі шифрування Hummingbird. 16-бітний блоковий шифр є типовою SP-мережею з 16-бітовим розміром блоку і 64-бітовим ключом. Він складається з 4 ітерацій, і на останній ітерації включає в себе тільки функцію змішання вмісту ключа і кроки заміщення S-боксу. Як і в будь-який інший SP-мережі, одна чергова ітерація складається з трьох етапів: крок змішання ключа, шар заміни і шар перестановки. У цьому 16-бітному блоковому шифрі для змішування ключа використовуються просте виключення або операція, з метою ефективної реалізації в програмної та апаратної частини.

2.4 Висновки до 2 розділу

У другому розділі було розглянуто сучасні реалізації «легковагових» алгоритмів шифрування як блочних так і поточних.

На даний момент три описаних потокових LW-шифру, мають характеристики, що задовольняють сучасні потреби. Це алгоритми Hummingbird, Trivium і GRAIN. Однак дані шифри не застосовні в пасивних RFID-системах в силу індивідуальних особливостей кожного з них. Так, наприклад, Trivium

вимагає площа на чіпі, що перевищує допустиму більш ніж в півтора рази (3488 GE1 при обмеженні у 2000 GE [13]). На поточну версію шифру GRAIN може бути успішно проведена атака на пов'язаних ключах [14]. Що стосується Hummingbird, то розробниками перевірена його стійкість лише до деяких атакам, однак цього недостатньо для забезпечення впевненості в його надійності. Таким чином, можна зробити висновок, що на даний момент серед поточних шифрів немає алгоритму, який задовольняє основним вимогам RFID-систем.

У розділі блокових шифрів ситуація виглядає дещо краще. Розглянемо докладніше деякі блокові LW-алгоритми. LW-алгоритмом, що задовольняє всім вимогам RFID-систем, є PRESENT. Даний шифр використовує ключ довжиною 80 біт, що значно підвищує його надійність. Розробниками проведено дослідження вразливості даного алгоритму до лінійного і диференційного аналізу, алгебраїчної атаки і деяким іншим видам атак. Показана PRESENT стійкість є прекрасним результатом для шифру, створеного «з нуля». На даний момент не відомо жодної успішної атаки на полнораундову версію алгоритму. Існують різні реалізації PRESENT. Найкомпактніша з них вимагає всього 1000 GE, що є одним з кращих результатів для LW-шифрів. Крім забезпечення безпеки даних, що передаються в RFID-системах, деякі модифікації PRESENT знайшли застосування і в інших ресурсозалежний пристроях. Так, наприклад, H-PRESENT-128 є самою компактною з відомих хеш-функцій. Крім того, можливе застосування алгоритму в якості генератора псевдовипадкових чисел для схеми crypto-GPS.

Також серед LW-шифрів можна виділити сімейства алгоритмів KATAN і KTANTAN [15]. Кожне з сімейств складається з трьох шифрів, що відрізняються кількістю раундів шифрування: 32, 48 або 64. Всі шифри мають 80-бітний ключ. Відмінність KTANTAN від KATAN полягає в тому, що перші вимагають меншої кількості ресурсів завдяки тому, що ключ шифрування «вшитий» в пристрій і не може бути змінений. В описі шифрів розробниками показана стійкість до таких атак, як різницевий і лінійний аналізи, атаці на пов'язаних ключах і алгебраїчної атаці. Апаратні реалізації представників KTANTAN показують кращі результати

в даній області криптографії. Так, наприклад, алгоритм KTANTAN48 може бути реалізований з використанням всього 588 GE, що майже вдвічі менше, ніж сама компактна реалізація PRESENT.

Однак, на ряду з усіма перевагами описаних вище блокових шифрів, і для них існують певні загрози, що не дозволяють використовувати їх повсюдно. Якщо використовується відносно короткий ключ, що робить його застосування в пристроях, які повинні забезпечувати безпеку на високому рівні, неможливим. алгоритми PRESENT і KTANTAN, незважаючи на безліч досліджень, проведених за останні кілька років, можуть нести в собі критичні уразливості, які зведуть нанівець всі наявні гідності.

Існує ще безліч блокових LW-алгоритмів. Однак вони мають певні недоліки. Наприклад, MIBS і TWIS показують непогані результати як в плані швидкодії, так і в плані економічності, проте проведено недостатньо їх досліджень, що, як і в випадку з поточними алгоритмами, не дозволяє з належною впевненістю судити про їх надійності. Інші ж шифри, такі як HIGHT або mCrypton, вимагають для апаратної реалізації занадто багато місця на чіпі.

Таким чином, узагальнюючи все вищесказане, можна зробити висновок, що завдання створення як поточного, так і блочного алгоритму шифрування для пасивних RFID-міток до сих пір актуальна і потребує вирішення.

РОЗДІЛ 3
РОЗГЛЯД РЕЗУЛЬТАТІВ

3.1 Результати перевірки шифрів за допомогою FELICS

Оглядаючись на конкурси NIST на вибір нових криптографічних стандартів [16,17], ми можемо побачити, що слабкі конструкції з точки зору безпеки були дискваліфіковані після першого етапу оцінки. На наступних етапах інші алгоритми мали подібні запаси безпеки, і, отже, були необхідні нові критерії оцінки. Це момент, коли оцінка апаратного та програмного забезпечення кандидатів відіграє дуже важливу роль. Як зазначено в [18], остаточний рейтинг кандидатів тісно пов'язаний з показниками продуктивності апаратного та програмного забезпечення. Оскільки рамки бенчмаркінгу дозволяють проводити послідовну оцінку, вони важливі не тільки в процесі відбору нових криптографічних стандартів, але і для справедливого порівняння показників шифрування за певних сценаріїв використання.

Беручи до уваги зростаючий ринок пристроїв IoT та необхідність галузі для стандарту для захисту додатків IoT, потрібні інструменти, призначені для отримання показників продуктивності легких примітивів на різних платформах за однакових умов. Ці інструменти допомагають криптографам оцінювати запропоновані проекти щодо попередніх і можуть бути використані для розриву зв'язку між кандидатами на наступних етапах процесу відбору.

Розглянемо доступні на даний момент інструменти та оберемо найефективніший. Було розроблено кілька платформ для тестування, щоб спростити оцінку криптографічних примітивів на різних апаратних або програмних платформах. На додаток до цих систем порівняльного аналізу були опубліковані огляди і статті [26,27,28,29,30]. Для кожного проаналізованого проекту розглянемо проектні вимоги і обмеження, витягнуті метрики і методологію, яка використовується для забезпечення справедливої і послідовної

оцінки.

Проект BLOC [26] спрямований на вивчення конструкції блокових шифрів, призначених для обмежених середовищ. В ході проекту було опубліковано статтю [27] про оцінку продуктивності полегшених блокових шифрів для бездротових сенсорних вузлів. Реалізованих на мові C досліджуваних шифрів разом з вихідним кодом, використовуваним для вилучення аналізованих метрик, доступні безкоштовно. Цільовим пристроєм є 16-розрядний мікроконтролер MSP430F1611, що часто використовується в вузлах датчиків. Три розглянутих показника (час виконання, вимоги до ОЗУ і розмір коду) витягуються для набору з 17 шифрів. Лічильник циклів вимірюється за допомогою симулятора циклічної точності MSPDebug. Вимога до ОЗУ визначається використанням стека для виконання розкладу ключів шифрування, операцій шифрування і дешифрування. Споживання стека обчислюється шляхом налагодження за допомогою msp430-gdb виконання програми на симуляторі MSPDebug. Точки зупинки вставляються на початку і в кінці виконання програми, а потім обчислюється кількість змінених слів у пам'яті. Дані, необхідні для зберігання стану шифру, головного ключа і ключів раунду, не входять до вимоги до ОЗУ. Розмір коду задається текстовою частиною виконуваного файлу і витягується за допомогою інструменту розміру msp430. Витяг показників виконується автоматично за допомогою скриптів Bash, а результати експортуються в таблиці LaTeX. Аналізуючи вихідний код проекту, можна прийти до висновку, що у бібліотеки є серйозні недоліки. По-перше, вимоги до ОЗУ, наведені в документі і на веб-сайті проекту, обчислені неправильно, оскільки розробники інфраструктури припускають, що тип даних unsigned int вимагає одного байта замість двох на 16-бітному мікроконтролері MSP430F1611. Таким чином, вимоги до оперативної пам'яті, зазначені в документі, складають половину фактичного значення. По-друге, бібліотека зовсім не гнучка і не дозволяє легко додавати нові пристрої або метрики. Надана бібліотека не має набору вимог, яким повинна слідувати кожна реалізація, і немає загального інтерфейсу для оцінки продуктивності реалізованих шифрів. Без чіткої методології оцінки еталонні реалізації, які обробляють один блок за раз,

порівнюються з реалізаціями з поділом бітів, які обробляють кілька блоків паралельно. По-третє, деякі реалізації досліджуваних шифрів не перевіряють тестові вектори (наприклад, LBlock). Також проект зупинився більш десяти місяців назад, тому можна не очікувати покращення та виправлення помилок. Перевагою проекту є те, що він є однією з перших спроб оцінити набір легких блокових шифрів на вбудованому пристрої. Він також містить одну з найбільших колекцій легких реалізацій шифрів, доступних безкоштовно.

Проект EBACS (ECRYPT Benchmarking of Cryptographic Systems) [27] був розроблений під час проекту ECRYPT II для вимірювання швидкості широкого спектра криптографічних примітивів на персональних комп'ютерах і серверах. Він об'єднує функції для вимірювання швидкості виконання систем з відкритим ключем (eBATS), потокових шифрів (eSTREAM) і хеш-функцій (eBASH). Розроблена структура SUPERCOP (Система уніфікованої оцінки продуктивності, що відноситься до криптографічних операцій і примітивам) забезпечує великий набір реалізацій криптографічних примітивів. Відкритий і безкоштовний вихідний код фреймворка написаний на C з вбудованою складанням, Bash і Python.

На веб-сторінці проекту представлена інформація про те, як відправляти нові реалізації, а також як збирати дані для існуючих реалізацій за допомогою платформи. Вимоги, яким повинна відповідати реалізація криптографічного примітиву, щоб його можна було оцінити за допомогою фреймворка, дуже добре описані і забезпечують узгоджену оцінку всіх реалізацій на всіх розглянутих цільових платформах. Платформа дозволяє проводити порівняльний аналіз реалізацій C, C ++ і assembly. Він автоматично компілює вихідний код, використовуючи різні компілятори і параметри компілятора. Показник кількості циклів обчислюється з використанням вбудованих інструкцій збірки для кожної з підтримуваних платформ. Оскільки час виконання - єдина яку видобувають метрика, представлені

FELICS (Fair Evaluation of Lightweight Cryptographic Systems) [19] - це безкоштовна, відкрита та гнучка структура, яка оцінює ефективність реалізації

програмного забезпечення C та assembly легких примітивів на вбудованих пристроях. Завдяки модульній конструкції, фреймворк може легко вмістити нові показники, сценарії використання або цільові пристрої. Це ядро зусиль щодо підвищення прозорості роботи легких алгоритмів, має на меті полегшити чесне порівняння оцінених алгоритмів. Таким чином існує веб-сторінка [19], куди можна завантажити інструмент та знайти оцінені результати примітивів. Наскільки відомо, це єдина платформа порівняльного тестування з відкритим кодом, розроблена для чесної та послідовної оцінки програмних реалізацій легких примітивів на різних вбудованих пристроях IoT в тих самих сценаріях використання. Оскільки в наступні роки очікується значне зростання сфери IoT, FELICS допоможе дослідницькому співтовариству та промисловості за допомогою справедливих та детальних показників ефективності легких примітивів. Опубліковані легкі конструкції дають різні показники продуктивності на різних платформах та різні умови оцінки. Точна методологія, яка використовується для отримання цифр, зазвичай незрозуміла. Враховуючи, що показники ефективності, як правило, повідомляються для різних пристроїв, як і вимірювані операції. Умови вимірювання відрізняються в різних документаціях, дуже важко використовувати дані значення для порівняння різних конструкцій. Відсутність порівняльних показників продуктивності створює необхідність у справедливому та послідовному способі отримання показників ефективності для легких шифрів. Результати, отримані з використанням однієї і тієї ж методології оцінки, можуть бути використані для порівняння різних алгоритмів. Використовуючи значення продуктивності, дизайнери шифру можуть зробити висновок, які конструкції конструкції кращі для різних архітектур. У той же час результати можуть допомогти інженерам вибрати найкращий шифр для конкретного випадку використання. Якщо перші запропоновані легкі шифри в основному були спрямовані на апаратну ефективність, то за останні роки ми помічаємо, що зараз фокус переходить до легких шифрів, розроблених для підвищення ефективності програмного забезпечення. Цей новий напрямок дизайну для легких шифрів посилює потребу

в надійних і точних показниках продуктивності.

Для подальших досліджень був обраний саме FELICS, бо він був створений, щоб дозволити порівняння програмних реалізацій С мови легких шифрів на різних вбудованих пристроях, які зазвичай використовуються в IoT. Основними характеристиками або цілями проекту є:

– Чесна оцінка. Для забезпечення справедливої оцінки була сформульована чітка методологія оцінки. Методологія чітко вказує, яким вимогам має відповідати кожна реалізація та як витягується кожен показник для кожного підтримуваного пристрою. Хоча іноді методологію можна вважати обмежувальною, вона створюється для забезпечення справедливої оцінки кожного впровадження.
– Послідовна оцінка. Ця ж методологія використовується для оцінки ефективності всіх реалізацій даного примітивного типу. Таким чином, оцінка узгоджується на всіх цільових вбудованих пристроях для всіх вивчених сценаріїв використання. Послідовна оцінка дозволяє легко порівняти показники продуктивності між подібними реалізаціями шифрів. Це також сприяє правильному ранжуванню реалізацій шифрів з використанням різних критеріїв.
– Точні вимірювання. Фреймворк повинен забезпечувати точні вимірювання. Для досягнення цієї мети інструменти, що використовуються для отримання метрик та умови вимірювання, повинні бути точними. Тренажери повинні мати точність циклу, а вимірювання на платах повинні бути ретельно відкалібровані.
– Безкоштовність. Для забезпечення широкого використання вихідний код системи порівняльного тестування та вихідний код оцінених реалізацій доступні безкоштовно [19]. Вихідний код дозволяє аналізувати та розуміти зв’язок між показниками ефективності та аспектами реалізації.
– Відкритий код. Для збільшення довіри до вимірювань вихідний код фреймворку відкритий. Будь-хто може проаналізувати вихідний код, виявити та виправити помилки кодування або навіть співпрацювати з

розробкою інструменту за допомогою нових модулів та функцій.

- Прозорість. Фреймворк спрямований на забезпечення повної прозорості способу збору показників для кожного впровадження. Таким чином, окрім публікації вихідного коду оцінених реалізацій, результати, отримані для кожної реалізації, публікуються на нашій веб-сторінці [19].
- Комплексні результати. Витягнуті показники дуже детальні і мають на меті допомогти зрозуміти, як різні частини реалізації алгоритму впливають на ефективність. Вбудовані програмні інженери можуть використовувати всебічні результати, щоб вибрати найкращі компроміси для конкретного випадку використання.
- Гнучкість. Фреймворк використовує модульну архітектуру, що сприяє подальшому розвитку. FELICS призначений для подальшої розробки нових модулів для оцінки інших типів криптографічних примітивів. Це також дозволяє інтегрувати нові цільові пристрої та показники. Процес інтеграції нової реалізації шифру дуже простий, і це можна зробити, дотримуючись методології та вимог основи.
- Автоматичне оцінювання. Фреймворк може перевірити, чи реалізація відповідає сформульованим вимогам. Він може автоматично перевірити, чи реалізація перевіряє тестові вектори, надані реалізатором для всіх цільових пристроїв. Процес збору показників ефективності повністю автоматизований і може виконуватися в пакетному режимі. Користувач може витягти результати для певного списку шифрів та для певного списку архітектур.

FELICS написаний на мові C з вбудованою компіляцією, Bash та Python і призначений для роботи в операційних системах Linux. Це дозволяє провести порівняльний аналіз реалізацій C та assembly, які відповідають заданому набору або вимогам. Мова програмування C була обрана через широке поширення та портативність. Якщо взяти до уваги, що зазвичай еталонні реалізації надаються мовою C, то природно вибирається збільшення кількості цільових користувачів. У той же час, фреймворк може націлювати декілька вбудованих пристроїв, що

використовуються в контексті IoT, використовуючи одну реалізацію, отже, обмежену. FELICS також сприяє бенчмаркінгу високооптимізованих реалізацій збірки, які є залежними від платформи. Сценарії використання написані на C, тоді як реалізації шифрів можуть бути написані на C або assembly. Кожен модуль має файл make, який може побудувати реалізацію для даної архітектури та сценарій. Фреймворк містить колекцію сценаріїв Bash (Bourne Again SHell), які дозволяють повністю автоматизувати процес видобутку метрики. Скрипти Python використовувались для виконання занадто комплексних операцій, які складно або неможливо зробити в Bash. FELICS здатний автоматично генерувати двійковий код, перевірити правильність реалізації за допомогою наданих тестових векторів та витягти реалізацію метрики для підтримуваних пристроїв.

Поточна версія фреймворку включає основний модуль, модуль для оцінки легкої ваги блочні шифри та модуль для оцінки характеристик легких потокових шифрів. Завдяки модульній структурі, зображеній на малюнку 3.1, FELICS можна легко розширити за допомогою нових модулів для вимірювання інших примітивів.

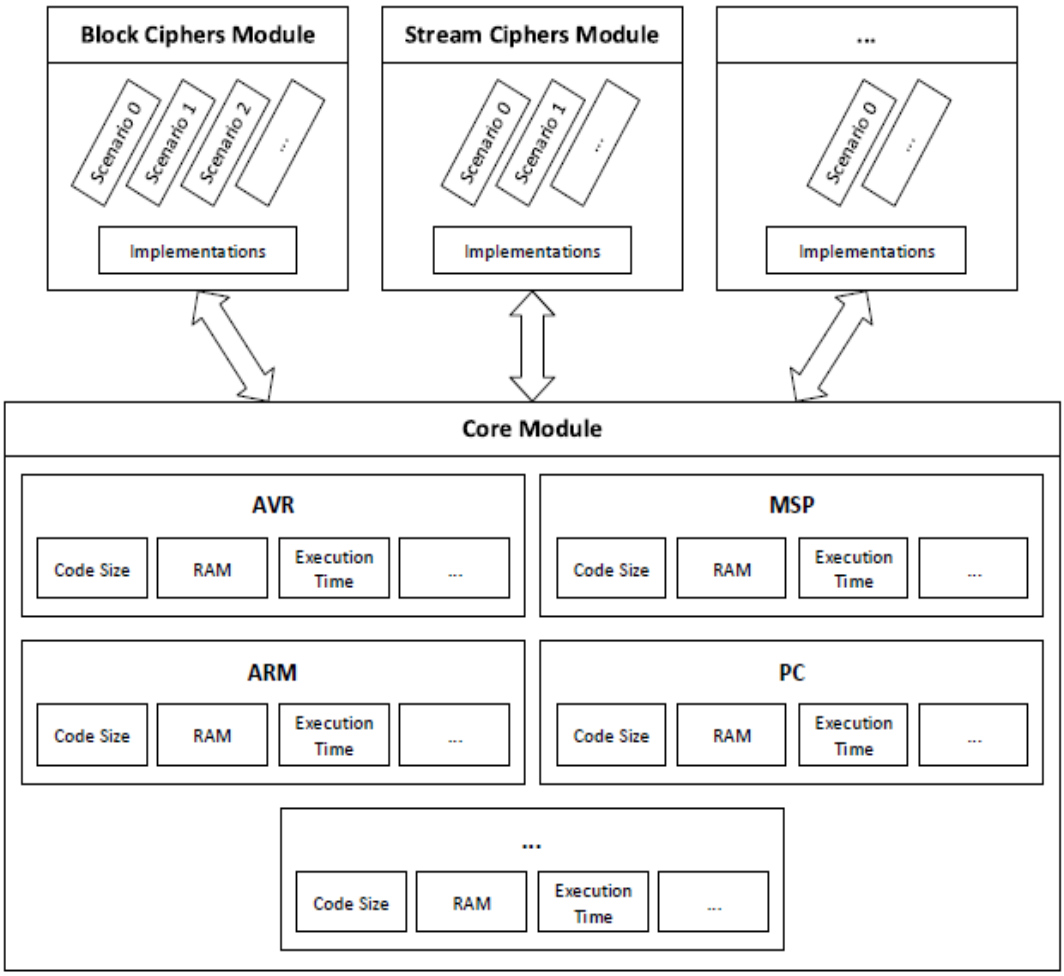

Рисунок 3.1– Модульна структура FELICS

Базовий модуль є серцем фреймворка і надає інструменти, необхідні для збору показників для кожного з підтримуваних пристроїв. Повний список використовуваних інструментів і версій інструментів, організований по витягнутим. Роль модуля полягає в полегшенні інтеграції нових цільових пристроїв і витягнутих метрик. Це дозволяє користувачеві збирати результати для одного або декількох модулів, які інтегровані в структуру. Крім підтримуваних вбудованих пристроїв, модуль дає можливість налагоджувати і оцінювати реалізації шифрів для персональних комп'ютерів. Ця функція в основному додана для зменшення складності процесу впровадження і інтеграції, а також полегшення завдання користувачам.

Для досягнення описаних цілей проектування кожен модуль формулює певний набір вимог, що кожна реалізація повинна слідувати. Незважаючи на те,

що вимоги створюють додаткові обмеження і обмежують можливість тестування високооптимізовані реалізацій, вони забезпечують справедливу і послідовну оцінка у всіх реалізаціях. У кожного модуля є файл конфігурації conf.sh, який надає модулю інформацію про інструменти в базовому модулі, які можна використовувати для отримання проаналізованих показників для кожного цільового пристрою. В той же час кожен модуль повинен надавати скрипт get results.sh, який може бути викликаний з ядра. Модуль для вилучення показників продуктивності в пакетному режимі. Розмір блоку, розмір ключа, розмір круглих ключів та кількість раундів шифру повинні бути визначені у файлі constants.h. Константи, що використовуються реалізацією, повинні бути оголошені в одному і тому ж файлі заголовка, тоді як визначення можна додати до файлу constants.c або будь-якого іншого файлу * .c, крім заздалегідь визначених файлів C. Константи можна зберігати у флеш-пам’яті або оперативній пам’яті пристрою і слід читати з відповідними макрокомандами. FELICS автоматично перевіряє, чи кожна реалізація перевіряє тестові вектори, вказані у файлі test vectors.c.

Файл інформації про реалізацію, Implementation.info, надає деталі реалізації для фреймворку, так що загальний код та дані повинні бути враховані лише один раз під час вилучення метрик. Реалізатори можуть розділити реалізацію на скільки завгодно файлів, якщо кожен файл реалізації правильно вказаний у файлі realisation.info. Файл інформації про реалізацію вказує, чи використовується розклад ключів для шифрування та дешифрування.

Реалізація шаблону шифру та файл, що описує вимоги до модуля, надається разом із модулем, щоб допомогти користувачам інтегрувати нові реалізації. Таким чином, процес інтеграції існуючої реалізації C або assembly дуже простий і полягає у заповненні функцій шифру шаблону реалізацією шифру, дотримуючись правил, описаних у файлі README.

Модуль містить три сценарії оцінки, але їх можна легко розширити за допомогою нових сценаріїв оцінки. Операція шифрування (сценарій 0) оцінює основні операції, що виконуються блочним шифром. У цьому сценарії блок даних шифрується, а потім дешифрується за допомогою наданих тестових

векторів. Протокол зв'язку (сценарій 1) передбачає шифрування та дешифрування 128 байт даних за допомогою режиму CBC. Цей сценарій підходить для безпечного зв'язку в контексті IoT та враховує обмеження протоколів IEEE 802.15.4 та ZigBee, що використовуються в сенсорних мережах. Протокол автентифікації виклик рукостискання (сценарій 2) створений для задоволення потреб аутентифікації в контекст IoT за допомогою блок-шифру в режимі CTR для шифрування 128 біт даних. Перелік ключів не потрібен, оскільки раундові ключі шифру попередньо обчислюються та зберігаються у флеш-пам'яті пристрою. Оскільки протоколи зв'язку та сценарії виклику рукостискання передбачають шифрування 128 байт і 128 бітів недоповнених даних відповідно, розмір блоку шифру в бітах повинен дорівнювати або ділити 128.

Показники продуктивності потокових шифрів можна отримати для кожної реалізації потокового шифру. Визначення кожної функції потрібно помістити в окремий файл C. Реалізація функції шифрування повинна мати можливість обробляти принаймні один байт. Процес шифрування виконується на місці, щоб зменшити споживання оперативної пам'яті. Головний ключ шифру не слід змінювати після запуску налаштування.

Розмір стану шифру, розмір ключа та розмір вектора ініціалізації повинні бути визначені у файлі constants.h. Константи, що використовуються потоковим шифром, повинні бути оголошені у файлі constants.h та визначені у файлі constants.c або будь-якому іншому файлі * .c, за винятком попередньо визначених файлів * .c. Реалізатор може вибрати для зберігання константи у Flash або ОЗУ і повинен використовувати відповідний макрос для зчитування констант. Тестові вектори, що використовуються FELICS для перевірки правильності реалізації, повинні бути визначені у файлі test vectors.c.

Інтеграція нової реалізації потокового шифру дуже проста, тому що користувач отримує шаблонну реалізацію потокового шифру та файл із інструкціями щодо реалізації. Реалізатор повинен просто заповнити функції із шифру шаблону вихідним кодом шифру відповідно до вимог, описаних у файлі README.

FELICS аналізує файл Implementation.info, щоб мати змогу підрахувати загальний вихідний код і константи лише один раз у витягнутих показниках. Реалізація кожної з необхідних функцій може бути розділена на кілька файлів за умови, що інформація про реалізацію правильно вказана у файлі реализации.info.

Для цього модуля реалізовано два сценарії оцінки, але нові сценарії можна додати будь-коли з мінімальними зусиллями. Робота шифру (сценарій 0) оцінюється за допомогою наданих тестових векторів. Протокол зв'язку (сценарій 1) призначений для захисту зв'язку між вузлами бездротових датчиків і полягає в шифруванні 128 байт даних. Оскільки умови оцінки подібні до тих, що використовуються для блочних шифрів, ці сценарії також можуть бути використані для порівняння показників ефективності блокових та потокових шифрів.

Отже фреймворк може експортувати витягнуті результати для кожного сценарію та цільову архітектуру у декілька форматів, щоб дозволити користувачеві аналізувати та обробляти результати. Підтримувані формати: таблиця необроблених даних, файл CSV, файл XML, сумісний з Microsoft Office Excel та LibreOffice Calc, таблиця MediaWiki та таблиця LaTeX. За необхідності можна легко додати нові формати. Архів із останніми результатами у всіх згаданих форматах доступний для завантаження на веб-сторінці FELICS [19]. На цій же веб-сторінці також можна знайти скрипт Python для обробки результатів CSV. Це дозволяє ранжувати існуючі реалізації шифрів, використовуючи показник заслуг (FOM), але може бути легко модифікований для обчислення інших цікавих значень.

3.2 Результати аналізу шифрів PRESENT та Humming bird-2

Проаналізовано характеристики двох різних апаратних реалізацій Humming bird-2. Дизайн високої продуктивності HB2-ee4c з 4-х тактовою реалізацією потрібно 3220 GE. Низька площа та потужність конструкції HB2-ee20c з 20 тактовими частотами на слово вимагає порівняно меншої потужності

та меншої вимоги до площі 2159 GE.

Апаратна реалізація PRESENT вимагає споживання енергії 5 мкВт і вимоги до площі 1570 GE та 32 тактові цикли для шифрування 64-бітного звичайного тексту за допомогою 80-бітного ключа. Порівняльний їх аналіз апаратні реалізації наведені в наступній таблиці 3.1.

Таблиця 3.1 – Порівняння апаратних реалізацій криптографічних алгоритмів

Алгоритм	Частота	Тактів за раунд	Піковий струм(μw)	GE
HB2-ee4c	100KHz	4	1.93	3220
HB2-ee20c	100 KHz	20	1.73	2159
PRESENT	100 KHz	32	5	1570

Пропускна здатність різних криптографічних алгоритмів з робочою частотою 80 МГц наведена в таблиці 3.2.

Таблиця 3.2 – Порівняння показників легких криптографічних алгоритмів

Шифр	Довжина блоку	Циклів за блок	Розмір ключа	Пропускна здатність при (80 MHz) в Mbps
Humming bird-2	16	4	128	300
PRESENT	64	32	80	160
HIGHT	64	34	128	150.6
DESXL	64	144	184	35.6
AES	128	1032	128	9.9

Пропускна здатність вказує на швидкість обробки шифру, тобто як швидко він перетворює звичайний текст зашифрувати текст. Пропускна здатність Humming bird-2 вище, ніж пропускна здатність інших алгоритмів, що показує швидкість обробки humming bird-2 швидша за інші алгоритми. Пропускна здатність різних шифрів за однакових умов порівнюється на наступному малюнку 3.2. Встановлено, що humming bird-2 має найвищу пропускну здатність.

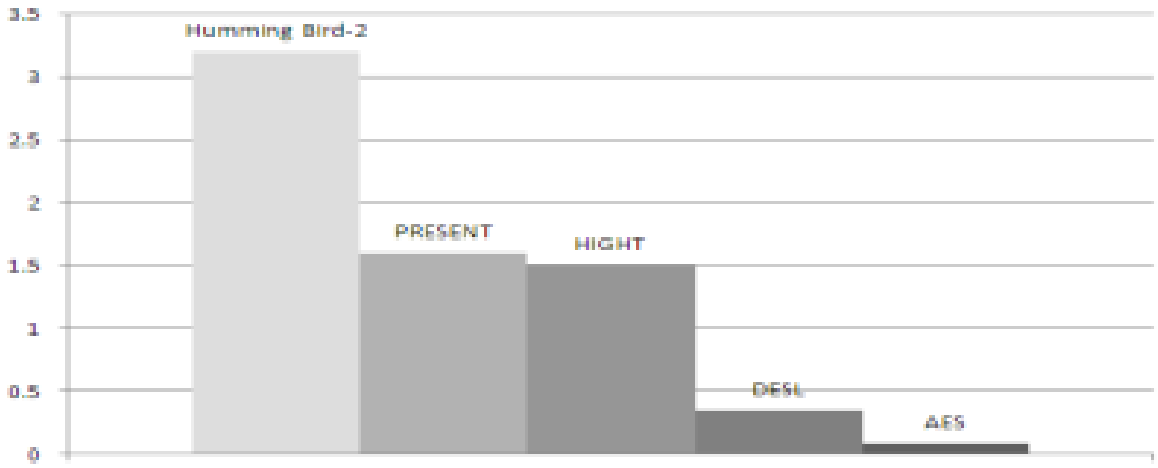

Рисунок 3.2 – Порівняння пропускної здатності різних шифрів при робочій частоті 80 МГц

humming bird-2 може протистояти атакам безпеки, таким як диференціальний криптоаналіз [20] та лінійний криптоаналіз [21]. З аналізу безпеки встановлено, що humming bird-2 стійкий до лінійного криптоаналізу до 12 раундів f. Чотири обертання у фазі ініціалізації створюють хороший опір проти атаки пов’язаних ключів. Алгебраїчний ступінь і кількість гілок S-Вохів перешкоджають різним формам алгебраїчних нападів.

SBoxLayer та pBoxLayer у PRESENT забезпечують хороший захисний механізм проти диференціальног та лінійного криптоаналізу. Для лінійного криптоаналізу PRESENT потрібно більше 2^{84} бітів вхідного або шифротексту. Таких вимог до даних досягти неможливо. PRESENT також стійкий до супутніх ключових атак [22] та слайд-атаки [23]. Раунд-залежні ключі використовуються в PRESENT, так що набори допоміжних ключів не можуть бути визначені слайд та лінійними операціями, які використовуються для змішування вмісту регістру ключів. Побітові операції в PRESENT витримують структурні атаки, такі як інтегральні атаки [24] та вузькі місця [19].

Зроблено порівняння продуктивності реалізацій ПЛІС Humming bird-2 та PRESENT з недорогими пристроями ПЛІС. Використовуються пристрої ПЛІС: Spartan-3 XC 35200 та Spartan-3 XC 35400. Зведення результатів реалізації показано в таблиці 3.3, де вимога до площі максимальна.

Таблиця 3.3 – Порівняння результатів реалізації ПЛІС криптографічних алгоритмів [20]

Шифр	Довжина ключа	Розмір блоку	ПЛІС пристрій	Частота	Пропускна здатність	Slice	Ефективність
Humming bird-2	128	16	S-3 XC 35200	40.1	160.4	273	0.59
PRESENT	80	64	S-3 XC 35400	258	516	176	2.93
PRESENT	128	64	S-3 XC 35400	254	508	202	2.51

Задано частоту, пропускну здатність та ефективність. Результати показують, що PRESENT є більш ефективним, ніж humming bird-2.

У цій роботі представлені результати реалізації легких криптографічних аналізів алгоритмів Humming bird-2 та PRESENT. З аналізу безпеки встановлено, що обидва алгоритми забезпечують належну безпеку. Апаратна реалізація PRESENT вимагає порівняно меншої площі. PRESENT є більш ефективним в плані впровадження ПЛІС. Але енергоспоживання Humming bird-2 порівняно менше в апаратному забезпеченні впровадження. Пропускна здатність humming bird-2 вища за інший алгоритм. В аналізі було визначено, що humming bird-2 більше підходить як криптографічний алгоритм для пристроїв, обмежених ресурсами.

3.3 Висновки до 3 розділу

У цьому розділі було розглянуто фреймворк FELICS і результати аналізів шифрів PRESENT та humming bird-2.

FELICS являє собою безкоштовну систему порівняльних показників для справедливого та послідовного оцінювання криптографічних примітивів. Метою

фреймворку є – підвищення довіри та прозорості результатів, отриманих за допомогою різних алгоритмів, та забезпечити незалежне середовище для оцінки характеристик нових конструкцій. FELICS підтримує порівняння показників ефективності між різними шифрами завдяки послідовній методології оцінки. На даний момент фреймворк здатний порівняти легкі шифри блоків і потоків на трьох різних вбудованих пристроях. Для кожного пристрою та сценарію оцінки витягуються такі показники: розмір двійкового коду, споживання оперативної пам'яті та час виконання. Завдяки своїй модульній конструкції FELICS дуже гнучкий і може бути легко розширений для порівняння нових легких примітивів, вилучення нових метрик, збору показників ефективності для інших цільових пристроїв або оцінки впроваджених алгоритмів у нових сценаріях використання. Вихідний код фреймворку разом із введеними вихідними кодами шифрів та показниками продуктивності доступні на нашому веб-сайті [19]. FELICS запозичує та вдосконалює концепції з попередніх фреймворків і одночасно додає нові ідеї та функції. Результатом є краща основа для справедливої та послідовної оцінки криптографічних примітивів.

ВИСНОВКИ

Неминуче розширення Інтернету речей створює новий світ інтелектуальних пристроїв, в яких питання безпеки дуже важливе. Якщо врахувати, що схеми стимулятора мозку та кардіостимулятори серця можуть бути безпосередньо підключені до мережі, щоб надати лікарям корисну інформацію при встановленні та корекції терапії без фізичного огляду пацієнта, безпека відіграє вирішальну роль, оскільки несанкціонований доступ до цих критично важливих пристроїв може загрожувати життю. Сектор охорони здоров’я - це лише одна із сфер, де очікується, що кількість пристроїв IoT значно зросте. Інші додатки IoT включають управління ланцюгами поставок, розумні будинки, зелені міста та багато іншого.

Окрім аспектів безпеки, IoT створює нові проблеми з точки зору споживання енергії. Таким чином, легкі криптографічні примітиви, призначені для пристроїв із підтримкою IoT, повинні споживати менше ресурсів, забезпечуючи при цьому заявлений рівень безпеки. У недалекому минулому інтерес дослідницької спільноти до легкої криптографії збільшився, і в результаті багато легких алгоритмів були розроблені та проаналізовані з точки зору безпеки. Зусилля, спрямовані на впровадження, були зосереджені на виборі найкращих конструкцій з метою зменшення споживання ресурсів, оцінці показників продуктивності, що досягаються апаратними та програмними реалізаціями на різних платформах, а також аналізі та вдосконаленні захисту від атак сторонніх каналів.

На даний момент відомі тільки три описаних потокових LW-шифру, що мають відносно прийняті характеристики. Це алгоритми Hummingbird, Trivium і GRAIN. Однак дані шифри не застосовні в пасивних RFID-системах в силу індивідуальних особливостей кожного з них. Так, наприклад, Trivium вимагає площа на чіпі, що перевищує допустиму більш ніж в півтора рази (3488 GE1 при обмеженні у 2000 GE). На поточну версію шифру GRAIN може бути успішно проведена атака на пов'язаних ключах. Що стосується Hummingbird, то

розробниками перевірена його стійкість лише до деяких атакам, однак цього недостатньо для забезпечення впевненості в його надійності. Таким чином, можна зробити висновок, що на даний момент серед поточних шифрів немає алгоритму, який задовольняє основним вимогам RFID-систем.

У розділі блокових шифрів ситуація виглядає дещо краще. Розглянемо докладніше деякі блокові LW-алгоритми. LW-алгоритмом, що задовольняє всім вимогам RFID-систем, є PRESENT. Даний шифр використовує ключ довжиною 80 біт, що значно підвищує його надійність. Розробниками проведено дослідження вразливості даного алгоритму до лінійного і диференційного аналізу, алгебраїчної атаки і деяким іншим видам атак. Показана PRESENT стійкість є прекрасним результатом для шифру, створеного «з нуля». На даний момент не відомо жодної успішної атаки на полнораундову версію алгоритму. Існують різні реалізації PRESENT. Найкомпактніша з них вимагає всього 1000 GE, що є одним з кращих результатів для LW-шифрів. Крім забезпечення безпеки даних, що передаються в RFID-системах, деякі модифікації PRESENT знайшли застосування і в інших ресурсозалежний пристроях. Так, наприклад, H-PRESENT-128 є самою компактною з відомих хеш-функцій. Крім того, можливе застосування алгоритму в якості генератора псевдовипадкових чисел для схеми crypto-GPS.

Наскільки відомо, FELICS є першим базовим тестом для оцінки легких примітивів для контексту IoT у різних сценаріях використання. Це також забезпечує повну прозорість показників ефективності шляхом опублікування результатів та відповідного вихідного коду на веб-сайті проекту. Можливі нові функції включають додавання нових модулів, які дозволяють порівняти інші криптографічні примітиви (автентифіковане шифрування, хеш-функції, криптографія відкритого ключа), вилучення нових показників або підтримка нових вбудованих пристроїв. Впровадження підтримки для плат розвитку, де використовуються програмні симулятори, є ще одним подальшим напрямком розвитку. Завдяки вільним і відкритим властивостям FELICS кожен бажаючий може внести свій внесок у розробку інструменту новими функціями та

виправленнями помилок. Відгуки користувачів та звіти про помилки допоможуть розширити можливості інструменту. Сприяння впровадженню шифру - ще один важливий напрям, який принесе користь всій спільноті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з «Розробки й оформлення магістерської атестаційної роботи» для студентів другого (магістерського) рівня вищої освіти за спеціальністю 125 Кібербезпека освітня програма «Безпека інформаційних і комунікаційних систем»/ Упоряд. І.Ш. Невлюдов, В.В. Косенко, В.В. Євсєєв. – Харків: ХНУРЕ, 2019. – 55 с.

2. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. - К.: ДП «УкрНДНЦ», 2016. – 31 с.

3 Основи наукових досліджень: [навч. посіб.] / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала. – Кривий Ріг : КК НАУ, 2017. – 344 с.

4. Положення про організацію освітнього процесу в ХНУРЕ [Електронний ресурс] / Режим доступу: www/ URL: https://nure.ua/polozhennya-pro-organizatsiyu-osvitnogo-protsesu-v-hnure - 29.08.2019р. – Загл.с екрана.

5. Положення про протидію академічному плагіату в ХНУРЕ [Електронний ресурс] / Режим доступу: www/ URL: https://nure.ua/wp-content/uploads/Main_Docs_NURE/Polozhennya-pro-protidiyu-akademichnomu-plagiatu-v-HNURE----290-vid-28.04.2017.pdf - 29.08.2019р. - Загл. с екрана.

6. Положення про роботу екзаменаційних комісій ХНУРЕ [Електронний ресурс] / Режим доступу: www/ URL: https://nure.ua/wp-content/uploads/Main_Docs_NURE/Polozhennya-pro-poryadok-stvorennya-ta-organizatsiyu-roboti-ekzamenatsiynih-komisiy....pdf - 29.08.2019р. - Загл. с екрана.

7. Положення про авторське право в ХНУРЕ [Електронний ресурс] / Режим доступу: www/ URL: https://nure.ua/wp-content/uploads/Main_Docs_NURE/Polozhennya-pro-avtorske-pravo-v-HNURE.pdf - 29.08.2019 р. – Загл. с екрана.

8. С. С. Агафїн LW-КРИПТОГРАФІЯ: ШИФРИ ДЛЯ RFID-СИСТЕМ, // Безпека інформаційних технологій № 2011-4.

9. Jacob J. Performance Analysis of New Light Weight Cryptographic Algorithms. // Sinhgad In-stitute of Technology, Pune, India. 2012.

10. Poschmann, A. 256 bit standardized crypto for650 GE: GOST revisited [Text] / A. Poschmann,S. Ling, H. Wang // Proceedings of the 12thinternational conference on Cryptographic hardwareand embedded systems in CHES'10. Springer-Verlag.Berlin, Heidelberg, 2010. – P. 219 – 233..

11. Qingju W., Bogdanov A. T The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers

12. Christophe De Canniere, Orr Dunkelman, Miroslav Knezevic, “KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers”// Proceedings of CHES 2009, LNCS 5747, Springer, 2009.

13. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L. Performance of the SIMON and SPECK families of lightweight
block ciphers. Tech. rep., National Security Agency, May 2012.

14. Shibutani K., Isobe T., Hiwatari H., Mitsuda A., Akishita T., Shirai T. Piccolo: An ultra-lightweight blockcipher. The 13th International
Workshop on Cryptographic Hardware and Embedded Systems - CHES 2011, Lecture Notes in Computer Science v. 6917, pp.
342-357, 2011.

15. Knudsen L., Leander G., Poschmann A., Robshaw M. J. B. PRINTcipher: A Block Cipher for IC-Printing. The 12th International
Workshop on Cryptographic Hardware and Embedded Systems - CHES 2010, Lecture Notes in Computer Science v. 6225, pp.
16-32, 2010.

16. NIST FIPS Pub. 197: Advanced encryption standard (AES). Federal Information Processing Standards Publication, 197:441–0311, 2001.

17. National Institute of Standards and Technology (NIST). SHA-3 Competition. http://csrc.nist.gov/ groups/ST/hash/sha-3/.

18. National Institute of Standards and Technology (NIST). Lightweight Cryptography Workshop 2015. http: //www.nist.gov/itl/csd/ct/lwc_workshop2015.cfm.

19.CryptoLUX. FELICS -Fair Evaluation of Lightweight Cryptographic Systems. https://www.cryptolux. org/index.php/FELICS, 2015.

20 E. Biham, A. Shamir. “Differential Cryptanalysis of the Data Encryption Standard," Springer-Verlag,1993.

21 M. Matsui, “Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology - EUROCRYPT'93, T. Helleseth, Ed., LNCS
765, Springer-Verlag, pp. 386-397, 1994.

22. E.Biham. “ New Types of Cryptanalytic Attacks Using Related Keys”. In T.Helleseth, editor, Proceedings of Eurocrypt ’93, LNCS,
volume 765, pages 398-409, Springer-Verlag , 1994.

23. H. Gilbert and M. Minier. “A Collision Attack on 7 Rounds of Rijndael”. In Proceedings of Third Advanced encryption Standard
Conference, National Institutes of Standards and Technology, 230-241, 2000.

24. Xinxin Fan, Guang Gong, K. Lauffenburger, T. Hicks “ FPGA Implementations of the Hummingbird Cryptographic Algorithm”
Hardware Oriented Security and Trust (HOST) 2010 IEEE International Symposium.

25. A. Biryukov and D Wagner. “Advanced Slide Attacks”. In B . Preneel , editor, Proceedings of Eurocrypt 2000, LNCS, volume1807, pages 589-606, Springer – Verlag 2000.

26. L R Kundsen and D Wagner , “Integral Cryptanalysis”. In J.Daemen and V.Rijmen, editors, Proceedings of FSE 2002, LNCS ,volume 2365, pages 112-137, Springer – Verlag 2002.

27. Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Lo Perrin, Johann Groschdl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/209, 2015. http://eprint.iacr.org/.

28. Thomas Eisenbarth, Zheng Gong, Tim G¨uneysu, Stefan Heyse, Sebastiaan Indesteege, St´ephanie Kerckhof, Fran¸cois Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, et al. Compact implementation and performance evaluation of block ciphers in ATtiny devices. In Progress in Cryptology-AFRICACRYPT 2012, pages 172–187. Springer, 2012.

29. Thomas Eisenbarth, Zheng Gong, Tim G¨uneysu, Stefan Heyse, Sebastiaan Indesteege, St´ephanie Kerckhof, Fran¸cois Koeune, Tomislav Nad, Thomas Plos,

Francesco Regazzoni, et al. Implementations of low cost block ciphers in Atmel AVR devices. http://perso.uclouvain.be/fstandae/lightweight_ciphers/, February 2015.

30. Kris Gaj, J Kaps, Venkata Amirineni, Marcin Rogawski, Ekawat Homsirikamol, and Benjamin Y Brewster. Athena-automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using fpgas. In Field Programmable Logic and Applications (FPL), 2010 International Conference on, pages 414–421. IEEE, 2010.

31. Ekawat Homsirikamol, Marcin Rogawski, and Kris Gaj. Comparing hardware performance of fourteen round two sha-3 candidates using fpgas. Cryptology ePrint Archive, Report 2010/445, 2010. http://eprint.iacr. org/.

32. IEEE Standards Association. IEEE 802.15: WIRELESS PERSONAL AREA NETWORKS (PANs). http: //standards.ieee.org/about/get/802/802.15.html

33. Texas Instruments. MSP430F1611 datasheet. http://www.ti.com/lit/ds/symlink/msp430f1611.pdf.