Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська кваліфікаційна робота на тему:
«Засіб потокового шифрування на основі латинських квадратів»

Виконав: студент 4 курсу групи 1БКС–21б
спеціальності 125 Кібербезпека
_____ Олександр КОРІННИЙ

Керівник: к.фіз-мат.н., доц. каф. ЗІ
_____ Галина ШЕЛЕПАЛО
«16» червня 2025 р.

Рецензент: к. т. н. доцент каф. ПЗ
_____ Олена КОВАЛЕНКО
«16» червня 2025 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» червня 2025 р.

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України
Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека критичних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т./н., професор
Володимир ЛУЖЕЦЬКИЙ
«24» березня **2025 року**

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Олександру КОРІННОМУ

1. Тема роботи: «Засіб потокового шифрування на основі латинських квадратів»
   керівник роботи: Галина ШЕЛЕПАЛО, к.фіз-мат.н., доц. каф. ЗІ.
   затверджені наказом ректора ВНТУ від 20 березня 2025 року №97.
2. Строк подання студентом роботи 16 червня 2025 року.
3. Вихідні дані до роботи:
   - Сучасні засоби потокового шифрування на основі латинських квадратів;
   - Відомі алгоритми шифрування на основі квазігруп;
   - Вимоги до засобів LW-криптографії;
   - Латинські квадраи 0-луп 4-го порядку
4. Зміст текстової частини: Вступ. Аналіз інформаційних джерел. Методи шифрування на основі латинських квадратів. Розробка структури та оцінка складності засобу. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Порівняння характеристики алгоритмів шифрування. Схема апаратного засобу. Схема роботи алгоритму потокового шифрування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Шелепало Г.В., к.фіз.-мат. н., доцент	24.03.25	10.06.25
2	Шелепало Г.В., к.фіз.-мат. н., доцент	14.05.25	10.06.25
3	Шелепало Г.В., к.фіз.-мат. н., доцент	30.05.25	10.06.25

7. Дата видачі завдання 24.03.2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	24.03.25 – 30.03.25	
2	Аналіз літературних джерел за напрямком бакалаврської кваліфікаійної роботи	31.03.25 – 13.04.25	
3	Розробка рішень, моделей, алгоритмів	14.04.25 – 04.05.25	
4	Практична реалізація, моделювання, експериментування, результати	05.04.25 – 14.05.25	
5	Аналіз виконання БКР	15.04.25 – 18.05.25	
6	Оформлення пояснювальної записки	19.05.25 – 26.05.25	
7	Попередній захист БКР	27.05.25 – 30.05.25	
8	Представлення БКР до захисту, рецензування	31.05.25 – 16.06.25	
9	Захист БКР	17.06.25 – 20.06.25	

Студент _____ Олександр КОРІННИЙ
(підпис)

Керівник роботи _____ Галина ШЕЛЕПАЛО
(підпис)

АНОТАЦІЯ

Дана робота нараховує в собі 60 сторінок формату А4, на яких є 20 рисунків, 10 таблиць, список використаних джерел, що налічує 35 найменувань

Бакалаврська кваліфікаційна робота присвячена розробці апаратного засобу потокового шифрування на основі латинських квадратів. В результаті аналізу існуючих методів потокового шифрування малоресурсної криптографії обрано такий крипто примітив, як 0-лупи, тобто квазігрупи, що мають нейтральний елемент. Використання цього криптопримітиву дозволяє зменшити апаратну складність засобу потокового шифрування на основі латинських квадратів за рахунок зменшення його апаратної складності. Обчислено апаратну складність засобу потокового шифрування. Після розробки програмного засобу потокового шифрування, виконано його тестування.

Ключові слова: захист інформації, алгоритм шифрування, апаратний засіб шифрування, квазігрупи, LW-криптографія.

ABSTRACT

This work consists of 60 A4 pages, including 20 figures, 10 tables, and a list of sources used contains 35 items.

The bachelor's qualification thesis is devoted to the development of a hardware-based stream cipher using Latin squares. As a result of analyzing existing stream cipher methods in lightweight cryptography, a cryptographic primitive known as 0-loops – that is, quasigroups with a neutral element – was selected. The use of this primitive allows for a reduction in the hardware complexity of the stream cipher based on Latin squares. The hardware complexity of the developed encryption device has been calculated. After developing the software implementation of the stream cipher, it was tested.

Keywords: information security, encryption algorithm, hardware encryption device, quasigroups, lightweight cryptography.

ЗМІСТ

ВСТУП

Сучасний науковий прогрес здійснив потужний скачок протягом останнього сторіччя, за рахунок цього, суспільство отримало надзвичайно великі можливості для роботи й розвитку. Сьогодення, яке бачимо зараз, це результат останніх десятиріч, протягом цього періоду людство виявляє появу нових технологій, комп'ютерів, телефонів, розумних годинників і інших технологій, в кожному домі. З появою цих винаходів людство змінило свій підхід до способу життя.

В результаті наукового прогресу, найважливішим ресурсом, яким можна володіти, стала інформація. Можливість мати інформацію, змінювати її, керувати нею, видозмінює світовий ринок, що призводить до потреби захисту інформації.

Тенденція до зменшення апаратного забезпечення породжує й проблему захисту інформації. Малоресурсні системи: мобільні телефони; пристрої, що вживлюють в людські тіла, та інші; потребують захисту з використанням їх обмежених ресурсів. У таких пристоях використовуються засоби, алгоритми малоресурсної криптографії. Тому сьогодні набуває популярності LW-криптографія, що використовується у вбудованих системах. На відміну від систем, що можуть повноцінно функціонувати, вбудовані системи мають внутрішні обмеження, зокрема: потужність, пам'ять, зберігання енергії, тощо. LW-криптографія призначена саме для таких типів систем.

Передова ціль LW-криптографії – забезпечення гарантій безпеки, зводячи до мінімуму вплив на системні ресурси. Це може досягатись за допомогою певних варіантів, можливим спрощенням алгоритму, зменшенням розмірів ключів та пошук ідеального балансу між безпекою та продуктивністю алгоритмів.

Актуальність бакалаврської кваліфікаційної роботи полягає в розвитку технологій, в напрямку зменшення апаратної частини засобів та розвитку концепції малоресурсних пристроїв.

Метою бакалаврської кваліфікаційної роботи є зменшення апаратних витрат засобу потокового шифрування на основі латинських квадратів 4-го порядку.

Об’єктом дослідження є процес малоресурсного потокового шифрування на основі латинських квадратів 4-го порядку

Предметом дослідження є засіб для потокового шифрування на основі латинських квадратів.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Обґрунтування актуальності дослідження

З розвитком технологій збільшувались обчислювальні можливості пристроїв, як наслідок цьому, алгоритми шифрування, які в минулому мали високу криптостійкість та забезпечували надійних захист даних втратили свою актуальність в сучасних системах інформаційної безпеки.

Зростання обчислювальної потужності комп’ютерних систем має значний вплив на сферу інформаційної безпеки. Сучасні процесори, суперкомп’ютери, а також розподілені обчислювальні мережі здатні виконувати трильйони операцій за секунду. Усе це суттєво зменшує час, необхідний для злому криптографічних алгоритмів, які ще кілька років тому вважались практично незламними. У зв’язку з цим, науковці та фахівці з кібербезпеки постійно вдосконалюють існуючі алгоритми шифрування, збільшуючи довжину ключів, розробляючи нові алгоритми, що будуть стійкішими до існуючих атак.

Наприклад, алгоритм DES був одним з перших алгоритмів симетричного шифрування, що забезпечував надійний захист інформації протягом декількох десятиліть [1]. Проте з розвитком обчислювальної потужності та розвитком методів криптоаналізу його 56-бітний ключ має вразливість до атаки повним перебором. Перша спроба підбору ключа відбулась в 1997 році, коли на це витратили 5 місяць часу великої мережі комп’ютерів, всього за рік роботи цей проміжок часу скоротився до 56 годин на звичайному комп’ютері лише з деякими вдосконаленнями. Розвиток обчислювальної потужності призвів до втрати актуальності DES, як засобу забезпечення сучасної криптобезпеки на ринку інформаційних послуг.

Одним з недавніх алгоритмів, що використовувались компаніями, був потоковий алгоритм шифрування RC4 [2]. Алгоритм RC4, високошвидкісний потоковий шифр, тривалий час використовувався в багатьох мережевих протоколах, наприклад: TLS, SSL, WEP. Проте, були в цьому алгоритмі виявлені криптоаналітичні вразливості, зокрема, слабкість перших байтів

ключового потоку. У зв'язку з цим, в лютому 2015 компанія IETF запропонувала в документі RFC 7465 припинити застосування RC4 в протоколі та реалізаціях TLS. А в серпні 2016 року компанія Microsoft теж припинила використання RC4 в інтернет-браузерах [3].

Паралельно з розвитком потокових алгоритмів, вдосконаленням існуючих алгоритмів та створенням нових, також створюються методи їх зламу та інших маніпуляцій, що планують використовуватись для дискредитації системи.

Окрім того, в суспільстві виникає необхідність в створенні нових алгоритмів шифрування, що можуть працювати в різних типах обчислювальних систем. Зокрема, збільшення обсягів передавання даних у мережах Інтернету речей, а також активне впровадження блокчейн-технологій створюють необхідність розробки методів шифрування, що здатні працювати в умовах обмежених ресурсів.

В алгоритмах, що використовуються в звичайній криптографії, виникають проблеми в роботі у вбудованих системах, оскільки вони мають значну процесорну потужність та споживають велику кількість енергії. В наслідок цього, почали впроваджуватися концепції малоресурсних пристроїв, під такими пристроями розуміють апаратні засоби, реалізація яких, не вимагає великих затрат. Незважаючи на розмір цих пристроїв, в них також є потреба впроваджувати методи захисту інформації.

Тому, сьогодні набуває популярності LW-криптографія, що використовується у вбудованих системах. На відміну від систем, що можуть повноцінно функціонувати, вбудовані системи мають внутрішні обмеження, зокрема: потужність, пам'ять, зберігання енергії, тощо. LW-криптографія призначена саме для таких типів систем.

Передова ціль LW-криптографії – забезпечення гарантій безпеки, зводячи до мінімуму вплив на системні ресурси. Це може досягатись за допомогою певних варіантів, можливим спрощенням алгоритму, зменшенням розмірів ключів та пошук ідеального балансу між безпекою та продуктивністю алгоритмів.

Функції та обґрунтування LW-криптографії:

– Ефективність: легкі криптографічні примітиви створені для продуктивних обчислень та вимагають менше пам’яті і обчислювальної спроможності. LW-криптографія прагне досягнути балансу між безпекою і продуктивністю.

– Низькоенергоспоживаність. Ця функція забезпечує оптимізацію мінімального споживання енергії, оскільки пристрої, для яких важлива LW-криптографія, здебільшого не підключені до мережі живлення і мають власну систему енергозберігання. Така оптимізація алгоритмів гарантує, що криптографічні операції виконуватимуться з мінімальним впливом на загальне використання ресурсів.

– Малоресурсність. Ця функція забезпечує програмно невеликий код та об’єм пам’яті. Алгоритми розробляються з врахуванням до розміру коду та до вимог пам’яті. У пристроях з обмеженою ємністю пам’яті це особливо відіграє значну роль.

– Стійкість до атак: незважаючи на свою мало ресурсну природу, легкі криптографічні алгоритми однозначно мають забезпечувати відповідний рівень захисту від різних типів атак, включаючи атаки типу грубої сили та інших типів атак.

– Гнучкість. Ця функція забезпечує можливість вибору і методів, і пристроїв для застосування. Легка криптографія має на меті стати універсальним варіантом щодо вибору алгоритмів, дозволяючи розробникам мати можливість вибирати необхідні варіанти алгоритмів відповідно до їх конкретних вимог, включаючи рівень безпеки, продуктивність та обмеження ресурсів, тощо [4].

LW-криптографія забезпечує захищений обмін даними, безпечний зв'язок та автентифікацію на пристроях, де використання ресурсів продуктивності та енергоспоживання є важливим аспектом у роботі пристрою. Це відіграє ключову роль у системі пристроїв Інтернету речей та інших подібних системах.

1.2 Аналіз потокових алгоритмів шифрування

З розвитком шифрів, почали також виокремлюватись певні патерни, які характеризують шифри, наприклад, симетричні шифри, тобто ті шифри, у яких для шифрування і розшифрування використовується один і той же ключ. Такими шифрами є, наприклад: AES, DES [5].

Також, є група асиметричних шифрів, тобто такі, що мають пару ключів, закритий та відкритий ключ. Найпоширеніший шифр цього типу є RSA, він використовується в цифрових підписах, SSL-сертифікатах [6].

Виділяють групи шифрів: потокові і блокові шифри. Для початку розглянемо блокові шифри. Блоковий шифр працює з блоками відкритого тексту фіксованого розміру, перетворюючи кожен блок на блок зашифрованого тексту такого ж розміру.

Потокові шифри, на відміну від блокових, шифрують тексту по одному біту або байту. Вони генерують псевдовипадковий потік ключів, який потім обробляється XOR з відкритим текстом для створення зашифрованого тексту. Самі потокові шифри можна класифікувати на два типи:

- синхронні;
- самосинхронізовані.

Синхронні потокові шифри включають в себе потік ключів, що генерується незалежно від відкритого та зашифрованого тексту. Прикладом можуть бути алгоритми: RC4, Salsa20, STRUMOK (ДСТУ 8845:2019), алгоритм A5/1, що використовується в мобільному GSM зв’язку [7-9].

В алгоритмі A5/1 кожному символу відкритого тексту відповідає символ шифротексту, текст не ділиться на блоки і не змінюється в розмірі, для спрощення апаратної реалізації а, отже, збільшення швидкодії використовуються лише найпростіші операції: XOR та зсув регістру.

Алгоритм RC4 – це потоковий шифр, що використовувався в різних системах захисту інформації в комп’ютерних мережах, наприклад, в протоколах SSL та TLS, алгоритмах забезпечення безпеки бездротових мереж WEP та WPA. Цей шифр будується на основі генератора псевдовипадкових

бітів, на вхід записується ключ, а на виході читаються псевдовипадкові біти. Довжина ключа може становити від 40 до 2048 біт.

Алгоритм ініціалізує масив станів секретним ключем, а потім генерує потік ключів шляхом перестановки масиву станів. Ключовий потік обробляється XOR з відкритим текстом для створення зашифрованого тексту. Незважаючи на історичну популярність, RC4 зараз вважається небезпечним через уразливості в його генерації потоку ключів.

Розглядаючи потокові алгоритми шифрування, які створені для реалізації на пристроях з обмеженими ресурсами, варто відмітити проект, для виявлення потокового шифру придатного для широкого прийняття eSTREAM. Історія цього проєкту розпочалась з невдач шифрів, що були подані до проєкту NESSIE [10]. Алгоритми, що подавались до eSTREAM потрапляли до одного чи двох профілів. Перший профіль визначав потокові шифри для програмного забезпечення з високими вимогами до пропускної здатності, відповідно, другий профіль визначав потоковий шифр для апаратного забезпечення з обмеженими ресурсами, наприклад, обмежене місце, кількість транзисторів чи інших пристроїв, або споживання енергії.

Переможцями конкурсу eSTREAM, які призначені для потокового шифрування у апаратному забезпечені з обмеженими ресурсами, стали 3 алгоритми, зокрема: Grain, MICKEY, та Trivium [11-13].

Grain – це симетричний шифр синхронного потокового шифрування, який в першу чергу орієнтований на апаратну реалізацію [11]. Цей шифр був поданий в 2004 році на конкурс eSTREAM. Шифр складається з трьох основних блоків: двох 80-бітних регістрів зсуву зі зворотнім зв’язком і вихідною функцією h(x). Один з регістрів є лінійною функцією зворотного зв’язку LFSR, другий регістр має нелінійну функцію зворотного зв’язку NFSR. Шифр Grain також дуже компактний і легко реалізується в апаратних засобах. Крім того, можна досить просто збільшувати швидкість за рахунок більшої кількості апаратних засобів, тобто тригерів. Це є відмінністю сімейства потокових шифрів Grain, і в багатьох інших шифрах явно не обгрунтовано.

Наступним переможцем конкурсу був MICKEY 2.0 [12]. Існує дві версії цього шифру, з довжиною ключа 80 біт та 128 біт, MICKEY-128. Цей шифр має два вхідних параметри: 128-бітний секретний ключ та вектор ініціалізації довжиною від 0 до 128 біт. Цей алгоритм має просту апаратну реалізацію за високого ступеня захищеності. У ньому використовується нерегулярне тактування зсувних регістрів, а також нові методи, що забезпечують досить великий період та псевдовипадковість ключової послідовності та стійкість до атак. Алгоритм MICKEY брав участь у конкурсному проекті eSTREAM, організованому спільнотою eCRYPT. Поточна версія алгоритму – 2.0. Вона увійшла в портфоліо eCRYPT як потоковий шифр для апаратної реалізації.

Третій і останнім переможцем eSTREAM був шифр Trivium. Trivium – симетричний алгоритм синхронного потокового шифрування, орієнтований, в першу чергу, на апаратну реалізацію з гнучкою рівновагою між швидкістю роботи та кількістю елементів, що має можливість досить ефективної програмної реалізації. Особливістю даного шифру є те, що він може обробляти одночасно до 64 біт вхідного повідомлення з 80-бітним ключем і 80-бітним вектором ініціалізації. Початковий стан Trivium демонструє собою 3 зсувні регістри сумарної довжини в 288 біт. Кожен такт відбувається зміною бітів у регістрах зсуву шляхом нелінійної комбінації прямого і зворотного зв’язку. Для ініціалізації шифру ключ та вектор записуються в 2 з 3 регістрів. Потім стан змінюється протягом 4 повних циклів, тобто протягом 1152 тактів, під час цього етапу шифр не генерує ключовий потік, лише перемішує дані всередині регістрів. Цей етап потрібний для того, щоб кожний біт в початковому стані залежав від усіх бітів ключа і вектора ініціалізації, цей процес називається дифузія. Стандартна реалізація алгоритму потребує наявності щонайменше 3488 логічних елементів, що змінюють 1 біт потоку даних за 1 такт. Для реалізації моделі під час якої буде за 1 такт виконуватись зміна 64 бітів даних, необхідно збільшити кількість логічних елементів до 5504. Це самий простий шифр з проєкту eSTREAM, який показує найкращі показники з

криптостійкості. Також Trivium включений в стандарт ISO/IEC 29192-3 в якості легковагового потокового шифру [13].

Одним з передових потокових шифрів, на даний час, вважається Salsa20 [7]. Алгоритм був представлений на конкурсі eSTREAM, метою якого було створення європейських стандартів для шифрування даних, переданих поштовими системами. Алгоритм став переможцем в першому профілі, потокових шифрів для програмного застосування з великою пропускною здатністю. Шифр Salsa20, використовує лише 3 операції: додавання 32-бітних чисел, операцію XOR та зсув бітів.

Наступним прикладом алгоритму потокового шифрування на основі квазігруп є Edon80 [14]. Edon80 – апаратний потоковий шифр, поданий на останній стадії проєкту eSTREAM. За рахунок своєї реалізації Edon80 потребує лише 2922 логічних елементи і може легко масштабуватись. Алгоритм постійно вдосконалювався, відтак у 2007 році була створена версія, що мала можливість працювати з комп'ютерами MAC. Також були розроблені інші алгоритми сімейста Edon на основі квазігруп [15], а саме Edon-R та Edon-L, алгоритми хешування з різними довжинами на виході, використовують квазігрупові тотожності [18].

Згодом, була проведена спроба атаки на алгоритм, під час якої Edon80 розглядали, як набір бінарних обчислень. Під час атаки були виявлені певні залежності в циклічності ключового потоку, що дозволяє відновити ключ з складністю 2^{69}. Це найдосконаліша атака на Edon80 із конкретними налаштуваннями, що показують, як відновити секретний ключ. Додавання лише декількох додаткових перетворень у ланцюг з 80 не змінить ситуацію, але якщо подвоїти кількість перетворень до 160, то це було б чудовим варіантом, щоб протидіяти атаці. Проте така модифікація збільшує апаратну складність алгоритму принаймні вдвічі, що призводить до втрати актуальності алгоритму.

Також, слід розглянути потоковий шифр малоресурсної криптографії розроблений спільно з студентами, у ході виконання бакалаврської дипломної роботи, LKPR [16-17]. Даний алгоритм з довжиною ключа 64 та VI 32

розроблений з метою зменшити апаратну складність потокового шифру на основі середніх CIP-квазігруп 4-го порядку, з функцією оборотності x^2 серед ізотопів групи Кляйна [20].

Самосинхронізовані потокові шифри залежать від попередніх n бітів зашифрованого тексту. Це дозволяє шифру повторно синхронізуватися, якщо біти втрачаються або додаються, що робить його стійким до помилок передачі.

Розглянемо порівняльну таблицю алгоритмів потокового шифрування, таблиця 1.1 [17].

Таблиця 1.1 – Таблиця порівняння

Назва алгоритму	Розмір ключа	Тактів ініціалізації	Складність (GE)
Grain	80/120	160	1294/1857
Trivium	80	80	749
Mickey	80/128	80/128	3188/5039
LKRP	64	32	483
Edon80	80/128	160	600/1110
Власний	64	32	427

1.3 Квазігруповий математичний апарат

Насамперед, варто відмітити, що таке квазігрупа. Квазігрупа – це алгебрична структура в абстрактній алгебрі, яка подібна до групи тим, що в ній завжди можливе ділення, але інших властивостей групи квазігрупа немає, зокрема асоціативності. Основні відміності групи від квазігрупи полягають в тому, що в квазігрупі не всі елементи мають обернений елемент. Для деяких елементів може не існувати такого елементу, з яким їх можна поєднати, щоб отримати одиничний елемент. В той же час можуть існувати комбінації при яких отримують однаковий результат, маючи рівносильні квазігрупові тотожності [19].

Квазігрупою називається групоїд $(Q;\cdot)$ такий, що система рівнянь $a \cdot x = b$, $y \cdot a = b$ має єдиний розвязок для довільних $a,b \in Q$.

Квазігрупою називається універсальна алгебра $(Q;\cdot;/;\backslash)$ сигнатури $(2;2;2)$, що задовольняє такі тотожності для будь-яких $x, y \in Q$

$$x\cdot(x \backslash y) = y, \qquad (y / x)\cdot x = y, \qquad x \backslash (x\cdot y) = y,$$

де Q — бінарний групоїд, з визначеним на ньому набором бінарних операцій $(\cdot,/,\backslash)$, де символом $(/)$ позначають ліве ділення головної операції $(\cdot)$, а символом $(\backslash)$ — праве ділення головної операції $(\cdot)$.

Також, інші означення квазігрупи доповнюють дане означення за допомогою тотожностей:

$$(y\cdot x) / x = y, \qquad x / (y \backslash x) = y, \qquad (x / y) \backslash x = y.$$

З властивостей, що попередньо розглядались, випливає, що операціям на квазігрупах невластиве поняття асоціативності, що відрізняє їх від інших алгебричних структур, таких як групи, де операція асоціативна[20].

Інакше кажучи, бінарна квазігрупа – це впорядкована пара *(Q;f)*, де *Q* – множина, *f* – бінарна операція, що призначена на цій множині, така, що кожне з рівнянь:

$$f(a, y)=b, \quad f(x, a)=b,$$

однозначно розв’язне для будь-якої пари елементів *a*, *b* із множини *Q*. Вивчення алгебричної теорії бінарних квазігруп займались Брук, Білоусов, Крапєж та багато інших [21].

Також, бінарні квазігрупи можна класифікувати за типом парастрофної симетрії, всього таких симетрій існує 6: тотально-симетричні, напів-симетричні, асиметричні, комутативні, ліво-симетричні, право-симетричні. Тотально-симетричні це, коли всі парастрофи, тобто оборотні функціх їх збігаються, тоді в результаті перетворень маємо один латинський квадрат. Напів-симетричні, це коли є два різних парастрофи, а в результаті перетворень отримуємо, два різних латинських квадрати. Комутативні квазігрупи описуються тотожністю:

$$xy=yx.$$

Ліво-симетричні квазігрупи, якщо виконується таку тотожність:

$$x\cdot xy=y$$

Право-симетричні квазігрупи, відповідно, якщо виконується така тотожність:

$$xy \cdot y = x.$$

Розглядаючи квазігрупи, можна виділити ряд основних функцій, властивостей, що діють над квазігрупами, наприклад: парастрофні перетворення, симетричність, інвертованість, асоціативність, бісиметрія, та інші. Функцію називають квазігруповою або оборотною, якщо вона оборотна по кожній своїй змінній, тобто вона є ліво-оборотною і право-оборотною одночасно при цьому тотожності, які описуються квазігрупами, називають визначальними.

Квазігрупа, яка має двосторонній одиничний елемент, називають лупою. Лупа – це квазігрупа, для якої існує такий елемент e, що виконується така тотожність $a \cdot e = e \cdot a = a$ для $\forall a \in Q$.

Дві квазігрупи $(A; *)$ та $(B; o)$ називають ізотопними, якщо існують такі бієкції α, β, γ із множини A на множину B, що задовольняє співвідношення

$$\alpha(x) o \beta(\gamma) = \gamma(x * \gamma)$$

для всіх x, $\gamma \in A$, трійку (α, β, γ) називають ізотопізмом. Ізотопію називають головною, якщо $\gamma = l$, а квазігрупи − ізоморфними, коли $\alpha = \beta = \gamma$. І відношення ізотопії, і відношення ізоморфізму є відношеннями еквівалентності, тому їм відповідають розбиття множини всіх оборотних операцій, що визначені на одному й тому ж носієві. Кожен групоїд, будучи ізотопним деякій квазігрупі, також є квазігрупою. Кожна квазігрупа, ізотопна лупі, з наперед заданим нейтральним елементом.

Відмітимо відомий результат, який подано в зручному вигляді – таблиці Келі. Названа на честь британського математика 19-го століття Артура Келі, таблиця Келі описує структуру скінченної групи шляхом розміщення всіх можливих продуктів усіх елементів групи в квадратній таблиці, що нагадує таблицю додавання або множення [18].

Багато властивостей групи – наприклад, чи є вона абелевою чи ні, які елементи є оберненими до яких елементів, а також розмір і вміст центру групи – можна виявити з її таблиці Келі.

Оскільки багато таблиць Келі описують групи, які не є абелевими, добуток ab відносно двійкової операції групи не гарантовано дорівнюватиме добутку ba для всіх a і b у групі. Щоб уникнути плутанини, прийнято, що фактор, що позначає рядок, названий Келі ближчим фактором, стоїть першим, а фактор, який позначає стовпець або наступний фактор, є другим. Наприклад, перетин рядка a та стовпця b є ab, але не є ba.

Розглянемо властивості та використання таблиць Келі.

Таблиця Келі повідомляє нам, чи є група абелевою, коли таблиця симетрична відносно головної діагоналі. Оскільки групова операція абелевої групи комутативна, група є абелевою тоді і тільки тоді, коли значення її таблиці Кейлі симетричні вздовж її діагональної осі. Наведена вище група і циклічна група порядку 3 при звичайному множенні є прикладами абелевих груп, і перевірка симетрії їхніх таблиць Келі це підтверджує. Навпаки, найменша неабелева група, двогранна група порядку 6, не має симетричної таблиці Келі.

Оскільки асоціативність сприймається як аксіома при роботі з групами, вона часто сприймається як належне при роботі з таблицями Келі. Проте, таблиці Келі, також, можна використовувати для характеристики роботи квазігрупи, що не передбачає асоціативність як аксіому. На жаль, зазвичай неможливо визначити, чи є операція асоціативною, просто глянувши на її таблицю Келі, як це відбувається з комутативністю. Це тому, що асоціативність залежить від даної тотожності, $(ab)c=a(bc)$, а таблиця Келі відображає лише результати операцій, лише між двома змінними.

Оскільки властивість скорочення виконується для груп (і навіть квазігруп), жоден рядок або стовпець таблиці Келі не може мати один і той же елемент двічі. Таким чином, кожен рядок і стовпець таблиці є перестановкою всіх елементів у квазігрупі. Це значно обмежує таблиці Келі, що можуть визначати дійсну квазігрупову операцію.

Щоб зрозуміти, чому рядок або стовпець не може мати один і той же елемент більше одного разу, нехай *a, x* і *y* є елементами квазігрупи, де *x* і *y* різні. Тоді в рядку, що представляє елемент *a*, стовпець, що відповідає *x,* має добуток *ax*, і аналогічно стовпець, що відповідає *y*, має добуток *ay*. Якби ці два добутки були рівними – тобто рядок *a* мав би один і той же елемент двічі, наша гіпотеза – тоді *ax* дорівнював би *ay*. Але оскільки виконується властивість скорочення, можемо зробити висновок, що якщо *ax = ay*, то (*x = y)*, суперечить припущенню.

Отже, дана гіпотеза хибна, і рядок не може мати один і той же елемент двічі. Точно такого ж аргументу достатньо, щоб довести випадок стовпця, і тому приходимо до висновку, що кожен рядок і стовпець не може мати елемент більше одного разу.

Оскільки група скінченна, принцип розподілу гарантує, що кожен елемент квазігрупи буде наведено в кожному рядку та в кожному стовпці рівно один раз.

Таким чином, таблиця Кейлі квазігрупи є прикладом латинського квадрата. Альтернативний і більш стислий доказ випливає з властивості скасування. Ця властивість передбачає, що для кожного *x* в квазігрупі функція однієї змінної *y f(x,y)= xy* має бути однозначним відображенням.

Інакше кажучи, латинський квадрат – квадратна матриця n×n, де в кожному рядку й стовпці є різні елементи з множини значень від 1 до n, наприклад квазігрупа 8-го порядку, є латинським квадратом 8×8. Ця особливість й описує основну властивість латинських квадратів, відсутність повторень в рядках і стовпцях, приклад поданий у таблиці 1.1.

Таблиця 1.1 – Латинський квадрат 4×4.

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Існує декілька методів побудови латинських квадратів, а саме: метод

Боуєрса, метод Ітані, також існують методи на основі блокового проектування та комбінаторних структур.

1.4 Висновки до розділу

На основі проаналізованих джерел інформації та огляду предметної області було сформовано висновки.

Можна з дійти до висновку, що галузь малоресурсної криптографії є надзвичайно важливою в сучасному світі за рахунок стрімкого розвитку технологій та тенденції до мініатюризації пристроїв.

У випадку появи перших працездатних нейронних імплантів у найближчому майбутньому виникне серйозна потреба у впровадженні ефективних механізмів захисту даних, які зберігаються та обробляються всередині таких пристроїв. Це підкреслює актуальність завчасної розробки криптографічних рішень, орієнтованих на середовища з обмеженими ресурсами.

Саме таку нішу займає напрям LW-криптографія. Його мета — створення захисних алгоритмів і протоколів, включаючи механізми автентифікації, хеш-функції та методи шифрування, що можуть функціонувати з мінімальним навантаженням на апаратну частину системи, споживаючи обмежену кількість енергії, зберігаючи компактність реалізації та високу продуктивність.

2 МЕТОД ШИФРУВАННЯ НА ОСНОВІ ЛАТИНСЬКИХ КВАДРАТІВ

Даний розділ призначений для опису латинських квадратів 4-го порядку та створення засобу потокового шифрування на основі квазігруп. Тут описано математичні методи для опису квазігруп 4-го порядку та зіставлення логічних формул.

2.1 Математичне підгрунття для опису латинських квадратів 4-го порядку

В даному параграфі роботи було досліджено квазігрупи 4-го порядку, які є лупами, тобто квазігрупами, що мають нейтральний елемент. При позначені нейтрального елемента через 0, її називаємо 0-лупою. Всього 0-луп четвертого порядку існує 4, одна з них група Кляйна, інші три ізоморфні циклічні групи, дані лупи наведені у таблиці 2.1 [23].

Таблиці 2.1 – 0-лупи четвертого порядку

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

⊕	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	1	0
3	3	2	0	1

◇	0	1	2	3
0	0	1	2	3
1	1	3	0	2
2	2	0	3	1
3	3	2	1	0

Кожна квазігрупа може створюватись за допомогою двох логічних виразів, тобто кожна квазігрупа L_n реалізується за допомогою двох логічних виразів: L_nz_1 та L_nz_2. Для ефективної реалізації квазігрупи в засобі потокового шифрування LW-криптографії необхідно провести спрощення логічних виразів перед їх апаратною реалізацією. Виконаємо побудову таблиць істинності для заданих нейтральних квазігруп. Для початку виконаємо подання таблиць в двійковій системі числення, де: 0-00; 1-01; 2-10; 3-11. З отриманих значень

отримуємо, в якій виконаємо необхідні перетворення для створення таблиці в 2-системі числення, таблиця 2.2.

Таблиця 2.2 – 0-лупи четвертого порядку в 2-системі числення

+	00	01	10	11
00	00	01	10	11
01	01	10	11	00
10	10	11	00	01
11	11	00	01	10

⊕	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

*	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	01	00
11	11	10	00	01

◇	00	01	10	11
00	00	01	10	11
01	01	11	00	10
10	10	00	11	01
11	11	10	01	00

Наступним кроком буде створення таблиці істинності, для кожної з наведених вище в таблиці луп. Розглянемо на рисунку 2.1, де показане створення таблиці істинності з 6 елементів.

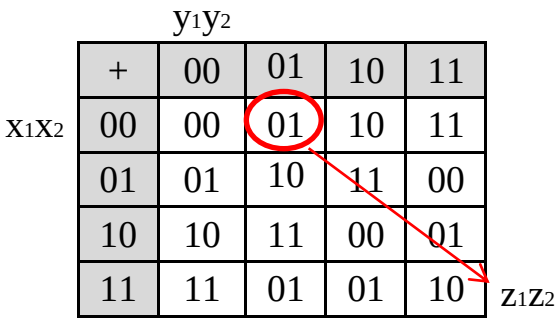


Рисунок 2.1 – Отримання значення для таблиці істинності

Елементи x_1x_2 в даному прикладі мають приймати перший і другий елемент відповідно, тобто $x_1=0$ та $x_2=0$, y_1 та y_2 будуть приймати наступні значення, $y_1=0$ та $y_2=1$. Значення, які отримуємо, будуть заповнюватись в z_1 та

z_2, отримаємо $z_1=0$ $z_2=1$. Отримані значення записані до таблиці істиності кожної з 4 квазігруп, в результаті отримаємо таблицю 2.3 [24].

Таблиця 2.3 – Таблиця істинності квазігруп *(Q;+), (Q; ⊕)*

+	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	0	0
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	0	0
14	1	1	1	0	0	1
15	1	1	1	1	1	0

⊕	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	1
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	1	0
14	1	1	1	0	0	1
15	1	1	1	1	0	0

Відповідно до кожної з 0-луп 4-го порядку з таблиці 2.2 будуємо таблиці істинності для подальших якісних обчислень мінімізації логічних формул. Таблиці істинності квазігруп *(Q;+), (Q; ⊕)* наведені в таблиці таблиці 2.3, в таблиці 2.4 побудовані для квазігруп *(Q; *), (Q; ◇)*.

Таблиця 2.4 – Таблиця істинності квазігруп *(Q; *), (Q; ◇)*

⊕	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	1

◇	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	1	1
6	0	1	1	0	0	0

Продовження таблиці 2.4

7	0	1	1	1	1	0	7	0	1	1	1	1	0
8	1	0	0	0	1	0	8	1	0	0	0	1	0
9	1	0	0	1	1	1	9	1	0	0	1	0	0
10	1	0	1	0	0	1	10	1	0	1	0	1	1
11	1	0	1	1	0	0	11	1	0	1	1	0	1
12	1	1	0	0	1	1	12	1	1	0	0	1	1
13	1	1	0	1	1	0	13	1	1	0	1	1	0
14	1	1	1	0	0	0	14	1	1	1	0	0	1
15	1	1	1	1	0	1	15	1	1	1	1	0	0

Одним із способів подання булевих функцій від невеликої кількості змінних є діаграми Вейча. Метод дозволяє швидко мати мінімальні диз'юнктивні нормальні форми булевої функції від невеликої кількості змінних. В клітинці діаграми Вейча ставиться одиниця, якщо булева функція набуває істинного значення на відповідному наборі. Нульові значення булевої функції в діаграмі Вейча не проставляються [26].

Для даного дослідження, діаграма булевої функції чотирьох змінних подана на рисунку 2.2.

	$X2$	$X2$	$\overline{X2}$	$\overline{X2}$	
$X1$	1100	1101	1001	1000	$\overline{Y1}$
$X1$	1110	1111	1011	1010	$Y1$
$\overline{X1}$	0110	0111	0011	0010	$Y1$
$\overline{X1}$	0100	0101	0001	0000	$\overline{Y1}$
	$\overline{Y2}$	$Y2$	$Y2$	$\overline{Y2}$	

Рисунок 2.2 – Діаграма Вейча чотирьох змінних

З таблиць істинності 0-луп, маємо такі набори істиності векторних значень, що покривають одиницями.

Для квазігрупової операції (+) визначено такі векторні набори істинності :

z_1=V(2;3;5;6;8;9;12;15); z_2=V(1;3;4;6;9;11;12;14).

Для квазігрупової операції $\oplus$ визначено такі векторні набори істинності :

z_1=V(2;3;6;7;8;9;12;13); z_2=V(1;3;4;6;9;11;12;14).

Для квазігрупової операції (*) визначено такі векторні набори істинності:

z_1=V(2;3;6;7;8;9;12;13); z_2=V(1;3;4;6;9;10;12;15).

Для квазігрупової операції ($\diamond$) визначено такі векторні набори істинності:

z_1=V(2;3;5;7;8;10;12;13); z_2=V(1;3;4;5;10;11;12;14).

З наведених наборів реалізуємо карти Карно 0-луп 4-го порядку. Для цього будується карта Карно для z_1;z_2, що подана на рисунку 2.3.

	$X2$	$X2$	$\overline{X2}$	$\overline{X2}$	
$X1$	12	13	9	8	$\overline{Y1}$
$X1$	14	15	11	10	$Y1$
$\overline{X1}$	6	7	3	2	$Y1$
$\overline{X1}$	4	5	1	0	$\overline{Y1}$
	$\overline{Y2}$	$Y2$	$Y2$	$\overline{Y2}$	

Рисунок 2.3 – Карта Карно для чотирьох змінних.

Для першої квазігрупи, таблиця істинності якої вказана в таблиці 2.3, пара карт Карно елементів z_1 та z_2 наведена в [24] і подана на рисунку 2.4.

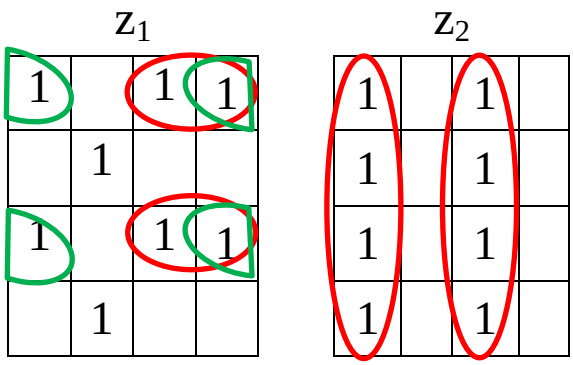


Рисунок 2.4 – Карта Карно для елементів квазігрупи *(Q;+)*

Наступним кроком реалізується групування одиниць, важливими аспектами під час групування одиниць є те, що блоки мають бути прямокутним і в степені двійки, тобто: 1, 2, 4, 8 і так далі. З отриманих значень визначаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Враховуючи Картик Карно елементів z_1 та z_2 квазігрупи *(Q;+)* з рисунку 2.4 логічні вирази мають вигляд:

$$z_1 = x_1\overline{y_1y_2} \vee x_1\overline{x_2y_1} \vee x_1x_2y_1y_2 \vee \overline{x_1y_2}y_1 \vee \overline{x_1x_2}y_1 \vee \overline{x_1y_1}x_2y_2$$

$$z_2 = x_2\overline{y_2} \vee \overline{x_2}y_2$$

Для подальшої роботи необхідно спростити отримані логічні вирази. Для цього спочатку винесемо спільні елементи за дужки і скористаємось законами логіки, а саме $a\bar{b} \vee \bar{b}a \equiv a \oplus b$ та $ab \vee \bar{a}\bar{b} \equiv \overline{a \oplus b}$:

$$z_1 = x_1\overline{y_1y_2} \vee x_1\overline{x_2y_1} \vee x_1x_2y_1y_2 \vee \overline{x_1y_2}y_1 \vee \overline{x_1x_2}y_1 \vee \overline{x_1y_1}x_2y_2 =$$

$$= \overline{y_2}(x_1\overline{y_1} \vee \overline{x_1}y_1) \vee \overline{x_2}(x_1\overline{y_1} \vee \overline{x_1}y_1) \vee x_2y_2(x_1y_1 \vee \overline{x_1y_1}) =$$

$$= \overline{y_2}(x_1 \oplus y_1) \vee \overline{x_2}(x_1 \oplus y_1) \vee x_2y_2\overline{(x_1 \oplus y_1)} =$$

$$= x_2y_2 \oplus (x_1 \oplus y_1)$$

$$z_2 = x_2\overline{y_2} \vee \overline{x_2}y_2 = x_2 \oplus y_2$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_0 = (z_1; z_2) = (x_2y_2 \oplus (x_1 \oplus y_1); x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обчислити апаратну складність засобу.

Квазігрупа L_1 реалізується відповідно сумою значень двох значень логічних виразів, які отримуються з таблиці істинності 2.3.

З пар карт Карно елементи z_1 та z_2 реалізується відповідно на рисунку 2.5.

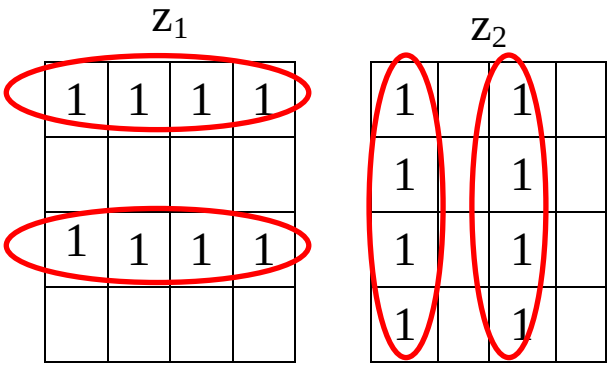


Рисунок 2.5 – Карта Карно для елементів квазігрупи *(Q; ⊕)*

З отриманих значень отримаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що ми отримали мають вигляд:

$$z_1 = x_1\overline{y_1} \vee \overline{x_1}y_1 = x_1 \oplus y_1$$

$$z_2 = x_2\overline{y_2} \vee \overline{x_2}y_2 = x_2 \oplus y_2$$

$$L_1 = (z_1; z_2) = (x_1 \oplus y_1; x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна пораховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 2 елементи XOR.

Квазігрупа L_2 реалізується відповідно сумою значень двох логічних виразів, які отримуються з таблиці істинності 2.4 для операції *(Q;∗)* .

З пар карт Карно елементи z_1 та z_2 реалізується відповідно на рисунку 2.6.

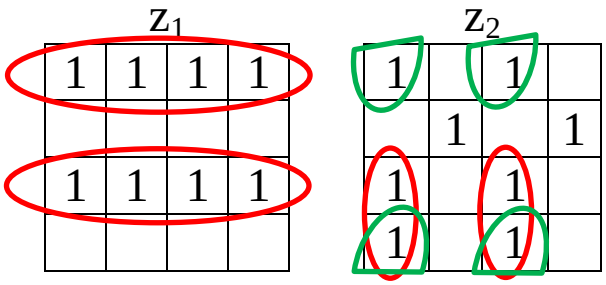


Рисунок 2.6 – Карта Карно для елементів квазігрупи *(Q;∗)*

З отриманих значень отримаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що ми отримали мають вигляд:

$$z_1 = x_1\overline{y_1} \vee \overline{x_1}y_1 = x_1 \oplus y_1$$

$$z_2 = x_2\overline{y_1y_2} \vee \overline{x_1}x_2\overline{y_1} \vee x_1x_2y_1y_2 \vee \overline{x_2y_1}y_2 \vee x_1\overline{x_2}y_1\overline{y_2}$$

Для подальшої роботи спрощуємо отримані вирази:

$$z_2 = x_2\overline{y_1y_2} \vee \overline{x_1}x_2\overline{y_1} \vee x_1x_2y_1y_2 \vee \overline{x_2y_1}y_2 \vee x_1\overline{x_2}y_1\overline{y_2} =$$

$$= \overline{y_1}(x_2\overline{y_2} \vee \overline{x_2}y_2) \vee \overline{x_1}(x_2\overline{y_2} \vee \overline{x_2}y_2) \vee x_1y_1(x_2y_2 \vee \overline{x_2y_2}) =$$

$$= (\overline{y_1} \vee \overline{x_1})(x_2 \oplus y_2) \vee x_1y_1\overline{(x_2 \oplus y_2)} = x_1y_1 \oplus (x_2 \oplus y_2)$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_2 = (z_1; z_2) = (x_1 \oplus y_1; x_1y_1 \oplus (x_2 \oplus y_2))$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна знайти апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 3 елементи XOR та 1 логічне I.

Квазігрупа L_3 реалізується відповідно сумою значень двох логічних виразів, які отримуються з таблиці істинності 2.4.

З пар карт Карно елементи z_1 та z_2 реалізуються на рисунку 2.7.

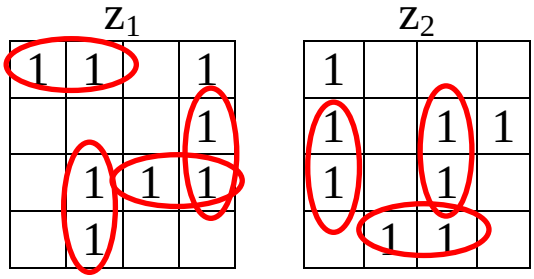


Рисунок 2.7 – Карта Карно для елементів квазігрупи *(Q;◇)*

З отриманих значень отримаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що ми отримали мають вигляд:

$$z_1 = x_1x_2\overline{y_1} \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1 \vee \overline{x_1}x_2y_2$$

$$z_2 = x_1x_2\overline{y_2} \vee x_1\overline{x_2}y_1 \vee \overline{x_1}x_2\overline{y_1}$$

$$L_3 = (z_1; z_2) = (x_1x_2\overline{y_1} \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1 \vee \overline{x_1}x_2y_2; x_1x_2\overline{y_2} \vee x_1\overline{x_2}y_1 \vee \overline{x_1}x_2\overline{y_1})$$

В результаті логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 5 елементів АБО, 10 логічних НІ та 14 логічних І для за шифрування вхідного потоку даних.

Попередні обчислення виконувались для операції на етапі зашифрування, подальші виконані обчислення будуть виконуватись для операції розшифрування, в ході яких отримаємо ще 4 логічних вирази.

Для зашифрування повідомлення використовувались чотири 0-лупи 4-го порядку, тобто квазігрупи, що мають нейтральний елемент, для розшифрування будуть використовуватись оборотні квазігрупи, тобто квазігрупи для яких існує обернена операція таблиця 2.5.

Таблиця 2.5 – Оборотні 0-лупи четвертого порядку

L0	0	1	2	3
0	0	3	2	1
1	1	0	3	2
2	2	1	0	3
3	3	2	1	0

L1	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Продовження таблиці 2.5

L2	0	1	2	3
0	0	1	3	2
1	1	0	2	3
2	2	3	0	1
3	3	2	1	0

L3	0	1	2	3
0	0	2	1	3
1	1	0	3	2
2	2	3	0	1
3	3	1	2	0

Кожна квазігрупа може створюватись за допомогою двох логічних виразів, тобто кожна квазігрупа L_n реалізується за допомогою двох логічних виразів: $L_n(z_1;z_2)$. Виконаємо побудову таблиць істинності для знайдених нейтральних квазігруп. Для початку виконаємо подання таблиць в двійковій системі числення, де: 0-00; 1-01; 2-10; 3-11. З отриманих значень отримуємо,в якій виконаємо необхідні перетворення для створення таблиці в 2-системі числення, таблиця 2.6.

Таблиця 2.6 – 0-лупи четвертого порядку в 2-системі числення

$^l+$	00	01	10	11
00	00	11	10	01
01	01	00	11	10
10	10	01	00	11
11	11	10	01	00

$^l\oplus$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$^l*$	00	01	10	11
00	00	01	11	10
01	01	00	10	11
10	10	11	00	01
11	11	10	10	00

$^l\diamond$	00	01	10	11
00	00	10	01	11
01	01	00	11	10
10	10	11	00	10
11	11	01	10	00

Наступним кроком буде створення таблиці істинності, що реалізується аналогічно до рисунку 2.1, з якого отримаємо першу таблицю істинності, таблиця 2.7.

Таблиця 2.7 – Таблиця істинності квазігруп $(Q;{}^{l}+)$, $(Q;{}^{l}\oplus)$

${}^{l}+$	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	0	1	0
3	0	0	1	1	0	1
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	1
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	0	1
10	1	0	1	0	0	0
11	1	0	1	1	1	1
12	1	1	0	0	1	1
13	1	1	0	1	1	0
14	1	1	1	0	0	1
15	1	1	1	1	0	0

${}^{l}\oplus$	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	1
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	1	0
14	1	1	1	0	0	1
15	1	1	1	1	0	0

Таблиця 2.8 – Таблиця істинності квазігруп $(Q;{}^{l}*)$, $(Q;{}^{l}\diamond)$

${}^{l}*$	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	1
3	0	0	1	1	1	0
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	0
7	0	1	1	1	1	1
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	1	0
14	1	1	1	0	0	1
15	1	1	1	1	0	0

${}^{l}\diamond$	x_1	x_2	y_1	y_2	z_1	z_2
0	0	0	0	0	0	0
1	0	0	0	1	1	0
2	0	0	1	0	0	1
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	0	0
6	0	1	1	0	1	1
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	0	1
14	1	1	1	0	1	0
15	1	1	1	1	0	0

В ході виконання даних дій отримаємо теж чотири таблиці істинності з яких отримаємо набори векторних значень, що покривають одиниці.

Для квазігрупової операції (${}^{l}+$) визначено такі векторні набори істинності:

z_1=V(1;2;6;7;8;11;12;13); z_2=V(1;3;4;6;9;11;12;14).

Для квазігрупової операції (${}^{l}\oplus$) всі парастрофи, тобто оборотні, операція квазігрупи *(Q;* ${}^{l}\oplus$*)* однакові. Це означає, що таблиці Келі одна й та сама, тому обчислення апаратної складності в операційному блоці опускається на кроці розшифрування.

Для квазігрупової операції (${}^{l}*$) визначено такі векторні набори істинності:

z_1=V(2;3;6;7;8;9;12;13); z_2=V(1;2;4;7;9;11;12;14).

Для квазігрупової операції (${}^{l}\diamond$) визначено такі векторні набори істинності:

z_1=V(1;3;6;7;8;9;12;14); z_2=V(2;3;4;6;9;11;12;13).

З отриманих значень з таблиці істинності 2.7, отримаємо векторні значення для оборотної квазігрупи (${}^{l}+$), згідно яких можна реалізувати функцію. Кожна квазігрупа описується двома логічними виразами, тому для кожного логічного виразу і його таблиці істинності будуються карти Карно. Для оборотної операції (${}^{l}+$) будуються такі такі карти Карно на рисунку 2.8:

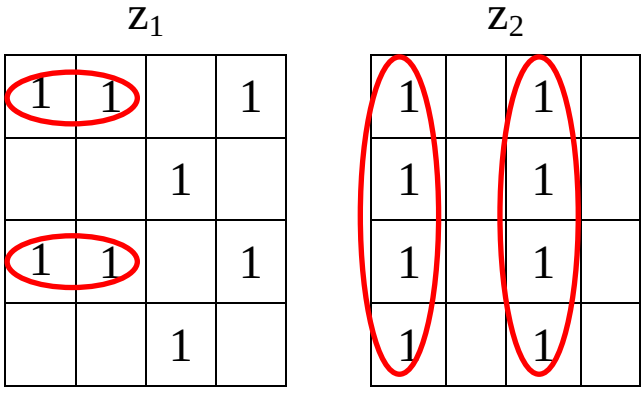


Рисунок 2.8 – Карта Карно для елементів квазігрупи *(Q;* (${}^{l}+$*))*

Наступним кроком реалізується групування одиниць, важливими аспектами під час групування одиниць є те, що блоки мають бути прямокутним і в степені двійки, тобто: 1, 2, 4, 8 і так далі. З отриманих значень отримаємо

логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що отримали мають вигляд:

$$z_1 = x_1x_2\overline{y_2} \vee x_1\overline{x_2}\,\overline{y_1y_2} \vee x_1\overline{x_2}y_1y_2 \vee \overline{x_1}x_2y_1 \vee \overline{x_1x_2}y_1\overline{y_2} \vee \overline{x_1x_2y_1}y_2$$

$$z_2 = x_2\overline{y_2} \vee \overline{x_2}y_2$$

Для подальшої роботи необхідно спростити дані вирази:

$$z_1 = x_2(x_1 \oplus y_1) \vee \overline{x_2}(x_1 \oplus (y_1 \oplus y_2)$$

$$z_2 = x_2\overline{y_2} \vee \overline{x_2}y_2 = x_2 \oplus y_2$$

$$L_0 = (z_1; z_2) = (x_2(x_1 \oplus y_1) + \overline{x_2}(x_1 \oplus (y_1 \oplus y_2); x_2 \oplus y_2)$$

В результаті логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 1 елемент АБО, 1 логічне НІ, 2 логічних І та 4 операції XOR для розшифрування вхідного потоку даних.

Для наступної квазігрупи з операцією (^{l}*) відповідно реалізовується таблиця істинності, таблиця 2.8.

З отриманих значень з таблиці істинності 2.8, отримаємо векторні значення для оберненої квазігрупи з операцією (^{l}*), згідно яких можна реалізувати функцію. Кожна квазігрупа описується двома логічними виразами, тому для кожного логічного виразу і його таблиці істинності будуються діаграми Вейча. Для операції (^{l}*) будуються наступні Карти Карно, що зображені на рисунку 2.9:

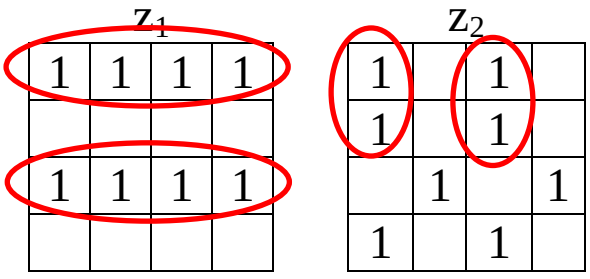


Рисунок 2.9 – Карта Карно для елементів квазігрупи *(Q; (^{l}*))*

З отриманих значень отримаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що ми отримали мають вигляд:

$$z_1 = x_1\overline{y_1} \vee \overline{x_1}y_1 = x_1 \oplus y_1$$
$$z_2 = x_1(x_2 \oplus y_2 + \overline{x_1}(y_1 \oplus (x_2 \oplus y_2)$$
$$L_1 = (z_1; z_2) = (x_1 \oplus y_1; x_1(x_2 \oplus y_2 \vee \overline{x_1}(y_1 \oplus (x_2 \oplus y_2))$$

В результаті логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 1 елемент АБО, 4 операції XOR та 1 операцію НІ для розшифрування вхідного потоку даних.

Квазігрупа L_3 реалізується відповідно сумою значень двох логічних виразів, які отримуються з таблиці істинності 2.8.

З пар карт Карно елементи z_1 та z_2 реалізуються наступним чином, рисунок 2.10.

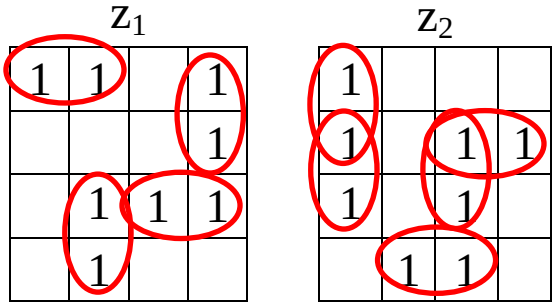


Рисунок 2.10 – Карта Карно для елементів квазігрупи *(Q; (l◇))*

З отриманих значень отримаємо логічну функцію у вигляді диз'юнктивної нормальної форми. Логічні вирази, що ми отримали мають вигляд:

$$z_1 = x_1x_2\overline{y_2} \vee x_1x_2\overline{y_1} \vee x_1x_2y_2 \vee \overline{x_1x_2}y_2$$
$$z_2 = x_1x_2\overline{y_1} \vee x_1\overline{x_2}y_2 \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1$$
$$L_1 = (z_1; z_2) = (x_1x_2\overline{y_2} \vee x_1x_2\overline{y_1} \vee x_1x_2y_2 \vee \overline{x_1x_2}y_2;$$
$$x_1x_2\overline{y_1} \vee x_1\overline{x_2}y_2 \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1)$$

В результаті логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 6 елементів АБО, 12 логічних НІ та 10 логічних І для розшифрування вхідного потоку даних.

Отже, в ході виконання даних обчислень отримали 8 логічних виразів, складність яких, необхідно враховувати під час обчислень апаратної складності даного засобу шифрування. Кожний вираз передає свій сигнал елемента мультиплексор, який буде обробляти вихідний сигнал і формувати потік шифротексту в бітовому представлені.

2.2 Розробка методу потокового шифрування

Для розробки методу потокового шифрування потрібно створити його повний опис, що описує роботу засобу потокового шифрування. Алгоритм, що розглядається має два типи роботи: за шифрування та розшифрування.

Алгоритм роботи за шифрування та розшифрування.

Першим кроком роботи алгоритму є введення вхідних даних та вибір режиму роботи, ключа К довжиною 64б та повідомлення М довільної довжини.

Наступний крок алгоритму визначає, яка квазігрупа буде використовуватись для за шифрування або розшифрування даних, для цього необхідно перших два біти вхідного ключа К довжиною 64б, представити у вигляді двійкого числа, тобто “01” – 1, отриманий номер буде відповідати номеру квазігрупи, яка використовується для шифрування або розшифрування в залежності від режиму роботи алгоритму.

Третій крок полягає в за шифруванні або розшифрувані даних, для цього використовується два біти послідовності ключа К та два біти послідовності повідомлення М. Для роботи використовується 3 та 4 біти ключа К та 1-2 біти повідомлення М. Отримані біти використовуються для виконання дій з

квазігрупою, що була отримана у другому кроці. Результат, що отримали після виконання дій, надсилається на мультиплексор MUX.

Після виконання шифрування або розшифрування виконується четвертий крок, у ньому підготовлюється послідовність ключа K та повідомлення M для наступного циклу роботу методу потокового шифрування. Для забезпечення більшої криптостійкості, перша пара ключа K після виконання зашифрування або розшифрування зникає після кожного циклу роботи програми. Оскільки ключ має постійну довжину, для цього необхідно добавити до послідовності ключа K два додаткових біти, для цього використовується 3-4 біти ключа, та 1-2 біти послідовності повідомлення M, над отриманими значення виконують дії в квазігрупі L^0, отриманий результат додається в кінець послідовності ключа K, перші біти повідомлення та ключа зникають.

Алгоритм завершує роботу, якщо довжина повідомлення дорівнює 0, інакше алгоритм повертається на крок 2.

Схема алгоритму зображена на рисунку 2.11.

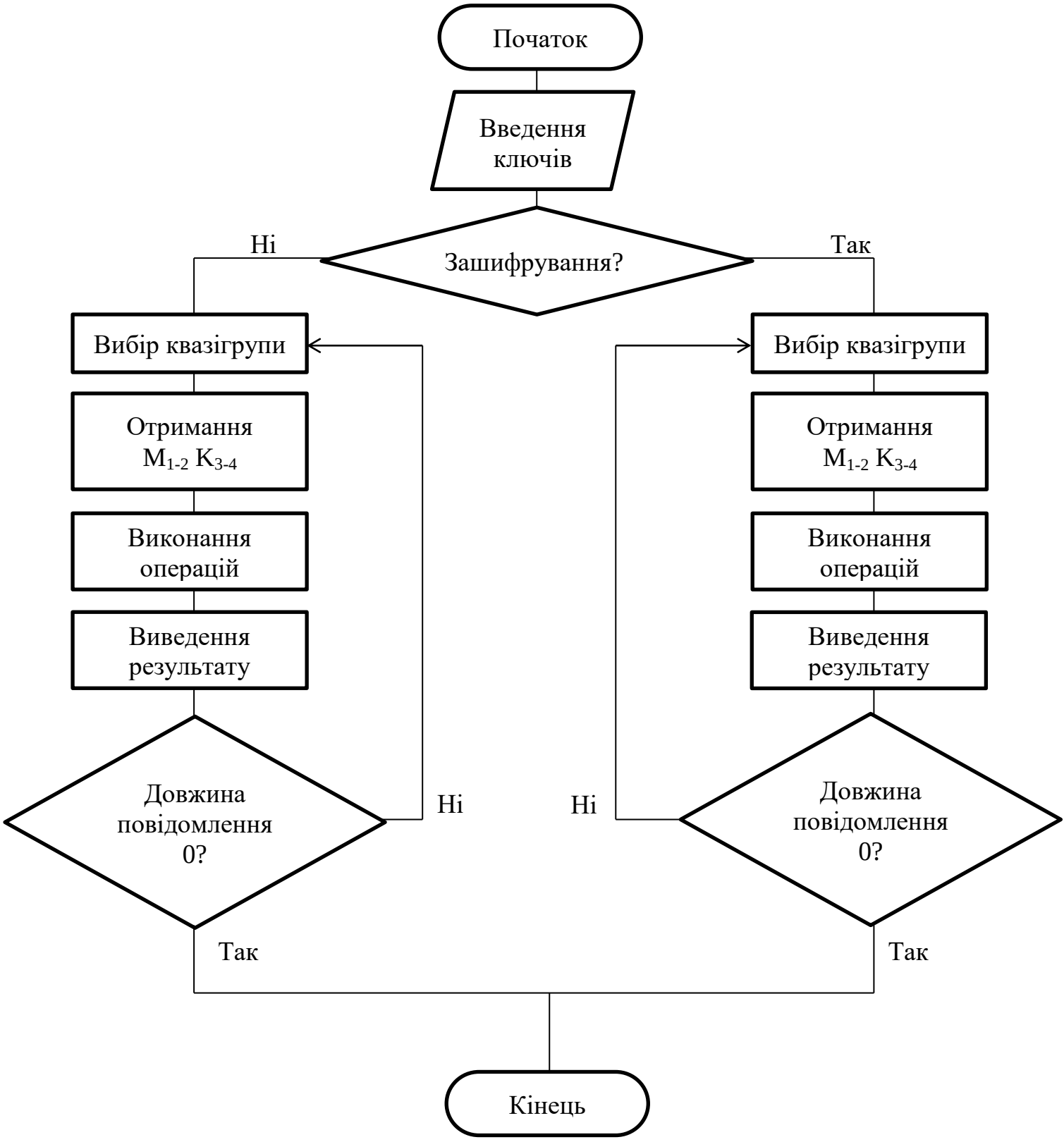


Рисунок 2.11– Алгоритм засобу потокового шифрування

2.3 Розгляд прикладу роботи засобу потокового шифрування

Для прикладу роботи засобу потокового шифрування будемо поступово виконувати кроки, які описані на етапі 2.2.

Першим кроком отримуємо послідовність ключа К, повідомлення М та обираємо режим роботи, за шифрування чи розшифрування. В ході вхідного ключа використаємо лише 8 біт, для прикладу роботи програми, для 64 біт виконуватись буде аналогічно, К=”10011011” . Вхідний потік повідомлення є довільної довжини, М=”10101111”. Режим роботи – шифрування.

Другим кроком визначається квазігрупа з якою будуть виконуватись дії. Перших два біти ключа визначають квазігрупу, $K_{0\text{-}1}$=”10”, тобто використовується квазігрупа з номером 2, дана квазігрупа подана у таблиці 2.1 з операцією (*).

Третій крок оперує всі вхідні дані та виводить результат. Для цього кроку потрібно використати третій і четвертий біт ключа та перший і другий біт повідомлення. $K_{2\text{-}3}$=”01”, $M_{0\text{-}1}$=”10”, відповідно операція вказана на рисунку 2.12.

$M_{0\text{-}1}$

$K_{2\text{-}3}$ \ +	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	01	00
11	11	10	00	01

Рисунок 2.12 – Результат обчислення

Результатом операції квазігрупи отримали значення “11”, отриманий результат надсилається на мультиплексор для подальшої обробки.

Четвертий крок визначає подальші дії програмного засобу. Для забезпечення більшої криптостійкості, перша пара ключа $K_{0\text{-}1}$ після виконання зашифрування або розшифрування зникає після кожного циклу роботи

програми. Оскільки ключ має постійну довжину, для цього необхідно добавити до послідовності ключа K два додаткових біти.

2.4 Висновки до розділу

В результаті проведеного аналізу криптографічного примітиву на основі латинських квадратів 4-порядку, що являють собою 0-лупи, було визначено, що вибрані квазігрупи за власними характеристиками підходять для використання в якості криптографічних примітивів для алгоритму потокового шифрування.

Особливістю обраних квазігруп є можливість реалізовувати метод шифрування і розшифрування одним процесом, але з використанням оборотних квазігруп. Даний аспект гарантує зменшення апаратної складності засобу при розробці засобу потокового шифрування на основі латинських квадратів.

Розроблено метод та описано алгоритм потокового шифрування, що базується на використанні операції квазігрупи для зашифрвання та операції лівого ділення для розшифрування операції.

Алгоритм реалізується в 4 кроки, трьох розгалужень та двох циклів. Даний алгоритм реалізується апаратно за допомогою 65 тригерів та 1 мультиплексора з 4 входами.

Також наведено приклад виконання за шифрування інформації.

3 РОЗРОБКА СТРУКТУРИ ТА ОЦІНКА СКЛАДНОСТІ ЗАСОБУ

У третьому розділі розглядається апаратний засіб для реалізації алгоритму потокового шифрування на основі латинських квадратів, розробці його структури, оцінці складності апаратного засобу та порівняльними оцінками сучасних засобів потокового шифрування.

3.1 Алгоритм потокового шифрування

Алгоритм потокового шифрування на основі латинських квадратів використовує для своєї роботи нейтральні лупи, тобто квазігрупи, що мають нейтральним елемент. Вхідними даними виступають ключ довжиною 64 біт та повідомлення довільної довжини.

Для забезпечення більшої криптостійкості в алгоритмі використовується додаткова квазігрупа за допомогою якої, довжина ключа залишається сталою і при кожному зашифровані чи розшифрувані, ключ залишається сталої довжини і змінює свій стан залежно від вхідних параметрів ключа та вхідного повідомлення, тобто повідомлення, ключ і квазігрупа, що використовується, генерують самі себе.

Застосування квазігруп є основною властивістю даного алгоритму шифрування. Використання 4 квазігруп для шифрування та 4 квазігруп для розшифрування збільшує криптостійкість алгоритму.

Алгоритм використовує перші біти ключа, для вибору квазігрупи за шифрування або розшифрування. Оскільки використовуються лише два біти, для визначення квазігрупи, то можливих варіантів вибору існує всього 4. Кожна квазігрупа має свій номер, згідно якого й виконуються шифрування в подальшому, наприклад: перших два біти ключа $K_{0\text{-}1}$=”10”, після переведення з двійкової системи числення отримуємо номер “10”=2, тобто використовуємо квазігрупу L_2.

Етап за шифрування та розшифрування даних вимагає пари бітів ключа та пари бітів повідомлення. Обрані біти використовуються для проведення операції квазігрупи, номер якої визначено за допомогою перших бітів ключа, рисунок 3.1.

K_{2-3} \ M_{0-1}	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	01	00
11	11	10	00	01

Рисунок 3.1 – Результат операції квазігрупи

Отриманий результат надсилається на 4-вхідний мультиплексор, за допомогою якого, вхідні потоки даних з квазігруп формують один єдиний потік інформації.

Для генерації пари біт ключа, що буде додаватись до послідовності ключа, використовуються пара біт ключа та пара біт повідомлення. Для отримання результату використовуються біти ключа K_{2-3} та біти повідомлення M_{0-1}. Отримані біти виконують операцію відповідної квазігрупи, отриманий результат добавляється до послідовності ключа К.

Даний процес відбувається поки послідовність потоку повідомлення не буде дорівнювати нулю.

3.2 Структурна схема для потокового шифрування

Апаратна реалізація даного алгоритму шифрування вимагає його структурного представлення, для цього й буде створена структурна схема алгоритму потокового шифрування на основі латинських квадратів.

Насамперед, варто визначити логічні елементи, що будуть використовуватись в роботі даного алгоритму шифрування. Для зберігання даних будуть використовуватись тригери, їх кількість залежить від довжини ключа та вхідного повідомлення. Довжина ключа 64 біт, тому для реалізації буде використовуватись 64 тригери та 2 тригери повідомлення для мінімально можливої роботи програми. Тригери використовуються для збереження стану ключа та повідомлення. Структурну схему схему зображено на рисунку 3.2.

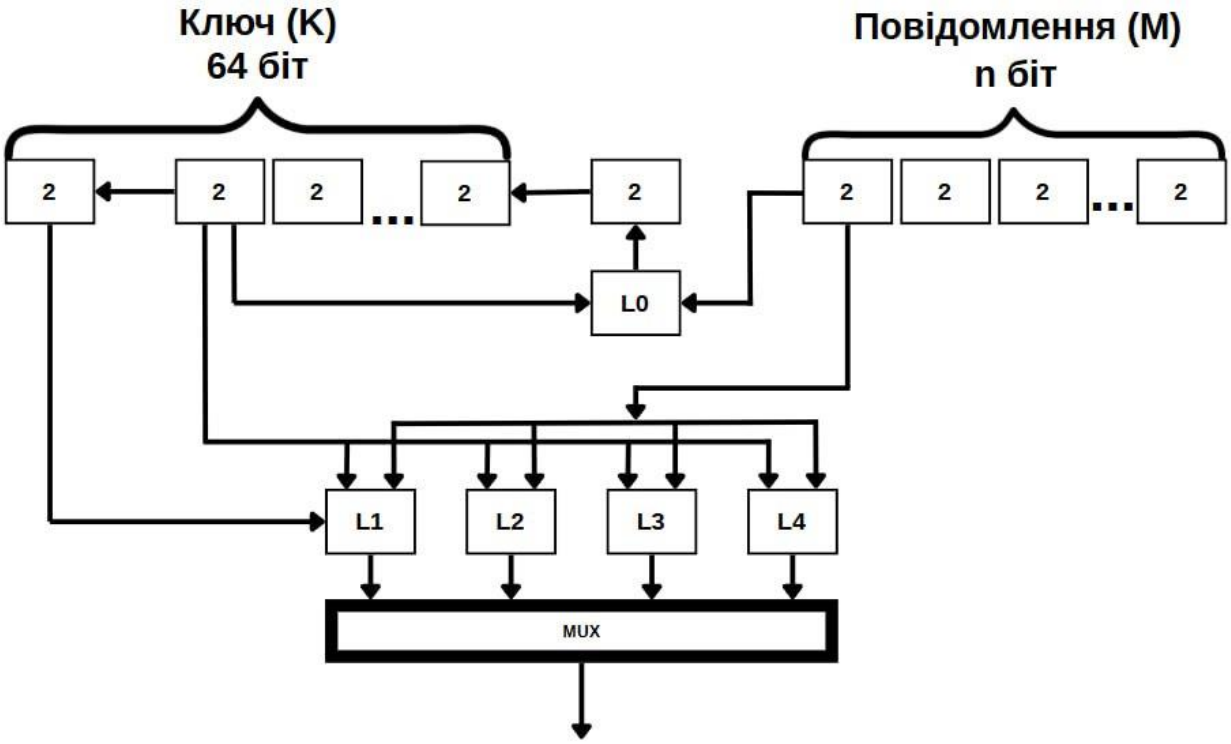


Рисунок 3.2 – Структурна схема алгоритму потокового шифрування

На рисунку 3.2 зображено блоками по 2 біти, кожні 2 тригери, що використовують квазігрупи. Квазігрупи обрахунку послідовності шифрування зображені з своїм номером L_1, L_2, L_3, L_4. Квазігрупа зображено з номером 0, тобто L_0, використовується для додавання 2 кінцевих бітів до послідовності ключа. На рисунку зображено також потік даних, що відбувається в роботі самої програми. Результат кожного обчислення квазігрупи передається на мультиплексор з 4 входами, на рисунку 3.2 має позначення MUX.

3.3 Програмна реалізація алгоритму потокового шифрування

Існує безліч можливих варіантів для реалізації даного алгоритму серед мов програмування. Три найпопулярніші мови – Python, C++, Java [28]. Їх можна використовувати для реалізації безлічі цілей. Розглядаючи їх потрібно вияснити, за допомогою якої буде реалізовуватись даний алгоритм шифрування.

Python є інтерпретованою мовою,що дозволяє виконувати код без компіляції у машинний код. Це забезпечує гнучкість, швидкий цикл розробки,

можливість інтерактивного тестування та налагодження програм.

Під час розробки Python був зроблений акцент на легкість читання та розуміння коду. Структура мови наближена до природної мови, що знижує поріг входу для початківців та забезпечує легкість супроводу та масштабування проектів [29].

Дана мова програмування підтримує роботу на багатьох операційних системах, включаючи Windows, Linux, macOS та інші.

Python має велику стандартну бібліотеку, що охоплює широкий спектр завдань – від роботи з файлами та мережами до обробки тексту, даних, шифрування, роботи з інтернет-протоколами та багатьох інших.

Навколо Python сформувалася одна з найбільших у світі спільнот розробників. Існує величезна кількість відкритих бібліотек, фреймворків та інструментів для найрізноманітніших сфер застосування – від веб-розробки та наукових обчислень до машинного навчання та кібербезпеки [30].

Python використовує динамічну типізацію, тобто типи змінних визначаються під час виконання програми. Це робить мову гнучкішою, але також вимагає уважності від розробника при роботі з типами даних.

Python активно розвивається, при цьому зберігається зворотна сумісність більшості версій. Нові версії мови покращують продуктивність, безпеку та розширюють функціональність.

Однією з ключових переваг мови Python є наявність великої кількості бібліотек, які значно розширюють її функціональні можливості. Бібліотеки – це готові модулі коду, які дозволяють виконувати специфічні завдання без необхідності самостійної розробки відповідних алгоритмів.

Python має широку стандартну бібліотеку, що охоплює основні сфери застосування: робота з файлами, обробка тексту, мережеві протоколи, криптографія, багатопоточність, бази даних тощо. Окрім стандартних модулів, існує велика кількість сторонніх бібліотек, які активно підтримуються спільнотою та охоплюють такі галузі, як:

- наукові обчислення (NumPy, SciPy),

- аналіз даних (Pandas),
- машинне навчання (scikit-learn, TensorFlow, PyTorch),
- веб-розробка (Django, Flask),
- автоматизація (Selenium, PyAutoGUI),
- візуалізація даних (Matplotlib, Seaborn),
- кібербезпека (Scapy, Requests, Paramiko).

Загалом, Python це надзвичайно потужний інструмент для розробки програмних застосунків, завдяки простоті та широкому вибору бібліотек та фреймворків.

C++ – це універсальна мова програмування високого рівня, яка поєднує можливості процедурного, об'єктно-орієнтованого та шаблонного програмування з ефективністю та низькорівневим контролем апаратних ресурсів.

C++ дозволяє працювати з пам'яттю та апаратними ресурсами безпосередньо, що забезпечує високу швидкість виконання програм та їх ефективність [31]. Це робить C++ придатною для системного програмування, розробки драйверів, операційних систем, ігор, вбудованих систем.

Код C++ перед виконанням компілюється у машинний код, що забезпечує максимальну продуктивність при виконанні. Компілятор аналізує код на стадії компіляції, що дозволяє виявляти багато помилок ще до запуску програми.

C++ активно підтримує концепції ООП, зокрема:

- інкапсуляцію,
- наслідування,
- поліморфізм,
- абстракцію.

Завдяки цьому можливо створювати складні програмні системи з чіткою структурою, повторним використанням коду та високою підтримуваністю.

C++ надає потужний механізм шаблонів, що дозволяє створювати універсальні алгоритми та структури даних, які можуть працювати з будь-якими типами даних без дублювання коду.

Стандартна бібліотека шаблонів містить набір готових до використання контейнерів, алгоритмів, ітераторів та інших інструментів, що полегшують розробку ефективних і надійних програм.

На відміну від деяких високорівневих мов, C++ вимагає уважного ставлення до управління пам’яттю, ресурсами та правильності використання мовних конструкцій, оскільки це безпосередньо впливає на стабільність і безпеку програми.

На відміну від Python, C++ складніша у вивчені мова, але має свої переваги за рахунок своєї продуктивності та ручним керуванням пам’яті.

Java – це об’єктно-орієнтована, високорівнева, багатоплатформна мова програмування, яка дозволяє розробляти надійні, масштабовані та безпечні додатки для широкого спектра пристроїв і середовищ. Java повністю побудована на основі об’єктно-орієнтованих принципів. Основні концепції ООП, такі як інкапсуляція, наслідування, поліморфізм і абстракція, лежать в основі всієї архітектури мови. Ключовою особливістю Java є принцип "Write Once, Run Anywhere" – один раз написана програма може виконуватись на будь-якій платформі, де є Java Virtual Machine. Це досягається завдяки компіляції коду в байт-код, який виконується JVM, незалежно від операційної системи чи апаратного забезпечення.

Java розроблена з урахуванням безпеки. Мова включає вбудовані механізми контролю доступу, захисту пам’яті, обробки помилок і винятків, що робить її надійною для створення критично важливих застосунків. У Java використовується автоматичний збірник сміття, який звільняє неактуальні об’єкти з пам’яті. Це зменшує ймовірність виникнення помилок, пов'язаних з неправильним управлінням пам’яттю. Типи змінних у Java визначаються під час компіляції. Це дозволяє виявляти багато помилок ще на етапі компіляції, що підвищує стабільність та безпеку програм. Java має вбудовану підтримку багатопоточності – можливості одночасного виконання кількох потоків, що дозволяє ефективно використовувати ресурси сучасних багатоядерних процесорів. Java має потужну стандартну бібліотеку, яка надає широкий спектр

готових інструментів для роботи з файлами, мережами, базами даних, колекціями, криптографією, інтерфейсами користувача та іншими сферами. Java активно використовується у великих корпоративних системах, серверних застосунках, банківських системах, телекомунікаціях, великих веб-сервісах та мобільних додатках завдяки своїй стабільності, надійності та масштабованості.

Отже, Java – це універсальна мова програмування, яка поєднує високу продуктивність, крос-платформенність та надійність. Завдяки потужній екосистемі, великій спільноті та сучасним інструментам, вона залишається оптимальним вибором для розробки складних корпоративних рішень, мобільних та веб-додатків. Її строга типізація, автоматичне керування пам'яттю та підтримка мультипотоковості роблять Java ідеальним інструментом для створення стабільних, безпечних та масштабованих програмних продуктів.

Серед розглянутих мов програмування, можна відзначити основні плюси кожної з них, наприклад: Python – має основну перевагу в тому, що існує безліч готових рішень, бібліотек для отримання результату. Також однією з переваг Python та Java над C++ є переносимість коду, хоча і швидкодія C++ має певні переваги.

Отже, для реалізація даного програмного засобу, краще використовувати Python, оскільки дана мова програмування має набагато ширшу в порівнянні з іншими мовами бібліотеку.

3.4 Оцінки апаратної складності реалізації алгоритму потокового шифрування

Для обчислення апаратної складності алгоритму потокового шифрування використовуються умовні одиниці GE (Gate Equivalent), які описують кількість логічних елементів, потрібних для реалізації програмного засобу.

Обчислення складності даного засобу виконується з використанням бібліотеки, що містить в собі інформацію про складність кожного з цих елементів в одиницях GE. Для виконання обчислення буде використовуватись бібліотека UMCL18G212T3, що описує всі логічні елементи, необхідні для

реалізації даного алгоритму. Список необхідних елементів, та їх апаратна складність в одиницях GE наведено в таблиці 3.1.

Таблиця 3.1 – Значення складності в GE для основних елементів апаратної реалізації

Елемент	GE(Gate Equivalent)
НІ	0,67
І	1,33
АБО	1,33
ВИКЛЮЧНЕ АБО	2,67
І-НІ	1,00
АБО-НІ	1,00
МУЛЬТИПЛЕКСОР(4)	8,75
ТРИГЕР	5,33

Для обчислення апаратної складності засобу потокового шифрування на основі латинських квадратів необхідно провести підрахунок елементів, що потрібно застосовувати для побудови засобу шифрування.

В даному програмному засобі використовується 64 бітний ключ, відповідно цьому використовується 64 тригери для зберігання стану. Для мінімальної роботи програми також необхідно використовувати 2 тригери повідомлення, для перших двох бітів. Загалом в даному програмному засобі використовується 66 D-тригерів, кожен з яких має апаратну складність 5,33 GE, що в загальному становить 351,78 GE.

В подальшому слід розглянути кожну з наявних квазігруп, для обчиленнях їх, необхідно розглянути самі квазігрупи. Для реалізації квазігрупи, необхідно використовувати логічні вирази. Кожна квазігрупа може створюватись за допомогою двох логічних виразів, тобто кожна квазігрупа L_n реалізується за допомогою двох логічних виразів: $L_n z_1$ та $L_n z_2$.

Квазігрупа L_0, що зображена у таблиці 3.2, реалізується відповідно двома логічними виразами.

Таблиця 3.2 – Квазігрупа L_0

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Її реалізація представлена нижче у формулах.

$$z_1 = x_2 y_2 \oplus (x_1 \oplus y_1)$$

$$z_2 = x_2 \oplus y_2$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_0 = z_1; z_2 = x_2 y_2 \oplus (x_1 \oplus y_1); x_2 \oplus y_2$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 1 елемент I та 3 елементи виключне АБО. Після виконання обчислень, отримуємо значення, в яких дана квазігрупа має складність 9.34 GE.

Квазігрупа L_1, що зображена у таблиці 3.3, реалізується відповідно двома логічними виразами.

Таблиця 3.3 – Квазігрупа L_1

⊕	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_1 y_1 \oplus (x_2 \oplus y_2)$$

$$z_2 = x_1 y_1 \oplus (x_2 \oplus y_2)$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_1 = (z_1; z_2) = x_1 y_1 \oplus (x_2 \oplus y_2);\ x_1 y_1 \oplus (x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї

квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 2 елементи I та 4 елементи виключне АБО. Після виконання обчислень, отримуємо значення, в яких дана квазігрупа має складність 13.34 GE.

Квазігрупа L_2, що зображена у таблиці 3.4, реалізується відповідно двома логічними виразами.

Таблиця 3.4– Квазігрупа L_2

*	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	01	00
11	11	10	00	01

Її реалізація представлена нижче у формулах.

$$z_1 = x_1 \oplus y_1$$

$$z_2 = x_1 y_1 \oplus (x_2 \oplus y_2)$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_2 = z_1; z_2 = x_1 \oplus y_1; x_1 y_1 \oplus (x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 1 елемент I та 3 елементи виключне АБО. Після виконання обчислень, отримуємо значення, в яких дана квазігрупа має складність 9.34 GE.

Квазігрупа L_3, що зображена у таблиці 3.5, реалізується відповідно двома логічними виразами.

Таблиця 3.5 – Квазігрупа L_3

◇	00	01	10	11
00	00	01	10	11
01	01	11	00	10
10	10	00	11	01
11	11	10	01	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_1x_2\overline{y_2} \vee x_1x_2\overline{y_1} \vee x_1x_2y_2 \vee \overline{x_1x_2}y_2$$

$$z_2 = x_1x_2\overline{y_1} \vee x_1\overline{x_2}y_2 \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_3 = z_1; z_2 = x_1x_2\overline{y_2} \vee x_1x_2\overline{y_1} \vee x_1x_2y_2 \vee \overline{x_1x_2}y_2;$$

$$x_1x_2\overline{y_1} \vee x_1\overline{x_2}y_2 \vee x_1\overline{x_2y_2} \vee \overline{x_1x_2}y_1$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 6 елементів АБО, 10 елементів І та 12 елементів НІ. Після виконання обрахунків, отримуємо значення, в яких дана квазігрупа має складність 29.32 GE.

Результатом обчислень є те,що складність даного алгоритму для зашифрування становить 61.32 GE.

Наступним етапом обрахунків, буде обрахування квазігруп для розшифрування вхідного потоку даних.

Квазігрупа ${}^{l}L_0$, що зображена у таблиці 3.6, реалізується відповідно двома логічними виразами.

Таблиця 3.6 – Квазігрупа ${}^{l}L_0$

${}^{l}+$	00	01	10	11
00	00	11	10	01
01	01	00	11	10
10	10	01	00	11
11	11	10	01	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_2(x_1 \oplus y_1) \vee \overline{x_2}(x_1 \oplus (y_1 \oplus y_2)$$

$$z_2 = x_2 \oplus y_2$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_0 = (z_1; z_2) = (x_2(x_1 \oplus y_1) \vee \overline{x_2}(x_1 \oplus (y_1 \oplus y_2); x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 1 елемент АБО, 2 елементи І та 1 елементів НІ, 4 елементи виключне АБО. Після виконання обрахунків, отримуємо значення, в яких дана квазігрупа має складність 15.34 GE.

Квазігрупа ${}^{l}L_1$, що зображена у таблиці 3.6, реалізується відповідно двома логічними виразами.

Таблиця 3.7 – Квазігрупа ${}^{l}L_1$

${}^{l}\oplus$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_1 y_1 \oplus (x_2 \oplus y_2)$$

$$z_2 = x_1 y_1 \oplus (x_2 \oplus y_2)$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_1 = z_1; z_2 = x_1 y_1 \oplus (x_2 \oplus y_2); x_1 y_1 \oplus (x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 1 елемент АБО, 2 елементи І та 4 елементи виключне АБО. Після виконання обрахунків, отримуємо значення, в яких дана квазігрупа має складність 13,34 GE.

Квазігрупа ${}^{l}L_2$, що зображена у таблиці 3.7, реалізується відповідно двома логічними виразами.

Таблиця 3.7 – Квазігрупа ${}^{l}L_2$

${}^{l}*$	00	01	10	11
00	00	01	11	10
01	01	00	10	11
10	10	11	00	01
11	11	10	10	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_1 \oplus y_1$$

$$z_2 = x_1(x_2 \oplus y_2 \vee \overline{x_1}(y_1 \oplus (x_2 \oplus y_2)$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_2 = z_1; z_2 = x_1 \oplus y_1; x_1(x_2 \oplus y_2 \vee \overline{x_1}(y_1 \oplus (x_2 \oplus y_2)$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу. Дана квазігрупа має в собі 1 елементи АБО, 2 елементи І та 4 елементи виключне АБО та 1 елемент НІ. Після виконання обрахунків, отримуємо значення, в яких дана квазігрупа має складність 15,34 GE.

Квазігрупа ${}^{l}L_3$, що зображена у таблиці 3.8 реалізується відповідно двома логічними виразами.

Таблиця 3.8 – Квазігрупа ${}^{l}L_3$

◇	00	01	10	11
00	00	10	01	11
01	01	00	11	10
10	10	11	00	10
11	11	01	10	00

Її реалізація представлена нижче у формулах.

$$z_1 = x_1 x_2 \overline{y_2} \vee x_1 x_2 \overline{y_1} \vee x_1 x_2 y_2 \vee \overline{x_1 x_2} y_2$$

$$z_2 = x_1 x_2 \overline{y_1} \vee x_1 \overline{x_2} y_2 \vee x_1 \overline{x_2 y_2} \vee \overline{x_1 x_2} y_1$$

Спрощені логічні вирази, що реалізують дану квазігрупу мають вигляд:

$$L_2 = z_1; z_2 = x_1 x_2 \overline{y_2} \vee x_1 x_2 \overline{y_1} \vee x_1 x_2 y_2 \vee \overline{x_1 x_2} y_2;$$

$$x_1x_2\overline{y_1} \vee x_1\overline{x_2}y_2 + x_1\overline{x_2y_2} + \overline{x_1x_2}y_1$$

В результаті отримали спрощений логічний вираз, з якого можемо обчислити кількість логічних елементів, що є необхідними для реалізації цієї квазігрупи в апаратному засобі, за рахунок цього можна обраховувати апаратну складність засобу, отже в цьому апаратному засобі необхідно застосувати 6 елементів АБО, 12 логічних НІ та 10 логічних І для розшифрування вхідного потоку даних. Після виконання обрахунків, отримуємо значення, в яких дана квазігрупа має складність 29,33 GE.

Результатом обчислень є те, що складність даного алгоритму для розшифрування становить 73.44 GE.

Також слід врахувати використання мультиплексора з 4 входами для реалізації вихідного потоку даних потокового алгоритму шифрування на основі латинських квадратів, його складність становить 8.75 GE. У таблиці 3.9 наведено таблицю складності модулів засобу шифрування даних.

Таблиця 3.9 – Складність модулів засобу шифрування

Модуль	GE(Gate Equivalent)
Квазігрупи	61.32
Тригери	351,78
Мультиплексор	8,75

Отримавши значення складності всіх модулів можна обрахувати загальному складність засобу потокового шифрування даних. В результаті обчислень, складність засобу становить 421.85 GE [32].

3.5 Аналіз атак на алгоритм шифрування

Під час створення алгоритму потокового шифрування, вимагається виконання певних тестів, для перевірки стійкості алгоритму до стандартних атак. Атака не повинна бути швидшою аніж, звичайний підбір ключа, тобто не швидше аніж 2^{64}.

Розглянемо типи атак та виконаємо класифікацію.

Якщо противник має на меті відновити ключ або початковий стан генератора, то атака такого типу називається спрямованою на відновлення ключа (key recover attack) або, відповідно, початкового стану.

Можливості противника. Ця ознака визначає, як саме обмежений доступ противника до оракула, що реалізує зашифрування повідомлень (при випадково обраному невідомому ключі). В залежності від можливостей противника виділяють: атаки на основі відомого шифротексту, атаки на основі відомих або підібраних відкритих текстів та атаки на основі відомих або підібраних векторів ініціалізації.

Якщо противнику відомі відкритий та шифрований тексти, він може обчислити гаму. Отже, в цьому випадку можна ставити задачу відновлення стану квазігрупи.

В залежності від методів, які використовуються для розв’язання криптоаналітичних задач, виділяють: алгебраїчні, статистичні (кореляційні), змішані атаки тощо.

Алгебраїчними називають атаки, що базуються на розв’язанні систем алгебраїчних рівнянь, які пов’язують знаки гами, виробленої генератором, з його початковим станом.

Статистичні атаки базуються на ймовірнісно-статистичних методах і, як правило, полягають у розв’язанні задач перевірки (двох або декількох) статистичних гіпотез.

Змішаними називають атаки, які базуються на сумісному застосуванні різних за сутністю математичних методів. Як приклад, відзначимо атаки типу TMDTO (Time Memory Data Trade Off), які поєднують перебір зі статистичними методами.

Зрозуміло, що клас, до якого відносять ту чи іншу атаку за типом методу криптоаналізу, визначається досить умовно. В залежності від наведених ознак, можна класифікувати практично будь-яку атаку на потоковий шифр [33].

3.6 Приклад програмного засобу

Програмний засіб, який було реалізовано має два режими роботи, за шифрування та розшифрування, найкращим чином для реалізації цього за допомогою Python, є RadioButton бібліотеки Tkinter. Як вхідні параметри ми приймаємо вхідний текст і ключ, отже слід використовувати поля для вводу тексту. Для ключа необхідно створити обмеження в 64 біти, тобто 8 елементів, реалізується за допомогою умови. Також слід створити поле для виведення зашифрованого та розшифрованого тексту. Для зручності в роботі програми також створено кнопку за допомогою якої, можна копіювати результат роботи програми. Вікно програми зображене на рисунку 3.1.

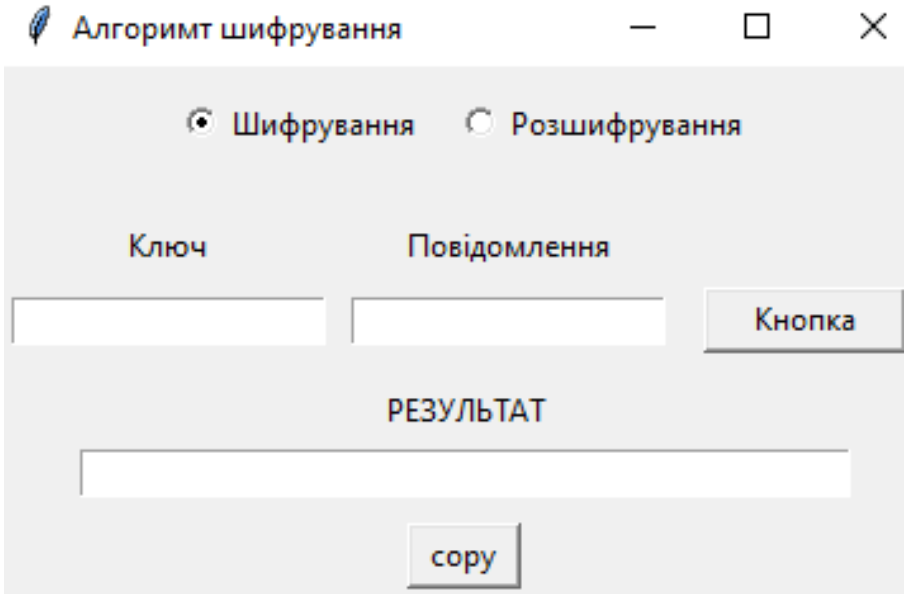


Рисунок 3.1 – Вікно програми шифрування

За змовчування вибраний режим роботи, шифрування, після його зміни, та натисненні кнопки «Кнопка», програма виконує роботу, приклад виконання рисунок 3.2.

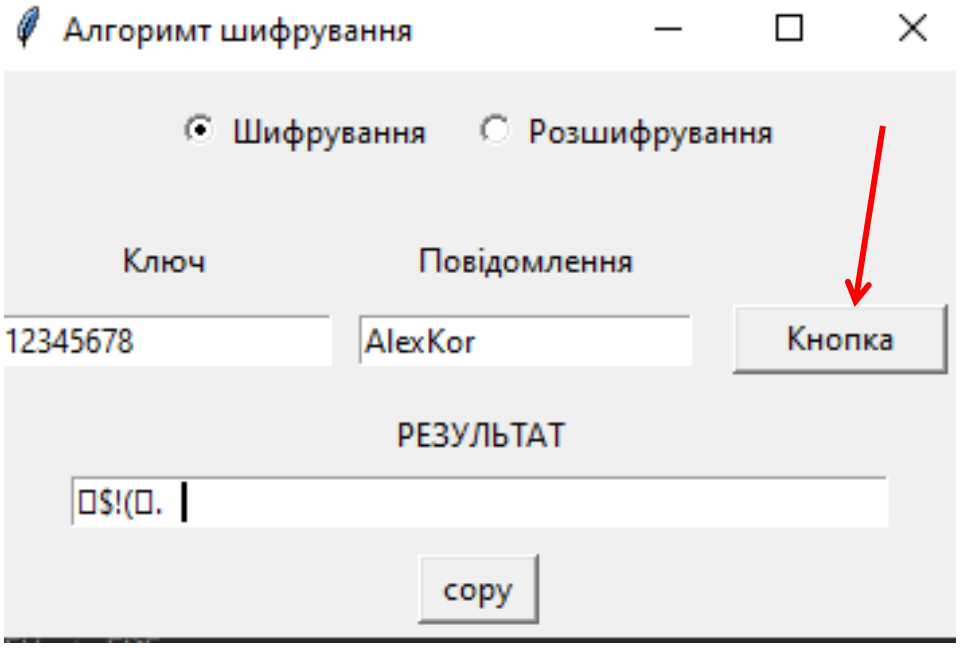


Рисунок 3.2 – Приклад виконання програми

В ході виконання у полі «Результат» ми отримали результат шифрування даних. Отриманий результат спробуємо розшифрувати, для цього необхідно змінити режим роботи, на розшифрування, рисунок 3.3.

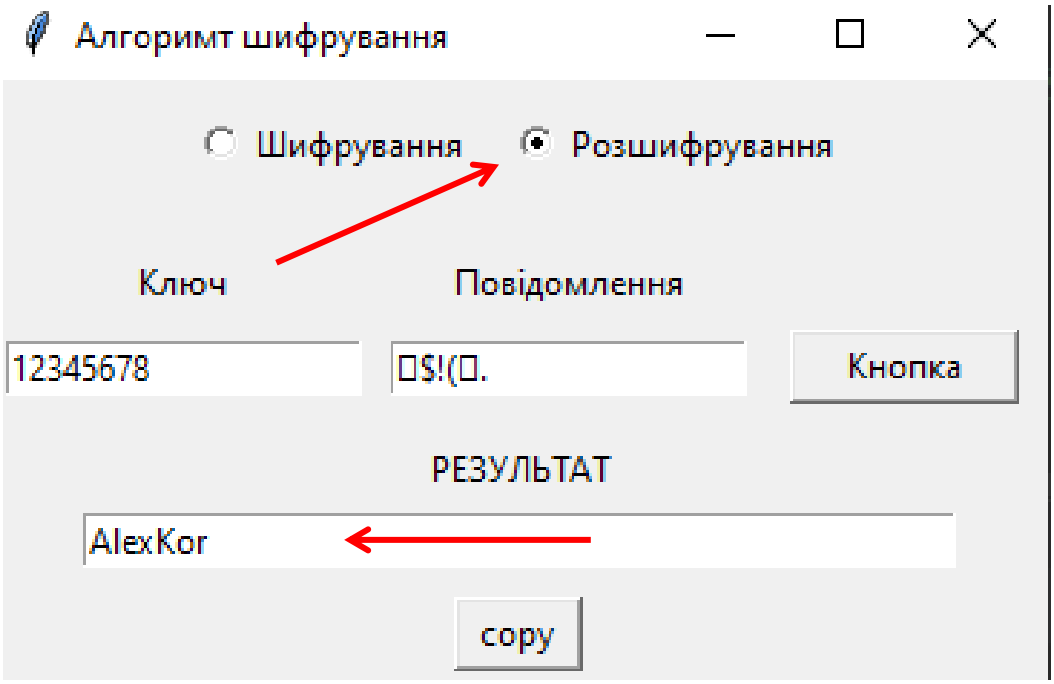


Рисунок 3.3 – Приклад роботи програми

Результат шифрування та розшифрування збігається, спробуємо спотворити зашифрований текст, для перевірки на правильність, рисунок 3.4.

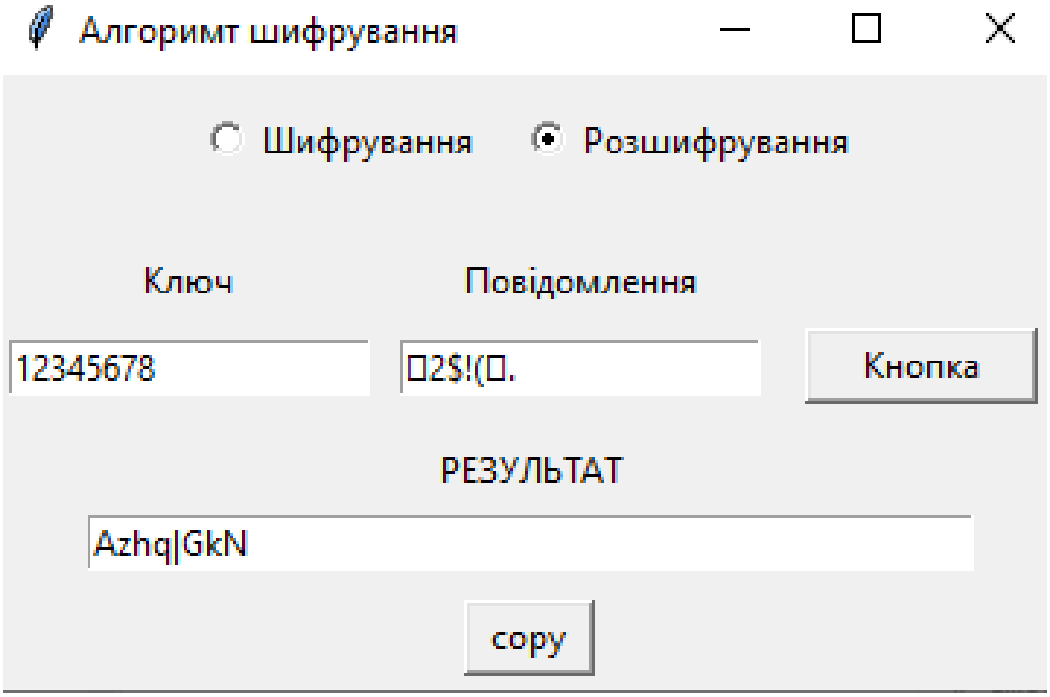


Рисунок 3.4 –Приклад роботи програми

Після внесення змін у вихідний зашифрований текст, результат змінюється. Наступна перевірка буде включати в себе зміну ключа, рисунок 3.5-3.6.

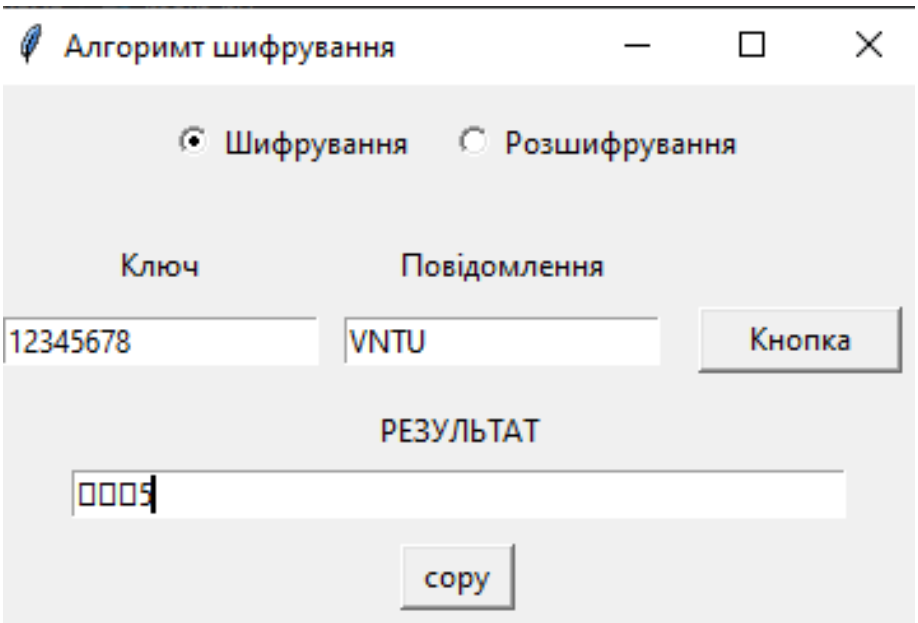


Рисунок 3.5 – Приклад роботи програми

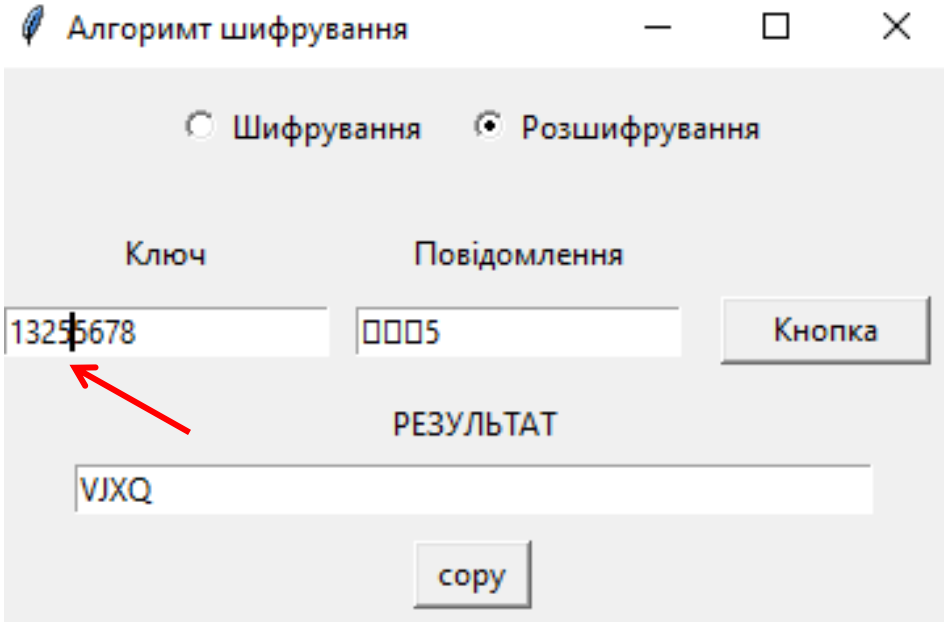


Рисунок 3.6 – Приклад роботи програми

Після внесення змін в ключ, результат виведення змінився, й значення, які ми отримали не дорівнюють вхідному текстові. Виконаємо перевірку на рівність ключа 64 бітам, рисунок 3.7.

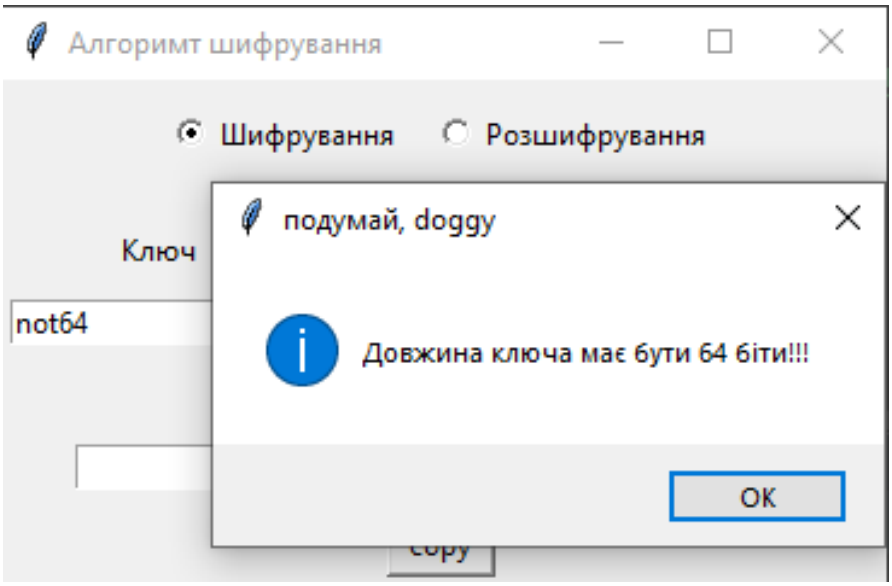


Рисунок 3.7 – Приклад роботи програми

Після введення некоректних даних ми отримуємо MessageBox, в якому повідомляється про необхідність внесення змін в довжину ключа.

3.7 Висновки до розділу

Проаналізувавши отриманий алгоритм роботи засобу потокового шифрування було визначено його апаратну складність, розроблено програмний

засіб, протестовано програмний засіб.

За створеною схемою апаратного засобу шифрування було обраховано складність засобу потокового шифрування, що становить 495.29 GE. Даний алгоритм є конкурентом серед засобів малоресурсної криптографії, оскільки його складність від 2 до 3.5 разів менша в порівнянні з іншими засоби потокового шифрування малоресурсної криптографії.

ВИСНОВКИ

Огляд інформаційних джерел виявив, що в сучасному світі існує нагальна потреба в розробці засобів потокового шифрування мало ресурсних засобів, з балансом між швидкодією та захистом інформації.

Під час дослідження інформаційних джерел, пов’язаних з засобами потокового шифрування, виявлено тенденцію, під час якої всі шифри потокового шифрування, мають особливість, під час використання певної кількості операції, виявляється необхідність до розширення секретного ключа, вектора ініціалізації та інших складових шифру потокового шифрування. Це є причиною того, що складність апаратної реалізації збільшується.

Розглянувши приклади застовування квазігруп на практиці, в якості криптографічних примітивів було виявлено, що застосування квазігруп, як основу для засобів шифрування є ефективним підходом. Наприклад, сімейство алгоритмів Edon80, Edon-R, Edon-L, показують свою ефективність на практиці.

Результатом виконання розробки алгоритму шифрування було розроблено метод, що описує процес шифрування та алгоритм роботи засобу, що демонструє процес за шифрування та розшифрування.

Для реалізації даного алгоритму використовуються 0-лупи, тобто квазігрупи, що мають нейтральних елемент, всього їх 4, всі вони використовуються.

В результаті розробки даного алгоритму, отримали можливість обрахунки апаратної складності даного засобу потокового шифрування на основі латинських квадратів. В результаті обчислень, складність засобу становить 421.85 GE.

Також був протестований програмний засіб, реалізований для реалізації методу за шифрування та розшифрування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DES: web-site. URL: https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/ (access 23.05.25)
2. RC4: web-site. URL: https://www.geeksforgeeks.org/rc4-encryption-algorithm/ (access 23.05.25)
3. RC4 не підтримується Mircosoft: веб-сайт. URL: Посилання (дата звернення 23.05.25)
4. Submission requirements and evaluation criteria for the lightweight cryptography standardization process. NIST Computer Security Resource Center | CSRC.:web-site. URL: https://csrc.nist.gov/CSRC/media/Projects/LightweightCryptography/documents/final-lwc-submission-requirements-august2018.pdf (access 23.05.25)
5. AES-256: веб-сайт.URL: https://www.vpnunlimited.com/ua/help/cybersecurity/aes-256?srsltid=AfmBOoritIEz3dvzaegS9OtXB_5amUeIysiek8nrEIo7c1BwB0jyYBgO (дата зверення 23.05.25)
6. What is the RSA algorithm: web-site. URL: https://www.techtarget.com/searchsecurity/definition/RSA (access 23.05.25)
7. Salsa20: web-site. URL: https://www.cryptopp.com/wiki/Salsa20 (access 23.05.25)
8. Kuznetsov O., M. Lutsenko and D. Ivanenko, "Strumok stream cipher: Specification and basic properties," 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 59-62.
9. A5/1 Stream Cipher: web-site. URL: https://asecuritysite.com/encryption/a5 (access 23.05.25)
10. NESSIE: web-site. URL: https://web.archive.org/web/20180612183337/https://www.cosic.esat.kuleuven.be/nessie/ (access 23.05.25)
11. GRAIN: web-site.URL: https://www.ecrypt.eu.org/stream/ciphers/grain/grain.pdf (access 23.05.25)
12. MICKEY: web-site.URL: https://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf (access 23.05.25)
13. TRIVIUM: web-site.URL:

https://web.archive.org/web/20170312043910/http://wp.iese.edu.ar/investigacion/Model%20Design%20for%20a%20Reduced%20Variant%20of%20a%20Trivium%20Type%20Stream%20Cipher.pdf (access 23.05.25)

14. Modification of Edon80 to Resist the Key Recovery Attack: web-site. URL: (PDF) Modification of Edon80 to Resist the Key Recovery Attack (access 23.05.25)
15. Gligorosko D., Markovski S., Kocarev L., Edon–R, An Infinite Family of Cryptographic Hash Functions // International Journal of Network Security. − 2009. − Vol.8, No.3. − P.293-300.
16. Пилявець І. Ю. Засіб шифрування на основі квазігрупи. Частина 1. Реалізація квазігрупи: бакалаврська робота. Вінниця, 2024. 84 с.
17. Радченко Є.В. Засіб шифрування даних на основі квазігрупи. Частина 2. Криптографічний алгоритм: бакалаврська робота. Вінниця, 2024. 78 с.
18. Шелепало Г. В. Класифікація квазігрупових функційних рівнянь і тотожностей мінімальної довжини : кандидатська дисертація. Хмельницький, 2013. 191 с.
19. Крайнічук Г. В., Корінний О.О. Про деякі квазігрупові тотожності мінімальної довжини у шифруванні. *Всеукраїнська науково-технічна конференція*: тези доп. всеукр. наук.-практ. конф. м.Вінниця, 24-27 бер. 2025 р.
20. Сохацький Ф.М., Луценко А.В., Фриз І.В .Побудова квазігруп з властивістю оборотності // Мат. методи та фіз.-мех. поля. 64, − 2021. – С
21. Keedwell A. D., Denes J. Latin Squares and their Applications // North Holland: Elsevier B.V. − 2015. − 455 p.
22. H. Krainichuk, L. Ivanova Logical schemes of some quasigroups up to the parastrophic symmetry for LW-cryptography.QRS, Кишинів, Молдова – 2025.
23. Н.Р. Кондратенко, А.В. Остапенко-Боженова Глави дискретної математики з питань захисту інформації. // Посібник: офіційне видання, ВНТУ – 2022. – 89с.
24. Крайнічук Г.В., Остафійчук Д.В. Логічна схема латинського квадрата квазігрупи Кляйна. Молодь в науці: *дослідження, проблеми, перспективи*: тези міжнародної наук.-прак. інтернет-конференції. м. Вінниця, 15-16 бер. 2025 р.
25. Guohao Liu, Yunqing Xu Cryptographic classification of quasigroups of order 4 // International Workshop on Cloud Computing and Information Security (CCIS 2013). – 2015.DOI: 10.2991/ccis-13.2013.65

26. F.~M. Sokhatsky, H.~V. Krainichuk, V.~A. Luzhetsky Canonical and matrix figuration of quasigroups of the fourth order – 2024. – Issue 22. – P. 95–105
27. Рейтинг мов програмування 2025: веб-сайт. URL: https://dou.ua/lenta/articles/language-rating-2025/ (дата звернення 18.06.25)
28. Python: web-site. URL: https://www.python.org/ (access 18.06.25)
29. Можливості мови програмування Python: веб-сайт. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/ (дата звернення 18.06.25)
30. Java: web-site. URL: https://www.java.com/en/ (access 18.06.25)
31. C++: web-site. URL: https://learn.microsoft.com/cpp/?view=msvc-170 (access 18.06.25)
32. Virtual Silicon. Standard cell library UMCL18G212T3 based on the UMC L180 0.18 µm 1P6M Logic process. Official release, – 2004, p.1.
33. А.М. Олексійчук, О.В. Курінний Методи крипто аналізу потокових шифрів// Посібник: електронне мережне навчальне видання, КПІ – 2023 –172 с.

ДОДАТКИ

Додаток А

ПЕРЕЛІК ПУБЛІКАЦІЙ ЗА ТЕМОЮ

1) Крайнічук Г. В., Корінний О.О. Про деякі квазігрупові тотожності мінімальної довжини у шифруванні. *Всеукраїнська науково-технічна конференція*: тези доп. всеукр. наук.-практ. конф. м.Вінниця, 24-27 бер. 2025 р

64

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Засіб потокового шифрування на основі латинських квадратів

Автор роботи: Корінний Олександр Олегович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ , група 1 БКС-21 б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **21,54** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д. т. н., проф _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Кібербезпека критичних систем» к. т. н., доц. _____ Юрій БАРИШЕВ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Галина Шелепало

Здобувач _____ Олександр Корінний

65

ІЛЮСТРАТИВНА ЧАСТИНА

Засіб потокового шифрування на основі латинських квадратів

Виконав: студент 4 курсу групи 1БКС–21б
спеціальності 125 Кібербезпека
Олександр КОРІННИЙ
16 червня 2025 р.

Керівник: к.фіз-мат.н., доц. каф. ЗІ
Галина ШЕЛЕПАЛО
16 червня 2025 р.

Таблиця порівняння потокових шифрів

Назва алгоритму	Розмір ключа	Тактів ініціалізації	Складність (GE)
Grain	80/120	160	1294/1857
Trivium	80	80	749
Mickey	80/128	80/128	3188/5039
LKRP	64	32	483
Edon80	80/128	160	600/1110
Власний	64	32	427

Таблиця 0-лупи 4-го порядку

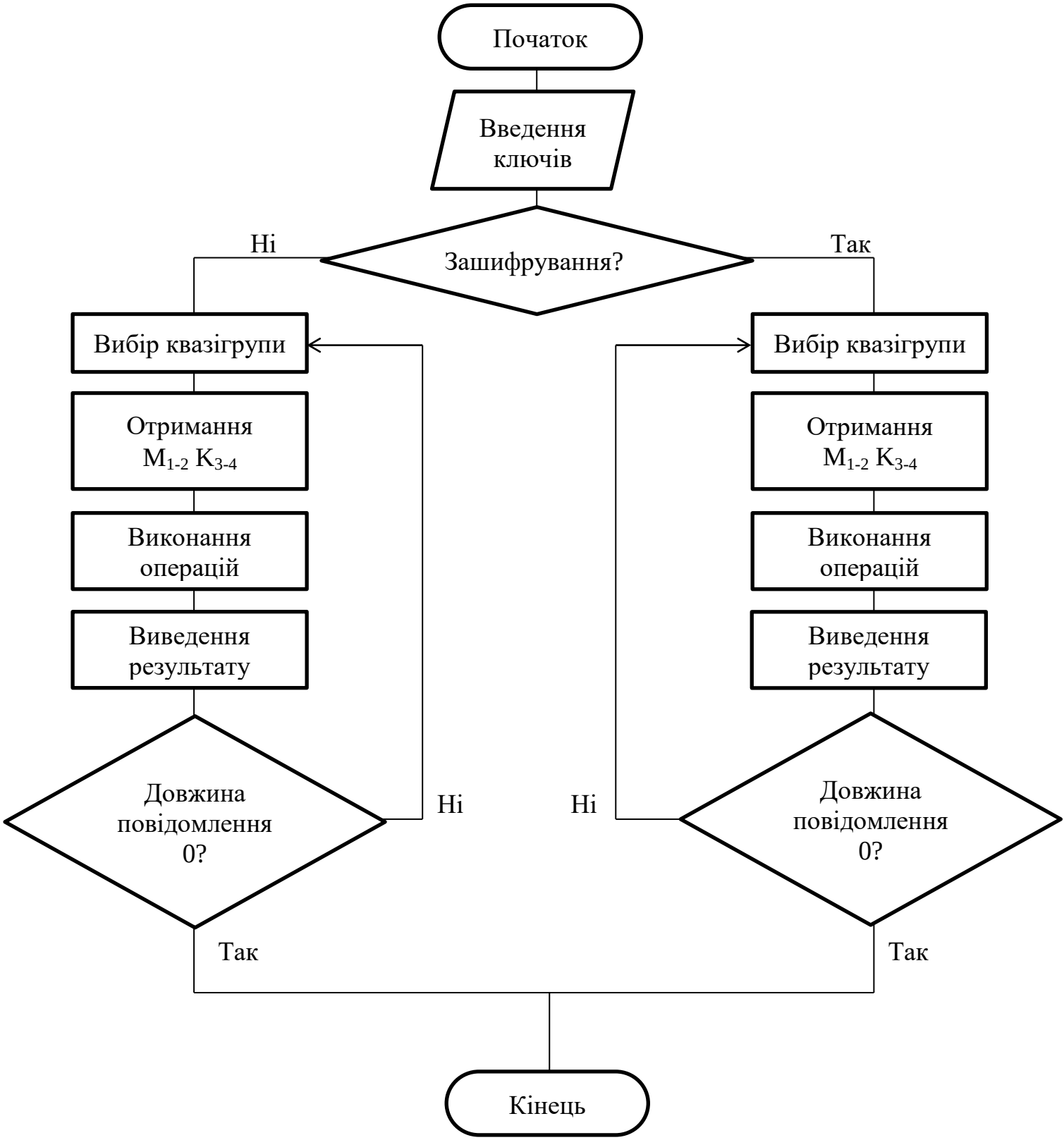
+	00	01	10	11
00	00	11	10	01
01	01	00	11	10
10	10	01	00	11
11	11	10	01	00

⊕	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

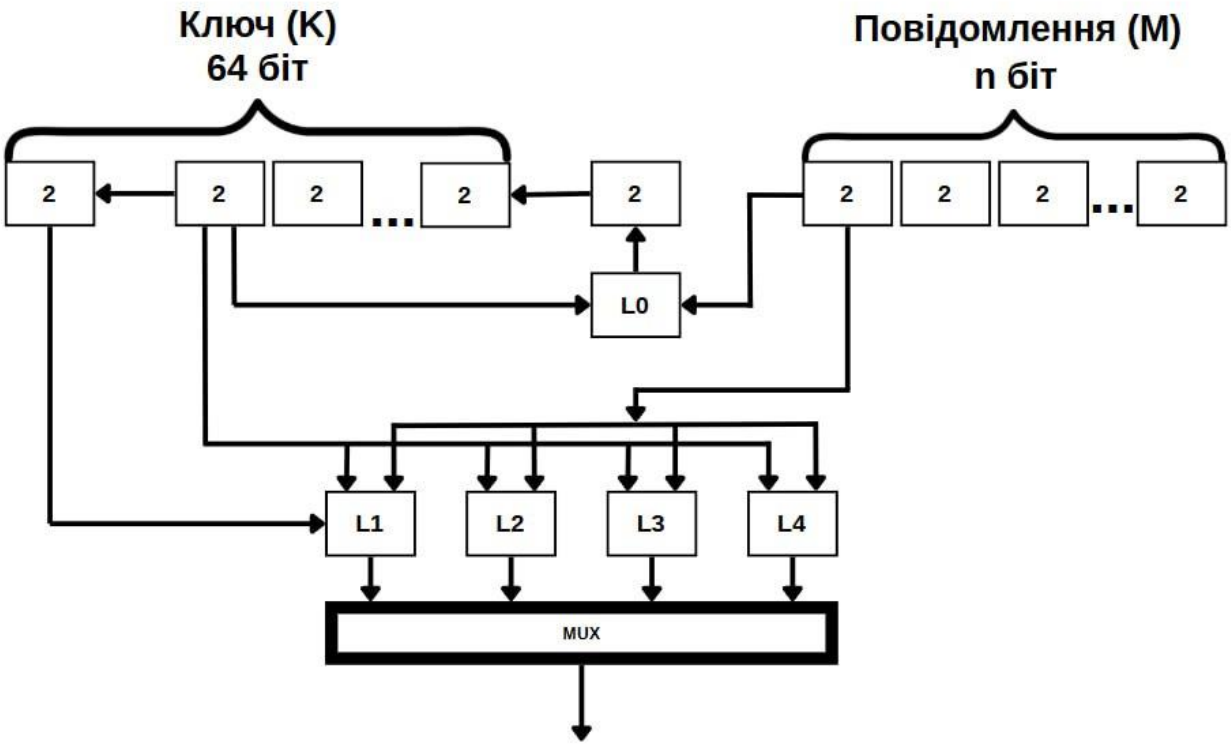
*	00	01	10	11
00	00	01	11	10
01	01	00	10	11
10	10	11	00	01
11	11	10	10	00

◇	00	01	10	11
00	00	10	01	11
01	01	00	11	10
10	10	11	00	10
11	11	01	10	00

Схема алгоритму шифрування



Структура засобу шифрування



Таблиця з оцінками складності апаратної реалізації засобу

Модуль	GE(Gate Equivalent)
Квазігрупи	66.64
Тригери	351,78
Мультиплексор	8,75

Приклад роботи програмного засобу