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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Методи та засоби криптографічних перетворень на основі латинських квадратів»

Виконав: студент 2-го курсу, групи 1БС-20м
спеціальності 125 – Кібербезпека
(шифр і назва напряму підготовки, спеціальності)
_____ Абрамюк О.А.
(прізвище та ініціали)
Керівник д.т.н., проф., зав. каф. ЗІ
_____ Лужецький В. А.
(прізвище та ініціали)
Опонент: к.т.н., проф. каф . ОТ
_____ Азарова А. О.
(прізвище та ініціали)

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ЗІ
д.т.н., проф.
_____Лужецький В. А.
«____»_____2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань 12 Інформаційні технології
Спеціальність 125 Кібербезпека
Освітньо-професійна програма Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д.т.н., проф.
_____ В.А. Лужецький
_____ _____ 2021 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Абрамюку Олегу Андрійовичу

1. Тема роботи: «Методи та засоби криптографічних перетворень на основі латинських квадратів» керівник роботи: Лужецький Володимир Андрійович, д. т. н., проф., зав. каф. ЗІ,
затверджені наказом ректора ВНТУ № 277 від 24 вересня 2021 р.
2. Строк подання студентом роботи 23 грудня 2021 р.
3. Вихідні дані до роботи:
   – криптографічні перетворення – блоковий та потоковий шифри;
   – довжина ключа – 64;
   – розрядність блоку – 64;
   – період генератора псевдовипадкових послідовностей – не менше 2^{32};
   – порядок латинських квадратів – 4.
4. Зміст розрахунково-пояснювальної записки:
Вступ. Аналіз інформаційних джерел. Метод генерування псевдовипадкових послідовностей на основі латинських квадратів. Метод блокового шифрування на основі латинських квадратів. Метод потокового шифрування на основі латинських квадратів. Апаратні засоби для шифрування даних. Економічна частина. Висновки. Перелік інформаційних джерел. Додатки.
5. Перелік ілюстративного матеріалу.
Підхід щодо генерування псевдовипадкових чисел на основі латинських квадратів (плакат, А4). Генератор послідовності псевдовипадкових чисел (плакат, А4). Структурна схема генератору псевдовипадкових чисел (плакат, А4). Структурна схема криптографічного генератору псевдовипадкових чисел

(плакат, А4). Апаратна реалізація латинських квадратів (плакат, А4). Структурна схема пристрою для блокового шифрування (плакат, А4). Структурна схема пристрою для потокового шифрування (плакат, А4). Оцінка складності апаратної реалізації потокових шифрів (плакат, А4). Оцінка складності апаратної реалізації блокових шифрів (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанти	Підпис, дата	
		завдання видав	завдання прийняв
1	Лужецький В. А.., д. т. н., проф., зав. каф. ЗІ		
2	Лужецький В. А.., д. т. н., проф., зав. каф. ЗІ		
3	Лужецький В. А.., д. т. н., проф., зав. каф. ЗІ		
4	Лужецький В. А.., д. т. н., проф., зав. каф. ЗІ		
5	Лесько О. Й., проф., зав. каф. ЕП і ВМ		

7. Дата видачі завдання 9 вересня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2021 – 04.09.2021	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2021 – 15.09.2021	
3	Науково-технічне обґрунтування	16.09.2021 – 22.09.2021	
4	Розробка технічного завдання	23.09.2021 – 04.10.2021	
5	Розробка методів криптографічних перетворень	05.10.2021 – 08.10.2021	
6	Розробка структурних схем пристроїв	09.10.2021 – 16.10.2021	
7	Розробка розділу економічного обґрунтування доцільності розробки	18.11.2021 – 21.11.2021	
8	Аналіз виконання ТЗ, висновки	22.11.2021 – 24.11.2021	
9	Оформлення пояснювальної записки	25.11.2021 – 29.11.2021	
10	Попередній захист та доопрацювання МКР	30.11.2021 – 12.12.2121	
11	Перевірка магістерської роботи на наявність плагіату	13.12.2121 – 18.12.2121	
12	Представлення МКР до захисту, опонування	19.12.2121 – 21.12.2121	
13	Захист МКР	22.12.2121 – 23.12.2121	

Студент _____ О. А. Абрамюк

Керівник роботи _____ В. А. Лужецький

АНОТАЦІЯ

УДК 681.325.5

Абрамюк О. А. Методи та засоби криптографічних перетворень на основі латинських квадратів. Магістерська кваліфікаційна робота зі спеціальності 125 – кібербезпека, освітня програма – Безпека інформаційних та комунікаційних систем. Вінниця: ВНТУ, 2021. 92 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 43; табл.: 11.

Магістерська кваліфікаційна робота присвячена розробці методів блокового та потокового шифрування, а також методу генерування псевдовипадкових послідовностей на основі латинських квадратів. Під час розробки методів шифрування проведено аналіз існуючих методів блокового та потокового шифрування, використання генераторів псевдовипадкових послідовностей у методах потокового шифрування та проаналізовано використання латинських квадратів у криптографії.

Графічна частина складається з 9 плакатів з демонстрацією структурних схем пристроїв.

В економічному розділі оцінено доцільність використання даної методики та витрати на її розробку.

ABSTRACT

Dushko AA Methodology of web application security testing. Master's thesis in the specialty 125 - cybersecurity, educational program - Security of information and communication systems. Vinnytsia: VNTU, 2021. 92 p.

In Ukrainian language. Bibliogr .: 31 titles; fig .: 43; tab .: 11.

The master's qualification work is devoted to the development of methods of block and streaming encryption, as well as the method of generating pseudo-random sequences based on Latin squares. During the development of encryption methods, the analysis of existing methods of block and streaming encryption, the use of pseudo-random generators in streaming encryption methods and the use of Latin squares in cryptography were analyzed.

The graphic part consists of 9 posters demonstrating the intermediate results of development, the final result and the experiments.

The economic section evaluates the feasibility of using this technique and the cost of its development.

ЗМІСТ

ВСТУП 7

1 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 9

1.1 Огляд блокового шифрування 9

1.2 Огляд потокового шифрування 11

1.3 Визначення латинського квадрату 19

1.4 Використання латинських квадратів в криптографії 21

2 РОЗРОБКА ГЕНЕРАТОРУ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ 26

2.1 Огляд методів генерування псевдовипадкових чисел 26

2.2 Розробка методу генерування псевдовипадкових чисел 29

2.3 Оцінка складності апаратної реалізації 32

3 РОЗРОБКА ПОТОКОВОГО ШИФРУ 52

3.1 Розробка методу потокового шифру 52

4 РОЗРОБКА БЛОКОВОГО ШИФРУ 61

4.1 Розробка методу блокового шифрування 61

5 ЕКОНОМІЧНА ЧАСТИНА 69

5.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки) 69

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько-технологічної роботи 74

5.3 Розрахунок мінімальної ціни та чистого прибутку від реалізації розробки методів та засобів криптографічних перетворень на основі латинських квадратів 79

5.4 Розрахунок терміну окупності коштів вкладених у наукову розробку методу та засобу автентифікації з нульовим знанням 80

5.5 Висновки до розділу 80

ВИСНОВКИ 82

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 83

Додаток А.Технічне завдання 86

Додаток Б. Критерії оцінювання комерційного потенціалу розробки 90

Додаток В. Результат перевірки роботи на плагіат 92

ВСТУП

На сьогоднішній день захист інформації є досить важливим завданням. У процесі розвитку суспільства люди поступово переходять до збереження важливої інформації та цінних паперів у електронному вигляді. З цієї причини виникає потреба захисту інформації від несанкціонованого доступу. Захист інформації в основному базується на використанні криптографічних методів, пов'язаних саме з шифруванням даних. Методи шифрування дозволяють досить надійно та ефективно захистити інформацію від несанкціонованого доступу.

Застосування криптографічного захисту, тобто використання процедури шифрування тексту за допомогою складних математичних алгоритмів поступово завойовує популярність. Методи шифрування бувають двох видів – блокові та потокові. Блокові шифри опрацьовують блоки даних розміром в декілька байт за одну ітерацію. Потокові шифри виконують шифрування кожного символу окремо та незалежно від решти символів.

Методів шифрування існує безліч, і всі побудовані на основі різних математичних операцій. Але пошук нових підходів до побудови шифрів не припинявся. Одним із можливих підходів є використання латинських квадратів. Вони уже були використані деякими шифрами раніше. На основі латинських квадратів було побудовано блоковий шифр IDEA. Також вони знайшли своє застосування у потоковому шифрі Edon80 та у сімействі геш-функцій Edon-R.

Актуальність дослідження, проведеного у магістерській кваліфікаційній роботі полягає у тому, що існуючі методи захисту інформації не є досконалими, оскільки не дозволяють у повній мірі забезпечити безпеку інформації, що передається каналами зв'язку.

Метою роботи є зменшення апаратних витрат на реалізацію пристроїв шифрування.

Для досягнення мети необхідно розв'язати такі задачі:

- проаналізувати відомі підходи та методи потокового та блокового шифрування
- проаналізувати використання латинських квадратів у криптографії;

- розробити метод та апаратний засіб генерування псевдовипадкових послідовностей на основі латинських квадратів;
- розробити метод та апаратний засіб потокового шифрування на основі латинських квадратів;
- розробити метод та апаратний засіб блокового шифрування на основі латинських квадратів.

Об'єкт дослідження – процеси криптографічного захисту інформації.

Предмет дослідження – методи та апаратні засоби потокового і блокового шифрування на основі латинських квадратів.

Наукова новизна отриманих результатів:

- вперше запропонований метод криптографічного генератора псевдовипадкових послідовностей, який відрізняється від відомих методів використанням набору латинських квадратів з певними властивостями, що забезпечують потрібні характеристики псевдовипадкових послідовностей;
- вперше запропонований метод потокового шифрування, який відрізняється від відомих методів використанням набору латинських квадратів і псевдонедетермінованим керуванням функціями шифрування, що забезпечує потрібний рівень криптографічної стійкості і зменшення апаратних витрат на його реалізацію;
- вперше запропонований метод блокового шифрування, який відрізняється від відомих методів використанням набору латинських квадратів і особливістю розгортання секретного ключа з використанням блоків відкритого повідомлення, що забезпечує потрібний рівень криптографічної стійкості і зменшення апаратних витрат.

Практична цінність отриманих результатів полягає у тому, що запропоновані апаратні рішення, порівняно з відомими апаратними засобами, мають складність від 1,06 до 15,8 разів меншу для пристроїв потокового шифрування та від 1,15 до 17,5 разів меншу для пристроїв блокового шифрування.

1 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Огляд блокового шифрування

На сьогодні реалізовано достатньо багато стійких блокових шифрів. Майже усі відомі шифри беруть до використання певний набір бієктивних (оборотних) алгебраїчних повторень [1].

Характерною ознакою блокових алгоримів шифрування є те, що під час своєї роботи вони виконують спотворення блоку вхідної інформації визначеної довжини і мають блок результату такого самого ж розміру, але недоступний для сторонніх осіб для прочитання, що не мають секретного ключа [1].

Роботу блокових шифрів можна показати наступними функціями [2]:

- Z=EnCrypt(X, Key)
- X=DeCrypt(Z, Key)

На основі блокових шифрів реалізовано дуже багато криптографічних систем. Властивістю блочних шифрів можна назвати перевагу у швидкості обробки даних, що не можна сказати про асиметричні криптоалгоритми, які є досить повільними по своїм характеристикам. Статистична кореляція між бітами вихідного потоку блокового шифру береться до використання для вираховування підсумкових сум пакетів інформації і в хешуванні ключів [2].

Криптологічний алгоритм зветься бажано стійким, у випадку коли розпізнати зашифрований блок інформації можливо якщо здійснити перебір усіх можливих ключів, до того моменту, доки повідомлення не стане читабельним та зрозумілим. Так як по теорії ймовірності ключ який шукатимуть буде виявлено з ймовірністю 1/2 після здійснення перебирання половини усіх ключів, то для завдання злому безперечно стійкого криптоалгоритма з ключем розміру N буде необхідно приблизно 2N-1 перевірок. Спираючись на це, в будь-якому випадку стійкість блочного шифру буде залежити лише від розміру ключа і зростати експоненціально з її зростанням. Навіть якщо привести припущення, що процес перебору ключів проходить на підготовленій для цього багатопроцесорній системі, в якій дякуючи діагональному паралелізму на процес перевірки одного ключа буде йти лише єдиний такт, то на завдання злому ключа розміром 128 біт

сьогоднішній техніці буде необхідно не менше ніж 1021 рік. Звісно, все вище сказане пов’язане лише з ідеально стійкими шифрами, які, для прикладу, з значною упевненістю були приведені в таблиці вище про алгоритми [2].

На додаток до цієї умови до ідеально стабільних криптографічних алгоритмів висувається ще одна дуже важлива вимога, якій вони повинні відповідати. Якщо відомі початкові та зашифровані значення блоку, ключ, за допомогою якого було виконано це перетворення, також можна дізнатися лише шляхом повного пошуку. Ситуації, коли частина оригінального тексту, відома сторонньому спостерігачеві, є всюдисущою. Це можуть бути стандартні написи в електронних формах, фіксовані заголовки форматів файлів, досить довгі слова, які зустрічаються в тексті, або послідовність байтів. У світлі цієї проблеми описана вище вимога не є надмірною і також суворо дотримується надійних криптографічних алгоритмів, як перший [17]. Отже, на функцію стійкого блокового шифру Z=EnCrypt(X, Key) накладаються наступні умови [2]:

- Функція EnCrypt має бути оборотною.
- Не повинно існувати інших методів прочитання повідомлення X по відомому блоку Z, окрім як повним перебором ключів Key.
- Не повинно існувати інших методів визначення яким ключем Key було вироблено перетворення відомого повідомлення X в повідомлення Z, окрім як повним перебором ключів.

Більшість блочних шифрів є ітераційними. Це означає, що цей шифр перетворює блоки відкритого тексту постійної довжини в блоки зашифрованого тексту такої ж довжини за допомогою циклічно повторюваних оборотних функцій, відомих як круглі функції. Це пояснюється простотою і швидкістю виконання як програмних, так і апаратних реалізацій. Як правило, функції раунду використовують різні ключі, отримані від оригінального ключа: Ci = RKi (Ci-1), де Ci — значення блоку після і-го раунду, C0 = M — відкритий текст, Ki — ключ, який використовується в і-го раунду та отримано з початкового ключа K [17].

Розмір блоку n є фіксованим параметром блочного шифру, зазвичай 64 або 128 біт, хоча деякі шифри допускають кілька різних значень. Різні схеми шифрування дозволяють шифрувати відкритий текст будь-якої довжини. Кожен має певні характеристики: ймовірність помилки, простота доступу, вразливість до атак [2].

Як і всі шифри, алгоритми яких відомі, блочні шифри піддаються криптографічним атакам. Метою атаки є розробка алгоритму злому, більш ефективного, ніж повний пошук усіх можливих ключів. Якщо таке рішення знайдено, атака вважається успішною. У цьому випадку шифр порушується, якщо є атака, яка дозволяє зламати час, протягом якого інформація залишається актуальною, і така атака вигідна зловмиснику [17].

1.2 Огляд потокового шифрування

Серед систем, що виконують функцію криптовалюти, особливе місце займають потокові шифри, в яких дані подаються та обробляються у вигляді нескінченного потоку або послідовності, гіпотетично може бути нескінченною [18]. Спочатку потокові шифри будувалися за допомогою генераторів псевдовипадкових послідовностей (PVP), а шифртекст був результатом операції додавання двох модулів (XOR) відкритого повідомлення з послідовністю ключів (гамма) від генератора. Основним і основним елементом генератора ПВП є регістр зсувного зворотного зв'язку з лінійним зворотним зв'язком (РЛЗЗ). Переваги такого генератора численні [5]:

- досить потужна швидкодія криптографічних алгоритмів;
- використання лише найбільш простих математичних операцій додавання та множення,апаратно побудованих майже у всіх розрахункових пристроях;
- гарні криптографічні властивості (генеруюча послідовність має великий період та добрі статистичні показники);
- добре підходять для систем з низьким рівнем енергоспоживанням;
- легкість аналізу з використанням алгебраїчних методів за рахунок лінійної структури.

Однак, незважаючи на численні переваги використання RZLZ в потокових шифрах, вони мають ряд недоліків, обумовлених лінійною структурою. Вихідна послідовність від генератора на основі RLZZ може бути передбачуваною. Тому з часом були спроби диверсифікувати генератори PVP для підвищення криптографічної стабільності алгоритму шифрування. Так з'явилися різні генератори [18]:

- генератор з різним тактуванням регістрів зсуву;
- генератор з декількома регістрами зсуву;
- генератор з нелінійним перетворенням;
- генератор, заснований на управлінні синхросигналом та ін.

Одними з найпоширеніших генераторів є Stop-and-Go і Geffe [6]. Проста структура генератора Stop-and-Go використовує три RZLZZ (рис. 1.1) [6]. Це генератор з різними тактовими регістрами.

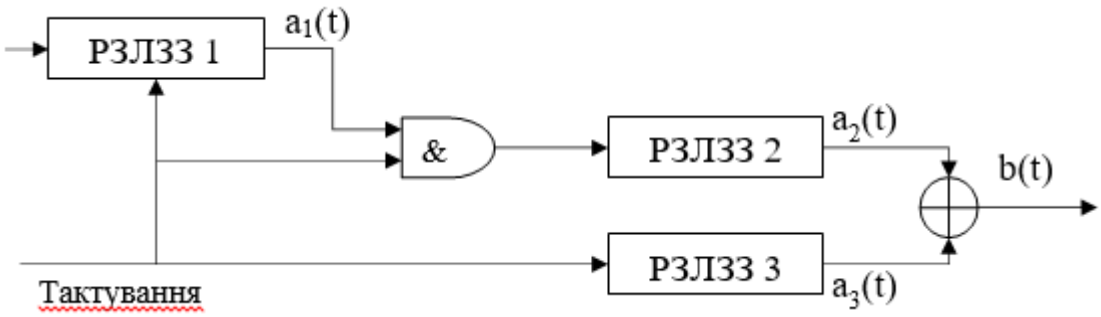


Рисунок 1.1 – Генератор Stop-and-Go

RZLZZ 1 керує тактовою частотою RZLZZ 2, так що RZLZZ 2 змінює свій стан у певний момент часу ti, за умови, що вихід RZLZZ 1 у момент часу ti-1 буде рівним 1. Через різний зсув тактових регістрів, наприклад генератор руйнує лінійні властивості звичайного генератора і підвищує криптографічну стабільність системи шифрування.

У структурі генератора Геффе використовуються три РЗЛЗЗ (рис. 1.2) [6]. Це генератор, який використовує нелінійне перетворення з комбінацією кількох RZLZZ.

Рисунок 1.2 – Генератор Geffe

RLZZ 2 і 3 служать входами мультиплексора, а RZLZZ 3 керує виходом мультиплексора. Довжини кожного з регістрів є попарно простими числами. Завдяки нелінійності цього генератора та комбінації кількох регістрів такий генератор може підвищити криптографічну стабільність системи, в якій він використовується.

Генератори PVP повинні бути розроблені так, щоб згенерована послідовність виглядала випадковою або непередбачуваною, і вони повинні мати високу криптографічну стабільність.

Потоковий шифр повинен забезпечувати високий рівень стабільності, мати необхідний рівень продуктивності та ефективно працювати на різних платформах [7]. Розробка потокових шифрів передбачає аналіз усіх можливих підходів до їх розробки. На думку Р. Рюппеля, було визначено чотири основних підходи до проектування потокових шифрів [7, 8]:

- системно-теоретичний підхід, заснований на створенні складної, раніше невідомої задачі для криптоаналітика;

- комплексно-теоретичний підхід, заснований на складній, але загальновідомій проблемі;

- підхід інформаційних технологій, заснований на спробі приховати відкритий текст від криптоаналітика, незалежно від того, скільки часу було витрачено на дешифрування, криптоаналітик не знайде правильного рішення;

- випадковий підхід, заснований на спробі створити тривимірну задачу. Таким чином, криптоаналітик намагається зробити вирішення проблеми дешифрування фізично неможливим.

При проектуванні систем потокового шифрування сформувався ряд вимог, до яких належать теоретичні критерії Р. Руппеля. Відповідно до цих вимог схеми генераторів ПВП повинні мати [8]:

- великі періоди вихідної послідовності;
- гарні статистичні показники властивостей вихідної ПВП;
- кожен біт потоку ключів має бути складним перетворенням більшості бітівключа;
- не лінійність або високу лінійну складність вихідної ПВП.

Ці вимоги забезпечуються ефективністю різних компонентів: обраної базової структури алгоритму, лінійного та нелінійного перетворення тощо. Хоча на сьогоднішній день немає теоретичних доказів необхідності та достатності цих вимог, але для створення криптографічно надійних систем потокового шифрування вони повинні бути виконані. Основна частина існуючих схем потокового шифрування створюється з окремих блоків, основними з яких є: регістри зсуву зворотного зв'язку, функції дискретної складності, запам'ятовуючі пристрої, вузли, що реалізують нерівномірний рух. Сучасні потокові шифри відповідно до eSTREAM поділяються на дві категорії: апаратно-орієнтовані та програмно-орієнтовані [9]. Найпопулярнішими серед апаратно-орієнтованих шифрів є: Trivium, Grain та інші.

Trivium — апаратно-орієнтований паралельний потоковий шифр, який має можливість програмної реалізації [10]. Він був створений як приклад для вивчення залежності спрощення потокового шифру без шкоди для безпеки, швидкості та гнучкості використання.

Алгоритм починає свою роботу [11] з ініціалізації завантаженням 80-бітового ключа K і 80-бітового вектора ініціалізації IV у початковий стан регістрів. Початковий стан цього шифру — три регістри зсуву, загальна довжина яких становить 288 біт. При нелінійній комбінації прямого і

зворотного зв’язку кожен такт змінює біти в регістрах зсуву. Використовується ключ ініціалізації K, а також вектор ініціалізації IV. Вони записуються тільки в два з трьох регістрів, потім виконується робота алгоритму, і за 4 × 288 = 1152 проходи. Ця кількість заповнень гарантує, що кожен біт початкового стану повністю залежить від кожного ключового біта та вектора ініціалізації.

Після завершення кроку ініціалізації Trivium генерує так звану послідовність ключів Z, яка є PVP. Генератор потоку використовує 15 біт із початкового стану 288 біт, щоб змінити 3 біти стану регістрів і обчислити один біт потоку ключів Zi (PVP). Це буде повторюватися, поки не буде згенеровано 264 біта PVP. Кожен біт Zi PVP проходить процедуру XOR з кожним наступним бітом тексту. Результатом такої процедури буде зашифрований текст. Щоб розшифрувати отриманий зашифрований текст, необхідно виконати ті ж дії, але в зворотному порядку. Тобто вам потрібно виконати процедуру XOR кожного біта Zi з кожним бітом зашифрованого тексту, і на виході буде відкритий текст.

Основна ідея Trivium — компактність у середовищі з обмеженими вхідними параметрами, енергоефективність на платформах з низьким рівнем енергоресурсів, а також швидкість у додатках, які вимагають високошвидкісного шифрування даних. Компактна реалізація цього шифру забезпечує біт-орієнтований підхід, що також призводить до використання нелінійного внутрішнього стану, щоб не витрачати всю побудовану нелінійність на виході генератора.

Gain — це апаратно орієнтований синхронний потоковий шифр з обмеженою кількістю елементів, споживанням пам’яті та потужністю [12, 13]. Алгоритм шифрування заснований на двох 80-бітних регістрах зсуву та нелінійній вихідній функції. Перший регістр має властивість функції лінійного зворотного зв'язку, другий - нелінійної функції зворотного зв'язку. Внутрішній стан шифру повністю визначається регістрами зсуву. Спочатку Gain приймає 80-бітний ключ K і 64-бітний вектор ініціалізації IV. Перш ніж ви зможете створити послідовність ключів Z (PVP), шифр повинен ініціалізувати свій стан.

Він завантажує біти K в регістр нелінійного зворотного зв’язку (NRR), а біти IV – в регістр лінійного зворотного зв’язку (NRR). Потім пропуски, що залишилися у функції лінійного зворотного зв’язку, заповнюються одиницями. Тоді код із 160 циклів працює без генерації ключового потоку, тобто сортує лише стани, але результат обробки надходить на вхід до NRZZ та LRZZ. Після процесу ініціалізації відбувається процес формування PVP.

Під час покоління Z Gain оновлює один біт LRZZ і один біт NRZZ. NRZZ оновлюється за допомогою нелінійної булевої функції «5 до 1» і 1 бітового входу, вибраного з LRZZ. LRZZ оновлюється за допомогою лінійної функції «6 до 1». Він повторює ці кроки для кожного біту шифрованого тексту Zi, створеного функцією нелінійного виведення. При цьому PVP, який стає зашифрованим текстом, формується при операції XOR з відкритим текстом.

Щоб збільшити швидкість генерації PVP, а також загальну роботу шифру, останні 15 бітів регістрів зсуву не використовуються у функціях зворотного зв’язку або на виході вихідної функції. Ця особливість шифру дозволяє йому працювати до 16 разів швидше, якщо доступна необхідна кількість апаратних ресурсів. Метою розробників Gain було створити шифр, щоб атака не могла здійснюватися на шифр швидше, ніж натискання клавіш, і їм це вдалося..

Найпопулярнішими серед програмно-орієнтованих потокових шифрів є: Rabbit, Sosemanuk та інші [9].

Rabbit — це високошвидкісний програмно-орієнтований потоковий шифр [14]. Основним компонентом шифру є генератор бітового потоку, який може перетворювати 128-розрядні повідомлення в зашифрований текст лише за одну ітерацію. Перевага цього шифру перед іншими полягає в тому, що він ретельно змішує свої внутрішні стани між двома послідовними ітераціями. Тобто кожні дві ітерації стан шифру змінюється. Функція shuffle повністю заснована на арифметичних операціях, які доступні для сучасних процесорів, тому блоки підстановки та таблиці пошуку не потрібні для реалізації шифру.

Rabbit використовує 128-бітний ключ і 64-бітний вектор ініціалізації. Внутрішній стан потокового шифру виглядає як послідовність бітів довжиною

513 біт. 512 з них розділені на 8 32-бітових змінних стану Xji і 8 32-бітних лічильників Cji. Крім того, Xji — змінна стану підсистеми j з відповідною ітерацією i, а Cji — змінна, що позначає відповідний лічильник змінних. 513-й біт — це біт передачі φ7, і, який необхідно зберігати між ітераціями. Цей біт ініціалізується нулем, а 8 станів змінних і 8 лічильників залежать лише від ключа, коли він ініціалізований.

Алгоритм ініціалізується шляхом розширення 128-бітового ключа до 8 станів змінних і 8 лічильників, щоб існувала взаємна однозначна відповідність між ключем, початковим станом змінної Xj, 0 і початковим значенням лічильника Cj0 [13]. Ключ K буде розділений на 8 підключів, стани змінних і лічильники, ініціалізовані підключами. Потім система виконується 4 рази відповідно до наступної функції стану, щоб зменшити кореляцію між ключовими бітами та бітами внутрішніх змінних стану. Нарешті, лічильники повторно ініціалізуються, щоб запобігти відновленню ключа шляхом інвертування системи лічильників. Після встановлення внутрішнього стану алгоритму починає працювати наступна функція стану, яка повертає нижній і верхній чотири байти 64 розрядного номера x з циклічним зсувом вліво. Після завершення роботи отриманий PVP повинен бути звільнений зі штатів. Звільнення відбувається шляхом виходу зі стану після кожної ітерації 128 біт згенерованої послідовності. Далі виконується операція XOR між випущеними бітами і повідомленням для отримання зашифрованого тексту, а для дешифрування, навпаки, - між випущеними бітами і шифротекстом.

Rabbit — це 128-бітний захист від зловмисників, метою яких є отримання унікального ключа. Якщо така атака відбувається відразу для кількох ключів, навіть якщо жоден з них не буде зламаний, безпека шифру знижується до 96 біт. Шифр дійсно стійкий до більшості криптографічних атак, але одним з його недоліків є те, що алгоритм забезпечує захист лише 128 біт за раз.

Sosemanuk – це відносно новий синхронно програмно-орієнтований потоковий шифр зі змінною довжиною ключа [14]. Довжина ключа може бути 128 і 256 біт. Шифр працює з 128-бітовим початковим заповненням, і, за

словами розробників алгоритму, будь-яка довжина ключа досягає 128-бітного захисту..

Алгоритм використовує деякі основні принципи шифру Snow2.0 і деякі трансформації блочного шифру Serpent [15]. Він використовує дві основні концепції: регістр зсуву лінійного зворотного зв’язку (LDL) і кінцевий автомат (CA). Дані, отримані за допомогою РЗЛЗЗ, надходять на вхід ЦС, де відбувається їх нелінійне перетворення. Потім S-box і XOR з відповідними значеннями регістра зміщення застосовуються до чотирьох вихідних значень CA. Вихідні значення раундів 12, 18 і 24 раунду перестановки Serpent використовуються для введення початкового стану: встановленого стану та регістрів CA.

Під час реалізації цього алгоритму розробники Sosemanuk передбачили можливість різноманітних атак на шифр, а тому забезпечили його захист і зробили шифр стійким до цих атак. Однак через кілька років з'явилися нові атаки, до яких шифр міг бути нестійким.

Проаналізувавши деякі потокові шифри, можна скласти таблицю 1.2 із порівняльними властивостями (параметрами безпеки) шифрів.

Таблиця 1.2 – Порівняння властивостей потокових шифрів

Назва властивості (параметру безпеки) потокового шифру	Назва шифру			
	Trivium	Gain	Rabbit	Sosemanuk
Довжина ключа, біт	80	80	128	128/256
Програмно орієнтований	-	-	+	+
Апаратно орієнтований	+	+	-	-
Генератор ПВП з нелінійним перетворенням	+	+	+	-
Генератор ПВП з декількома регістрами зсуву	+	+	-	+
Генератор ПВП з різним тактуванням регістрів зсуву	+/-	-	-	-
Стійкість до атаки встановлення ключа	+/-	+/-	+	+/-

Стійкість до атаки послідовного перебору	-	+/-	+	+
Стійкість до атаки Guess and Determine	+	+	+	+/-
Стійкість до атаки Guess and Verify	+	+	+	+
Стійкість до алгебраїчних атак	+	+	+/-	+
Стійкість до кореляційних атак	+	+	+/-	+

Отже, проаналізувавши генератори PVP та потокові шифри, ми можемо сказати, що наведені вище шифри дійсно гарні для захисту даних. Вони мають відносно просту реалізацію і стабільний захист, що дозволяє використовувати шифри в різних сферах. Проте всі вони стикаються з проблемами криптографічних атак. Хоча ідеї PVP-генераторів досить непогані у своїй, навіть найпростішій реалізації. Проте сучасні алгоритми шифрування потребують удосконалення, з одного боку, а концепція псевдоневизначеної моделі перетворення дозволяє це зробити, з іншого боку..

1.3 Визначення латинського квадрату

Латинський квадрат – це квадратна матриця порядку *n*, де кожний рядок і стовпчик якої являється перестановкою елементів кінцевої множини *S*, яка складається з *n* елементів. При чому в кожному стовпці та у кожному рядку кожний елемент зустрічається тільки один раз. Приклад такого латинського квадрата представлено нижче [16]:

$$\mathrm{L} = \begin{pmatrix} 0 & 1 & \cdots & n-2 & n-1 \\ 1 & 2 & \cdots & n-1 & 0 \\ \vdots & & \ddots & & \vdots \\ n-2 & n-1 & \cdots & n-4 & n-3 \\ n-1 & 0 & \cdots & n-3 & n-2 \end{pmatrix}$$

Припустимо, що ми маємо множину *S*, яка складається з наступних елементів: *S = {A, B, C}*. Тоді латинський квадрат буде мати наступний вигляд:

$$\mathrm{L} = \begin{pmatrix} A & C & B \\ C & B & A \\ B & A & C \end{pmatrix}$$

Цей латинський квадрат задається формулою $L(x,y) = x + y$, де x та y суть «номеру» рядка та стовпчику квадрата, $x, y \in \Omega = \{0,1,\dots,n-1\}$, і під складанням розуміється складання по модулю n (можна сказати, що формула $L(x,y) = x + y$) задає латинський квадрат над абелевой групою Z_n). Довільний латинський квадрат порядку n над групою Z_n задається формулою $L(x,y) = x + y + f(x,y)$, де f – це деяка функція $Z_n * Z_n \to Z_n$ [16].

У теперішні часи у якості множини M зазвичай береться множина натуральних чисел від 1 до n, однак Леонард Ейлер (1707 – 1783) використав букви латинського алфавіту, звідки й отримали свою назву латинські квадрати [16].

Відмітимо, що багато різних підходів до вивчення і формування латинських квадратів сходять до Ейлера. Він будував латинські квадрати з латинських прямокутників. Важливою віхою у вивченні латинських квадратів була робота А. Келлі, у якій він привів формулу для числа таких прямокутників з двох рядків. Наступне просування – виведення формули для числа прямокутників з трьох рядків – відбулось у 1953 році [16].

Формула для розрахунку числа $L(n)$ латинських квадратів порядку n не знайдена. У таблиці 1.1 приведено відомі точні значення $L(n)$ на сьогоднішній день [16]:

Таблиця 1.1 – Число латинських квадратів

2	
576	
161280	Euler (1682)
812851200	Frolov (1890)
61479419904000	Sade (1948)
108776032459082956800	Wells (1967)

552475149615689284253122560 0	Bammel (1975)
998243765821303987172506475692032000 0	Rogoyski (1995)
776966836171770144107444346734230682311065600000	McKay (2005)

Найкращі оцінки для *L(n)* дає наступна формула:

$$\prod_{k=1}^{n}(k!)^{n/k} \leq L(n) \leq \frac{(n!)^{2n}}{n^{n^2}}$$

1.4 Використання латинських квадратів в криптографії

Латинські квадрати знаходять застосування в комбінаториці, алгебрі (вивчення латинських квадратів тісно пов'язано з вивченням квазігруп), теорії кодів, статистиці та в багатьох інших областях.

Вперше в криптографії латинський квадрат був використаний в шифрі Тритемія [6]. Він побудував таблицю, яка відповідає таблиці Келі групи (Z_{26}, +), і використав її для багатоалфавітного шифрування. У такому шифруванні перша буква відкритого тексту шифрується першим алфавітом, тобто першим рядком таблиці, друга буква – другим алфавітом і так далі. Згодом цей шифр удосконалив Дж. Белазо, який придумав пароль. В сукупності з ідеєю Л. Б. Альберті використовувати довільний алфавіт це привело до появи нового шифру на основі квазігрупи, який став важливою віхою на шляху розвитку криптографії [7].

Значення латинських квадратів для криптографії ілюструє теорема Шеннона, у відповідності з якою єдиними досконалими шифрами являються шифри гамування, накладання гами в яких визначається латинським квадратом [8]. У ряду прикладів застосування латинських квадратів для побудови потокових шифрів необхідно виділити запропонований у 2005 році шифр Edon80, який добрався до третього туру конкурсу ESTREAM. Розробники шифру з 576 існуючих латинських квадратів 4-го порядку ретельно вибрали 4, на основі

яких в криптосхемі будується конвеєр з 80 латинських квадратів, для вироблення гами [9].

При розробці блокового шифру IDEA автори використовували три квазігрупи, які відповідають операціям додавання за модулем 2, додавання за модулем 2^{16} і множення за модулем $2^{16} + 1$. При цьому високі криптографічні властивості шифру були обґрунтовані тим, що серед відповідних квазігруп перша та друга не ізотопні, перша та третя не ізотопні, а єдиною ізотопією між другою і третьою квазігрупами являється функція логарифму [10].

Розроблене в 2008 році сімейство Edon-R для участі у конкурсі SHA-3 на новий американський стандарт не пройшло в другий тур, але цікаве тим, що в основі конструкції лежить побудова і використання некомутативної, неасоціативної та нелінійної квазігрупи. Автори використовували квазігрупи порядків 2^{256} та 2^{512}, ізотопні групам складання в 8-вимірних векторних просторах над відповідними полями, і два ортогональних латинських квадрати 8-го порядку [11].

Латинські квадрати знайшли своє застосування для побудови схем поділу секрету. Можна побудувати схему поділу секрету у якій секретним ключем буде латинський квадрат L порядку n. Так як, латинський квадрат вважається секретним ключем, то він і буде залишатись приватним. Однак порядок латинського квадрату n робиться загальнодоступним. Протокол виконується у наступні 4 пункти [12]:

- обирається латинський квадрат L порядку n (порядок n робиться публічним але латинський квадрат L зберігає секретність і приймається за ключ);
- визначається множина S, яка є об'єднанням певної кількості критичних множин в L;
- кожний елемент, який належить множині S, приватно розподіляється учаснику;
- коли група учасників, чиї частки містять критичні множини збирається разом, вони можуть реконструювати латинський квадрат L звідки і отримується секретний ключ.

Латинські квадрати використовуються у протоколі з нульовим розголошенням. Кожний учасник a має відкритий ключ, яким являються два ізотопних латинських квадрата L_a та L_b. Секретним ключом являється ізотопія між цими двома латинськими квадратами. Для аутентифікації учасник протоколу, який доводить свою достовірність, багаторазово виробляє з L_a випадковим чином ізотопний йому латинський квадрат H, направляє його учаснику який перевіряє достовірність першого учасника, і доводить йому в залежності від питання, що H ізотопний L_a або H ізотопний L_b [13].

Латинський квадрат лежить в основі конструкції, яку було запропоновано у 2005 році, у якості однонаправленої функції. Там введено поняття *e*-перетворення з лідером l: вектор A = (a_0, a_1, …, a_{t-1}) воно перетворює за допомогою квазігрупи (Q,*) в вектор B = (($l*a_0$),(($l*a_0$)*a_1), …, (($l*a_0$)*a_1)*…*a_{t-1}). Після цього за допомогою e-перетворення вводяться функції R_1 та R_2 [14]:

$$R_1(A) = e_{a_{t-1}}(\dots(e_{a_1}\left(e_{a_0}(A)\right)\dots),$$

$$R_2(A) = e_{a_{t-1}}(\dots(e_{a_1}\left(e_{a_0}(e_{a_{t-1}}(\dots(e_{a_1}(e_{a_0}(A)\right)\dots).$$

Доведена теорема про те, що якщо квазігрупа (Q, *) неасоціативна і некомутативна, то для обчислення оберненої до R_1 функції необхідно $O(n^{[t/3]})$ звернень в пам’ять, тобто до латинського квадрату який відповідає квазігрупі Q. А для обчислення оберненої до R_2 функції потребується $O(n^t)$ звернень у пам’ять [14].

Також була запропонована схема аутентифікації повідомлень. Повідомлення в цій схемі ділиться на блоки по t знаків. Для кожного j-го блоку виду $a_1a_2\dots a_t$ обчислюється у квазігрупі (Q,*) значення b_j = (…((a_1 * a_2) * a_3) * … * a_{t-1}) * a_t, яке являється j-м елементом підпису [15].

Досить популярним прикладом використання латинських квадратів в шифруванні являється шифр Віженера. Даний шифр є методом поліалфавітного шифрування буквенного тексту з використанням ключового слова. Вперше цей метод був описаний Джовані Беллазом у 1553 році. Шифр досить легкий до

розуміння та реалізації але являється недоступним для простих способів криптоаналізу [16].

Шифр Віженера складається з послідовності декількох шифрів Цезаря з різними значеннями зсуву. Для зашифрування може бути використана таблиця яку називають квадрат Віженера. Відповідно до латинського алфавіту таблиця Віженера складається з рядків по 26 символів, при чому кожний наступний рядок зсувається на декілька позицій. Таким чином, в таблиці отримуються 26 різних шифрів Цезаря. На кожному етапі шифрування використовуються різні алфавіти, які обираються в залежності від символу ключового слова [16].

Шифрування Віженера можна представити у такому вигляді:

$$\text{шифрування - } c_j = m_j + k_j (mod\ n),$$

$$\text{розшифрування - } c_j = m_j - k_j (mod\ n),$$

де m_j – букви відкритого тексту; k_j – букви ключа; n – кількість букв алфавіт [16].

Для спрощення алгоритму використаємо таблицю, яка представляє собою латинський квадрат 3x3, який назвемо головним (рис. 1.2):

I	II	III
II	III	I
III	I	II

Рисунок 1.2 – Латинський квадрат 3x3

Кожний елемент головного латинського квадрату представляє собою латинські квадрати 11x11, які будуть називатись дочірніми. Приклад такого квадрату представлено на рис. 1.3.

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й
Б	В	Г	Д	Е	Ё	Ж	З	И	Й	А
В	Г	Д	Е	Ё	Ж	З	И	Й	А	Б
Г	Д	Е	Ё	Ж	З	И	Й	А	Б	В
Д	Е	Ё	Ж	З	И	Й	А	Б	В	Г
Е	Ё	Ж	З	И	Й	А	Б	В	Г	Д
Ё	Ж	З	И	Й	А	Б	В	Г	Д	Е
Ж	З	И	Й	А	Б	В	Г	Д	Е	Ё
З	И	Й	А	Б	В	Г	Д	Е	Ё	Ж
И	Й	А	Б	В	Г	Д	Е	Ё	Ж	З
Й	А	Б	В	Г	Д	Е	Ё	Ж	З	И

Рисунок 1.3 – Дочірній латинський квадрат 11x11

Процес шифрування ділиться на такі етапи [16]:

1. Шляхом повторення ключа необхідну кількість разів, розмірність ключа збільшується до розмірності повідомлення. Кожній букві повідомлення ставиться у відповідність буква ключа;

2. За допомогою першого рядка головного латинського квадрату визначається до якого дочірнього латинського квадрату відноситься буква тексту;

3. Також визначається до якого дочірнього квадрату відноситься буква ключа;

4. Визначається порядковий номер букви повідомлення в дочірньому латинському квадраті;

5. До порядкового номеру букви ключа додається порядковий номер букви повідомлення за модулем 11;

Кроки 2-5 повторюються до тих пір, поки все повідомлення не буде зашифроване.

2 РОЗРОБКА ГЕНЕРАТОРУ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

2.1 Огляд методів генерування псевдовипадкових чисел

Розглянемо алфавіт (тобто кінцеву множину) Q та позначимо $Q^+ = \{a_0a_1 \dots a_n | a_i \in Q\}$ множиною всіх непорожніх слів (тобто кінцевих рядків) утворених елементами Q. Нехай * буде операцією на множині Q заданою латинським квадратом (рис. 2.1), тобто розглянемо латинський квадрат (Q, *).

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

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

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

Рисунок 2.1 – Приклади латинських квадратів

Для кожного $a \in Q$ визначимо функцію $e_{a,*} : Q^+ \to Q^+$ наступним чином. Нехай $a_i \in Q, \alpha = a_1a_2 \dots a_n$. Тоді перетворення для кожного елементу можна представити наступною формулою:

$$e_{a,*}(\alpha) = b_1b_2 \dots b_n \leftrightarrow b_1 = a * a_1, b_2 = b_1 * a_2, \dots, b_n = b_{n-1} * a_n \quad (2.1)$$

Функція $\boldsymbol{e_{a,*}}$ називається е-перетворенням $\mathbf{Q^+}$ на основі операції ***** з лідером **a**.

Графічне представлення е-перетворення можна побачити на рис. 2.2:

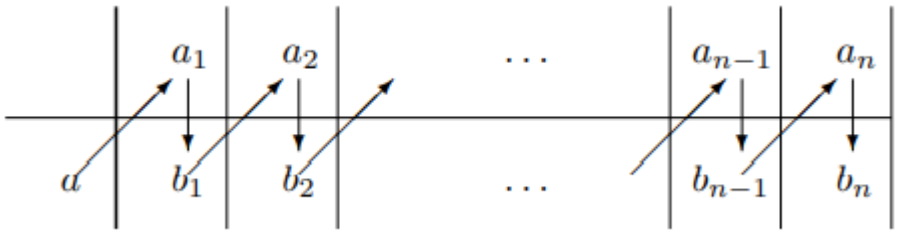


Рисунок 2.2 – Графічне представлення функції е-перетворення

Розглянемо приклад. Нехай дано множину Q = {0, 1, 2, 3} та латинський квадрат на основі цієї множини (Q, *) (див. рис. 2.1).

Нехай рядок α = 1 0 2 1 0 0 0 0 0 0 0 0 0 1 1 2 1 0 2 2 0 1 0 1 0 3 0 0, а лідером обрано 0. Тоді, за допомогою функції $\boldsymbol{e_{a,*}}$ виконаємо перетворення вхідного рядку:

$$b_1 = a * a_1 = 0 * 1 = 1$$
$$b_2 = b_1 * a_2 = 1 * 0 = 3$$
$$b_3 = b_2 * a_3 = 3 * 2 = 2$$
$$b_4 = b_3 * a_4 = 2 * 1 = 2$$
$$b_5 = b_4 * a_5 = 2 * 0 = 1$$
$$b_6 = b_5 * a_6 = 1 * 0 = 3$$
$$b_7 = b_6 * a_7 = 3 * 0 = 0$$
$$b_8 = b_7 * a_8 = 0 * 0 = 2$$
$$b_9 = b_8 * a_9 = 2 * 0 = 1$$
$$b_{10} = b_9 * a_{10} = 1 * 0 = 3$$
$$\dots$$
$$b_{28} = b_{27} * a_{28} = 3 * 0 = 0$$

Отже, у результаті буде отримано такий перетворений рядок $e_{a,*}(\alpha)$:

$e_{0,*}(\alpha)$ = 1 3 2 2 1 3 0 2 1 3 0 2 1 0 1 1 2 1 1 1 3 3 0 1 3 1 3 0.

Нижче представляється чотири послідовні застосування е-перетворення для кожного з яких значення лідеру буде рівне 0. Результат цих перетворень можна побачити на рис. 2.2:

$e_{0,*}(\alpha)$ = 1 3 2 2 1 3 0 2 1 3 0 2 1 0 1 1 2 1 1 1 3 3 0 1 3 1 3 0,

$e^2_{0,*}(\alpha)$ = 1 2 3 2 2 0 2 3 3 1 3 2 2 1 0 1 1 2 2 2 0 3 0 1 2 2 0 2,

$e^3_{0,*}(\alpha)$= 1 1 2 3 2 1 1 2 0 1 2 3 2 2 1 0 1 1 1 1 3 1 3 3 2 3 0 0,

$e^4_{0,*}(\alpha)$= 1 0 0 3 2 2 2 3 0 1 1 2 3 2 2 1 0 1 0 1 2 2 0 3 2 0 2 1.

Операцій для виконання перетворення може бути безліч і кожна буде представлятись окремим латинським квадратом.

Тепер розглянемо ще один підхід до генерування псевдовипадкових послідовностей.

Розглянемо алфавіт (тобто кінцеву множину) Q та позначимо $Q^+ = \{a_0 a_1 \dots a_n | a_i \in Q\}$ множиною всіх непорожніх слів (тобто кінцевих рядків) утворених елементами Q. Припустимо, що множина Q представлена латинським квадратом. Тоді множина повідомлень M являється $C = Q^+$. Зафіксуємо елемент $l \in Q$, який називається лідером та визначимо елементарне перетворення $E_l = M \to C$ по наступному правилу:

$$E_l(x_1 x_2 \dots x_t) = y_1 y_2 \dots y_t \tag{2.2}$$

$$\text{де } y_i = \begin{cases} l * x_i, & i = 1 \\ y_{i-1} * x_i, & 2 \le i \le t \end{cases}$$

Наведемо приклад застосування елементарних перетворень для латинського квадрату 4 порядку (рис. 2.3):

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

Рисунок 2.3 – Латинський квадрат четвертого порядку

Далі будемо брати множину повідомлень M, лідера l та виконувати степені E_l^k. У результаті буде отримано послідовність перетворень:

$$M = 00000000000000000000, l = 0, k = 5;$$

$$E_0^1(M) = 13013013013013013013;$$

$$E_0^2(M) = 03003003003003003003;$$

$$E_0^3(M) = 11303011303011303011;$$

$$E_0^4(M) = 00303000303000303000;$$

$$E_0^5(M) = 13220130303013220130;$$

$$M = 1111111111111111111, l = 1, k = 5;$$

$$E_1^1(M) = 21212121212121212121;$$

$$E_1^2(M) = 00210021002100210021;$$

$$E_1^3(M) = 30213021302130213021;$$

$$E_1^4(M) = 13120100302113120100;$$

$$E_1^5(M) = 20022130302120022130;$$

По такому ж принципу окремо виконується перетворення для множин повідомлень які заповнені числами 2 та 3. Кількість ітерацій може бути будь-якою.

2.2 Розробка методу генерування псевдовипадкових чисел

Метод генерування псевдовипадкових чисел буде реалізовано на основі латинських квадратів. Тому, розглянемо принцип їх побудови (рис. 2.1):

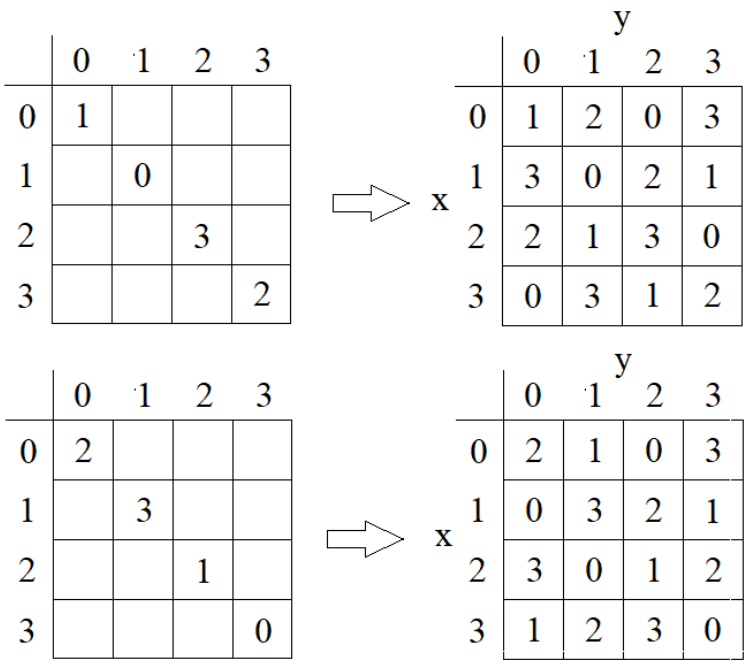


Рисунок 2.4 – Принцип побудови латинських квадратів

Для побудови латинського квадрату необхідно спочатку заповнити елементи діагоналі таким чином, щоб значення елементу на діагоналі не дорівнювало значенню поточного рядка та стовпчика. Так, для комірки (0; 0) може бути обрано значення 1, 2 або 3. Після отримання елементів діагоналі відповідно до неї необхідно заповнити квадрат значеннями так, щоб кожний

елемент латинського квадрату не повторювався більше одного разу в своєму рядку та стовпчику.

Для побудови генератору псевдовипадкових чисел буде необхідно 7 різних квадратів (рис. 2.5):

x \ y	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

x \ y	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

x \ y	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

x \ y	0	1	2	3
0	2	1	3	0
1	3	0	2	1
2	0	3	1	2
3	1	2	0	3

x \ y	0	1	2	3
0	0	1	2	3
1	2	3	0	1
2	3	2	1	0
3	1	0	3	2

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

Рисунок 2.5 – Латинські квадрати для генератору ПВЧ

Структурна схема генератору псевдовипадкових чисел зображено на рис. 2.6:

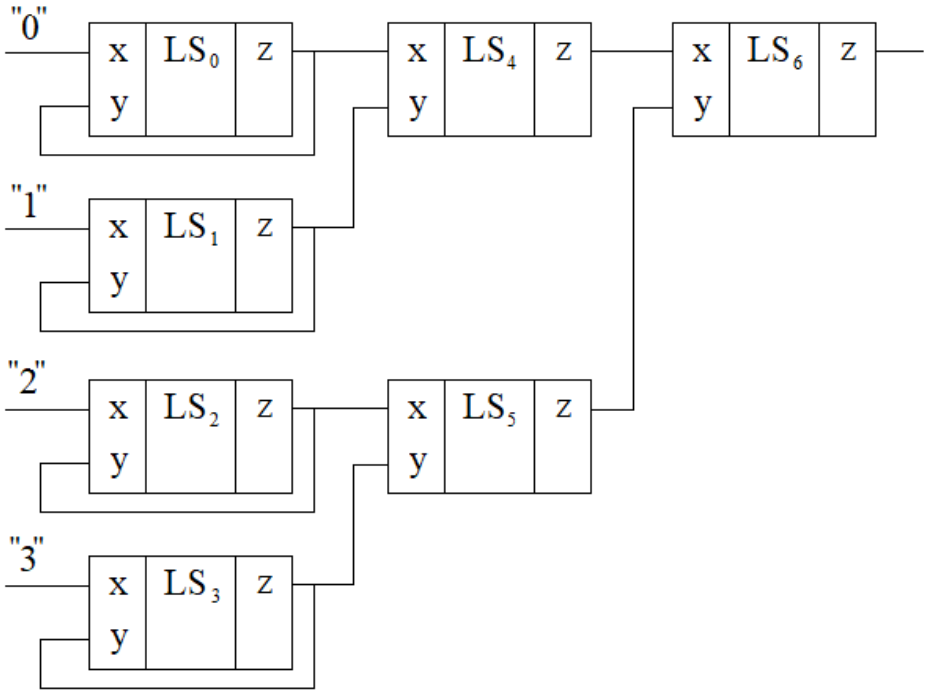


Рисунок 2.6 – Структурна схема генератору псевдовипадкових чисел

Кожний елемент LS_n схеми вище реалізує латинський квадрат. Елементи LS_{0-3} використовуються для реалізації функції перетворення (2.1). Вхід **x** приймає константне значення що являється лідером. Вхід **y** приймає значення, яке є результатом що надходить з виходу **z**.

Пари елементів LS_{0-1} та LS_{2-3} передають отриманий результат на елементи LS_4 та LS_5 відповідно. За таким же принципом, результат, який надходить з цих елементів передається на елемент LS_6 де уже формується значення елементу псевдовипадкової послідовності.

На рис. 2.7 зображено графічне позначення генератору псевдовипадкових чисел:

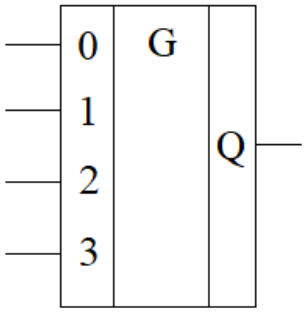


Рисунок 2.7 – Графічне позначення генератору псевдовипадкових чисел

Так як, генератор має бути секретним, далі необхідно побудувати криптографічний генератор псевдовипадкових чисел. Схема такого генератору зображено на рис. 2.8:

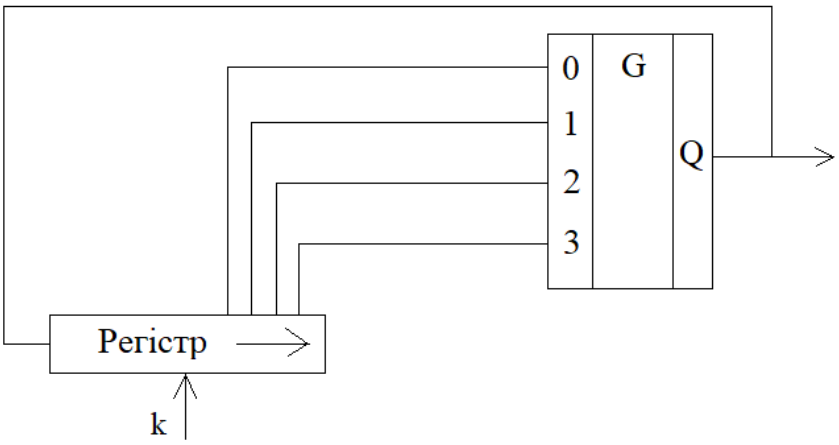


Рисунок 2.8 – Структурна схема криптографічного генератору псевдовипадкових чисел

Секретний ключ **k** довжиною 64 біт буде подаватись на регістр. Кожний вхід генератору **G** приймає 2-розрядне значення, тобто 2 біта. Таким чином за один крок буде оброблятись 1 байт секретного ключа. На виході **Q** буде отримано 2-рорзрядний результат який далі вштовхується у регістр. Відповідно, 2 розряди секретного ключа будуть виштовхуватись із регістру. Таким чином за 32 кроки буде сформовано псевдовипадкову послідовність.

2.3 Оцінка складності апаратної реалізації

Генератор псевдовипадкових чисел складається з 7 латинських квадратів. Також у схемі присутній регістр на який буде подаватись ключ.

Для оцінки складності кожного латинського квадрату буде використано мінімізацію функцій які реалізує той чи інший латинський квадрат.

Розглянемо латинський квадрат LS_0 представлений на рис. 2.9:

x \ y	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

Рисунок 2.9 – Зображення першого латинського квадрату

Побудуємо таблицю істинності для даного квадрату. Для цього заповнимо таблицю значеннями від 0 до 15 у двійковій системі числення використовуючи сторони латинського квадрату X та Y як чотири розряди. Для заповнення Z використовуються відповідні значення X та Y. Таким чином, наприклад, для $x_1 = 1$, $x_0 = 0$ та $y_1 = 0$, $y_0 = 1$ маємо наступне:

$$10_2 = 3_{10}$$

$$01_2 = 1_{10}$$

За допомогою отриманих значень знайдемо Z використовуючи латинський квадрат (див. рис. 2.9):

$$3_{10} = 10_2$$

Таким чином маємо $z_1 = 1$, $z_0 = 0$. По аналогії необхідно заповнити значення Z для всіх значень N.

Таблиця 2.1 – Таблиця істинності для латинського квадрату LS_0

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	0	1
1	0	0	0	1	1	0
2	0	0	1	0	0	0
3	0	0	1	1	1	1
4	0	1	0	0	1	1
5	0	1	0	1	0	0
6	0	1	1	0	1	0
7	0	1	1	1	0	1
8	1	0	0	0	1	0
9	1	0	0	1	0	1
10	1	0	1	0	1	1
11	1	0	1	1	0	0
12	1	1	0	0	0	0
13	1	1	0	1	1	1
14	1	1	1	0	0	1
15	1	1	1	1	1	0

Наступним кроком необхідно мінімізувати функції z_1 та z_0 використовуючи діаграму Вейча (рис. 2.10):

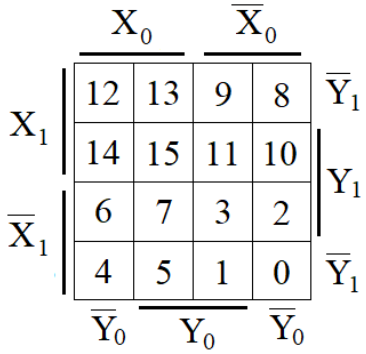


Рисунок 2.10 – Діаграма Вейча з номерами наборів

Розмітимо дану діаграму на основі z_1. Таким чином наприклад, для N = 7 значення $z_1 = 0$, тому комірка 7 на діаграмі Вейча пропускається. Далі візьмемо N = 10 для якого у нашому випадку $z_1 = 1$. У такому випадку 1 записується у комірку під номером 10. Нижче, на рисунку 2.11 наведено кінцевий результат заповнення таблиці:

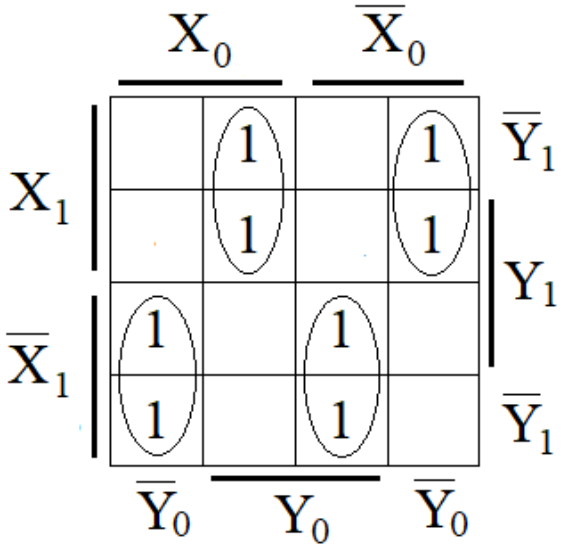


Рисунок 2.11 – Розмічена діаграма Вейча для функції z_1

Далі виконаємо власне мінімізацію функції. На діаграмі Вейча можна об'єднувати 2, 4, 8 одиниць які є сусідніми. На рисунку 2.11 було об'єднано по дві одиниці у чотирьох різних областях. Кожна пара одиниць у даному випадку належить хоч одній зоні яка покриває пару повністю. Таким чином нижня ліва пара повністю покривається зоною $\overline{x_1}, x_0$ та $\overline{y_0}$. Аналогічно визначаємо всі об'єднані пари одиниць:

$$z_1 = \overline{x_1}x_0\overline{y_0} + x_1x_0y_0 + x_1\overline{x_0y_0} + \overline{x_1x_0}y_0$$

Далі здійснимо спрощення функції. Наприклад, значення $\overline{x_1}$ та x_1 можна винести за дужки:

$$z_1 = \overline{x_1}(x_0\overline{y_0} + \overline{x_0}y_0) + x_1(x_0y_0 + \overline{x_0y_0}) \qquad (2.3)$$

Після отримання спрощеної функції оцінимо її складність за Квайном. Даний метод оцінки передбачає підрахування кількості входів всіх елементів пристрою. Відповідно до функції 2.3, маємо $\overline{x_1}$, $\overline{x_0}$ та $\overline{y_0}$ – логічна операція «НЕ», елемент

якої має один вхід. Також у функції представлено ряд логічних операцій «І» та «АБО», елементи яких мають по два входи. Маємо 6 елементів «І» та 3 елементи «АБО». Таким чином розрахуємо складність функції Z_1:

$$S_{z1} = 6*2 + 3*2 + 3 = 21 \quad (2.4)$$

По аналогії виконується мінімізація функції Z_0. Маємо розмічену діаграму Вейча представлену на рисунку 2.12:

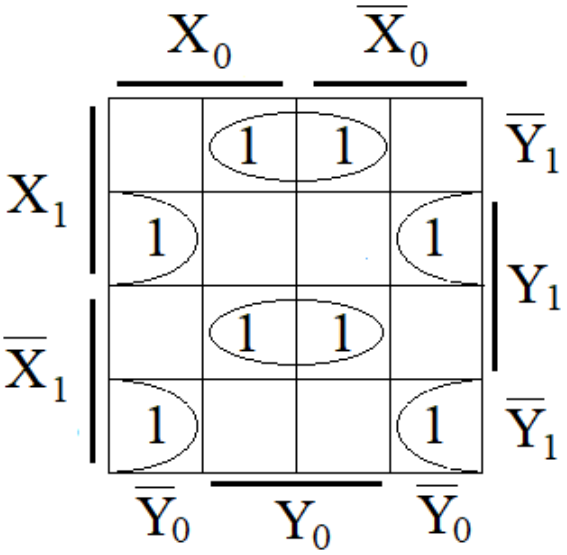


Рисунок 2.12 – Розмічена діаграма Вейча для функції z_0

З діаграми маємо функцію

$$z_0 = \overline{x_1 y_0 y_1} + \overline{x_1} y_0 y_1 + x_1 \overline{y_0} y_1 + x_1 \overline{y_0} y_1$$

яку приводимо до спрощеного вигляду шляхом виносу за дужки спільних множників:

$$z_0 = \overline{x_1}(\overline{y_0 y_1} + y_0 y_1) + x_1(\overline{y_0} y_1 + y_0 \overline{y_1}) \quad (2.5)$$

Результат спрощення функції z_0 аналогічний до фукнції z_1. Після підрахування складності маємо $S_{z0} = 21$.

Отже, складність реалізації латинського квадрату LS_0:

$$S_0 = S_{z0} + S_{z1} = 21 + 21 = 42$$

На рисунку 2.13 зображено структурну схему латинського квадрату.

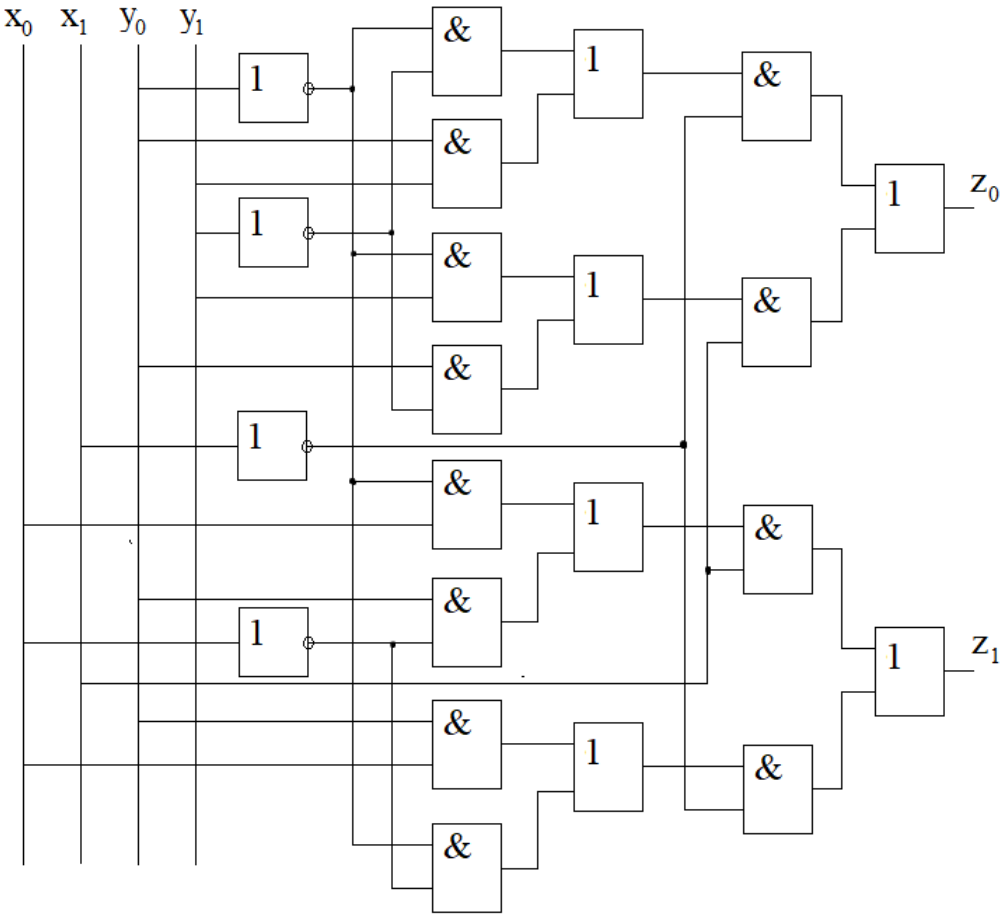


Рисунок 2.13 – Структура схема латинського квадрату LS_0

По аналогії проведемо оцінку складності для решти шести латинських квадратів які будуть реалізовані у генераторі.

Латинський квадрат LS_1 представлено на рис. 2.14:

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

Рисунок 2.14 – Латинського квадрат LS_1

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.2 – Таблиця істинності латинського квадрату LS_1

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	1	0
1	0	0	0	1	0	1
2	0	0	1	0	0	0
3	0	0	1	1	1	1
4	0	1	0	0	0	0
5	0	1	0	1	1	1
6	0	1	1	0	1	0
7	0	1	1	1	0	1
8	1	0	0	0	1	1
9	1	0	0	1	0	0
10	1	0	1	0	0	1
11	1	0	1	1	1	0
12	1	1	0	0	0	1
13	1	1	0	1	1	0
14	1	1	1	0	1	1
15	1	1	1	1	0	0

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.15:

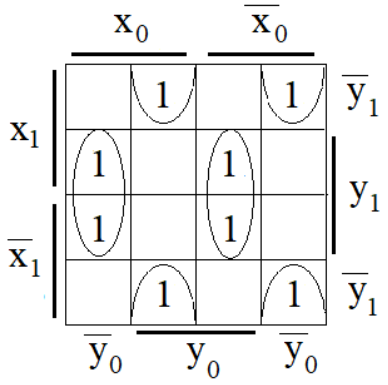


Рисунок 2.15 – Розмічена діаграми Вейча для функції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = x_0\overline{y_0}y_1 + \overline{x_0}y_0y_1 + x_0y_0\overline{y_1} + \overline{x_0y_0y_1} = x_0(\overline{y_0}y_1 + \overline{y_1}y_0) + \overline{x_0}(y_0y_1 + \overline{y_0y_1})$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 21$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.16:

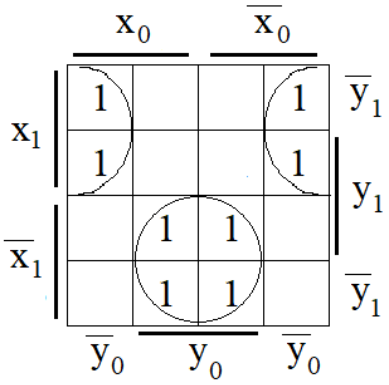


Рисунок 2.16 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = x_1\overline{y_0} + y_0\overline{x_1}$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 8$.

Загальна складність латинського квадрату $S_1 = S_{z0} + S_{z1} = 8 + 21 = 29$.

На рисунку 2.17 зображено структурну схему латинського квадрату.

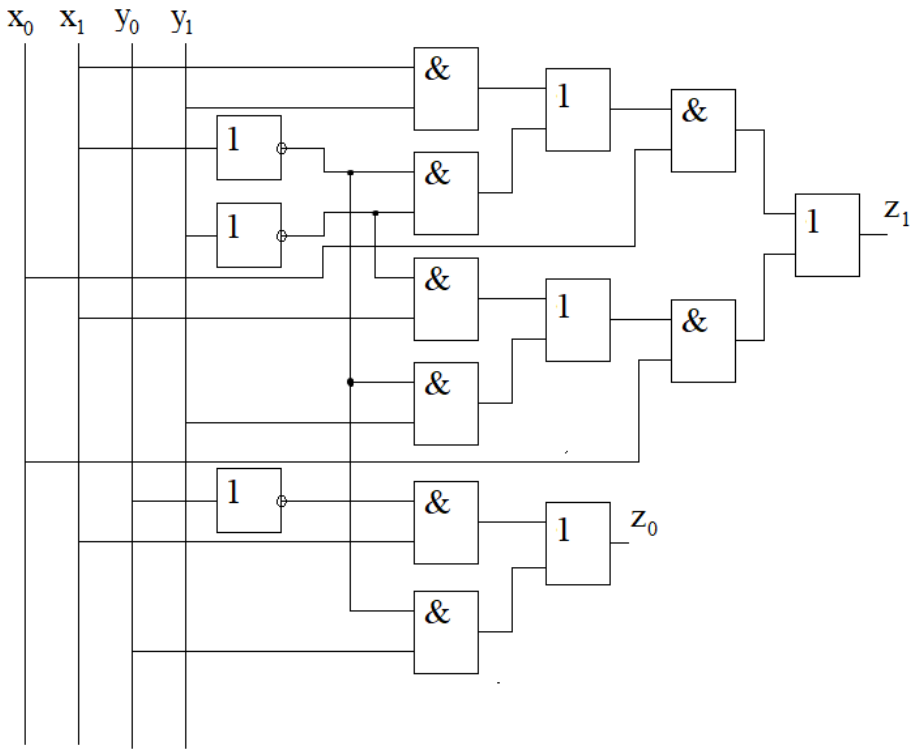


Рисунок 2.17 – Структурна схема латинського квадрату LS_1

Латинський квадрат LS_2 представлено на рис. 2.18:

x \ y	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

Рисунок 2.18 –Латинський квадрат LS_2

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.3 – Таблиця істинності латинського квадрату LS_2

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	1	1
1	0	0	0	1	0	0
2	0	0	1	0	0	1
3	0	0	1	1	1	0
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	0	1
10	1	0	1	0	0	0
11	1	0	1	1	1	1
12	1	1	0	0	0	0
13	1	1	0	1	1	1
14	1	1	1	0	1	0
15	1	1	1	1	0	1

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.19.

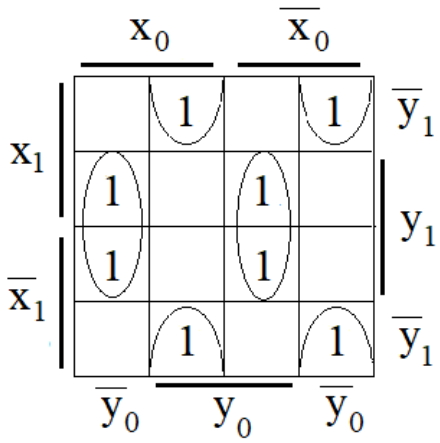


Рисунок 2.19 – Розмічена діаграми Вейча для функції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = x_0\overline{y_0}y_1 + \overline{x_0}y_0y_1 + x_0y_0\overline{y_1} + \overline{x_0y_0y_1} = x_0(\overline{y_0}y_1 + \overline{y_1}y_0) + \overline{x_0}(y_0y_1 + \overline{y_0y_1})$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 21$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.20:

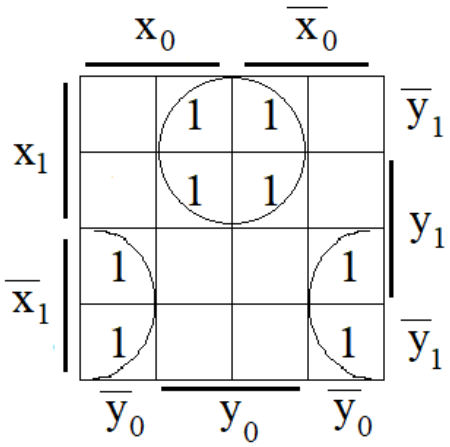


Рисунок 2.20 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = x_1y_0 + \overline{y_0x_1}$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 8$.

Загальна складність латинського квадрату $S_2 = S_{z0} + S_{z1} = 8 + 21 = 29$.

На рисунку 2.21 зображено структурну схему латинського квадрату.

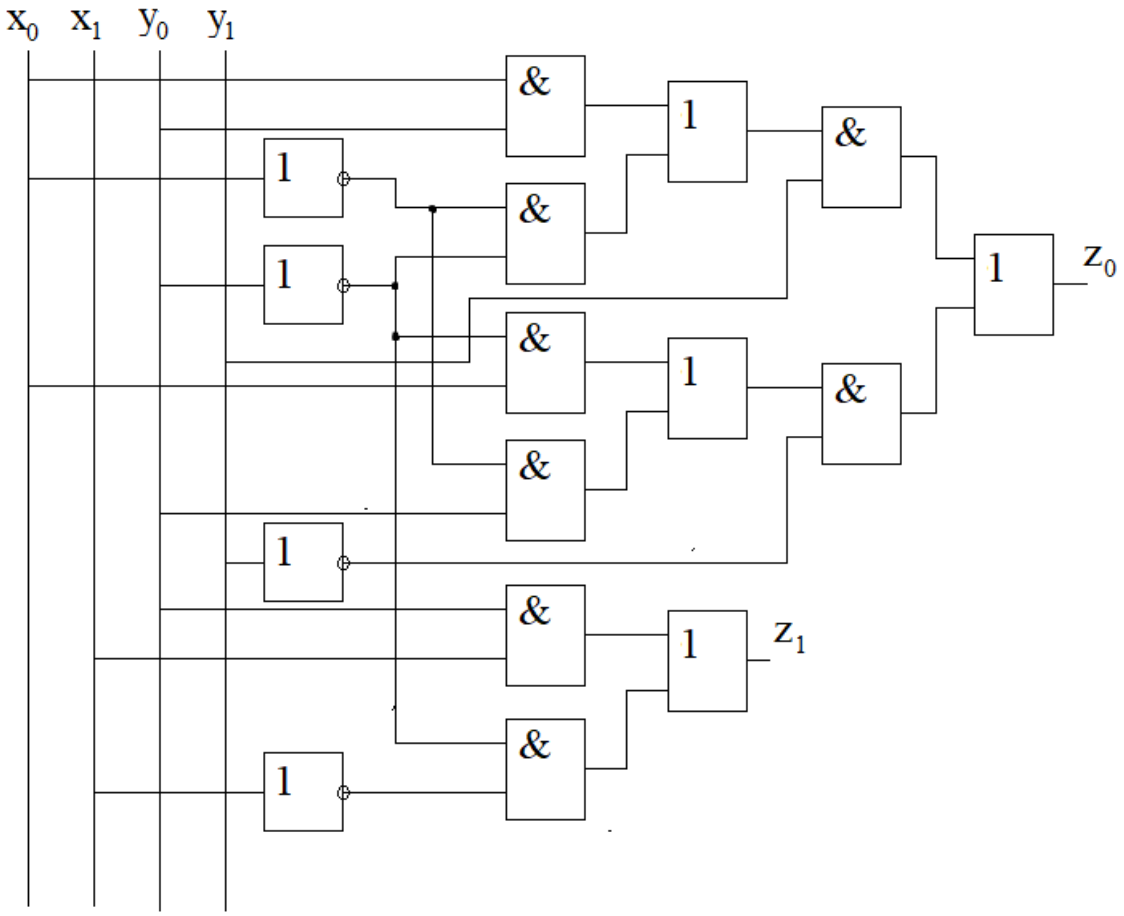


Рисунок 2.21 – Структурна схема латинського квадрату LS_2

Латинський квадрат LS_3 представлено на рис. 2.22:

x \ y	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

Рисунок 2.22 – Латинський квадрат LS_3

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.4 – Таблиця істинності для четвертого латинського квадрату

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	0	1
1	0	0	0	1	1	0
2	0	0	1	0	1	1
3	0	0	1	1	0	0
4	0	1	0	0	0	0
5	0	1	0	1	1	1
6	0	1	1	0	1	0
7	0	1	1	1	0	1
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	0	0
14	1	1	1	0	0	1
15	1	1	1	1	1	0

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.23:

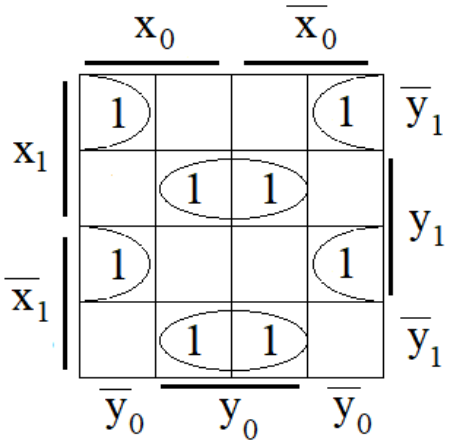


Рисунок 2.23 – Розмічена діаграми Вейча для функції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = x_1 y_0 y_1 + \overline{x_1} y_0 \overline{y_1} + x_1 \overline{y_1 y_0} + \overline{x_1 y_0} y_1 = y_0(x_1 y_1 + \overline{y_1 x_1}) + \overline{y_0}(x_1 \overline{y_1} + y_1 \overline{x_1})$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 21$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.24:

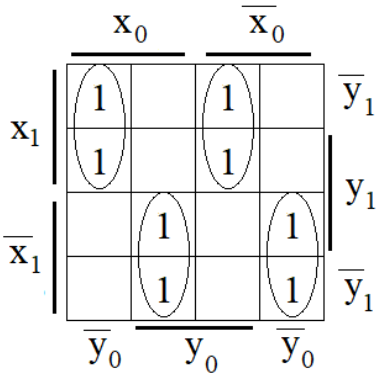


Рисунок 2.24 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = x_1 x_0 \overline{y_0} + x_1 \overline{x_0} y_0 + \overline{x_1} y_0 x_0 + \overline{x_1 y_0 x_0} = x_1(x_0 \overline{y_0} + \overline{x_0} y_0) + \overline{x_1}(y_0 x_0 + \overline{y_0 x_0})$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 21$.

Загальна складність латинського квадрату $S_3 = S_{z0} + S_{z1} = 21 + 21 = 42$.

На рисунку 2.25 зображено структурну схему латинського квадрату.

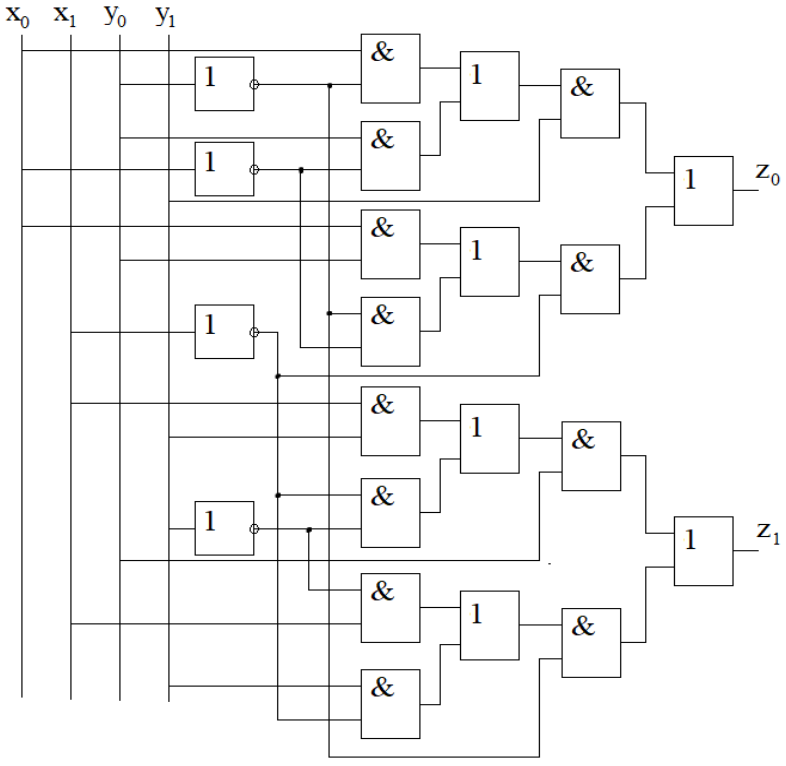


Рисунок 2.25 – Структурна схема латинського квадрату LS_3

Латинський квадрат LS_4 представлено на рис. 2.26:

x \ y	0	1	2	3
0	2	1	3	0
1	3	0	2	1
2	0	3	1	2
3	1	2	0	3

Рисунок 2.26 – Латинський квадрат LS_4

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.5 – Таблиця істинності латинського квадрату LS_4

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	1	0
1	0	0	0	1	0	1
2	0	0	1	0	1	1
3	0	0	1	1	0	0
4	0	1	0	0	1	1
5	0	1	0	1	0	0
6	0	1	1	0	1	0
7	0	1	1	1	0	1
8	1	0	0	0	0	0
9	1	0	0	1	1	1
10	1	0	1	0	0	1
11	1	0	1	1	1	0
12	1	1	0	0	0	1
13	1	1	0	1	1	0
14	1	1	1	0	0	0
15	1	1	1	1	1	1

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.27:

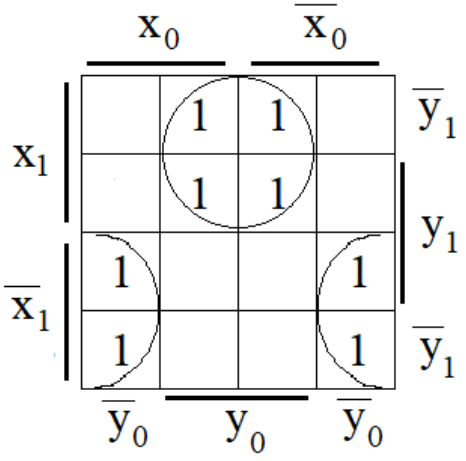


Рисунок 2.27 – Розмічена діаграма Вейча для фукнції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = x_1 y_0 + \overline{y_0 x_1}$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 8$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.28:

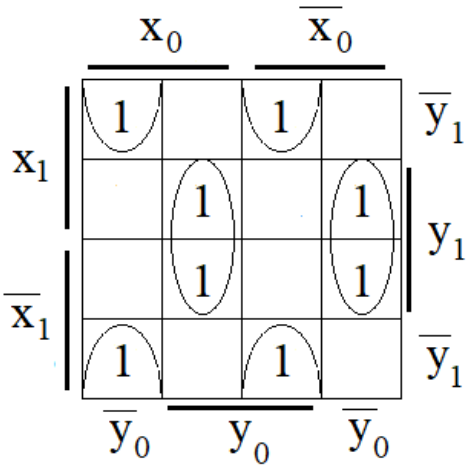


Рисунок 2.28 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = y_1 x_0 y_0 + y_1 \overline{x_0 y_0} + \overline{y_1 y_0} x_0 + \overline{y_1} y_0 \overline{x_0} = y_1 (x_0 y_0 + \overline{x_0 y_0}) + \overline{y_1} (\overline{y_0} x_0 + y_0 \overline{x_0})$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 21$.

Загальна складність латинського квадрату $S_4 = S_{z0} + S_{z1} = 21 + 8 = 29$.

На рисунку 2.29 зображено структурну схему латинського квадрату.

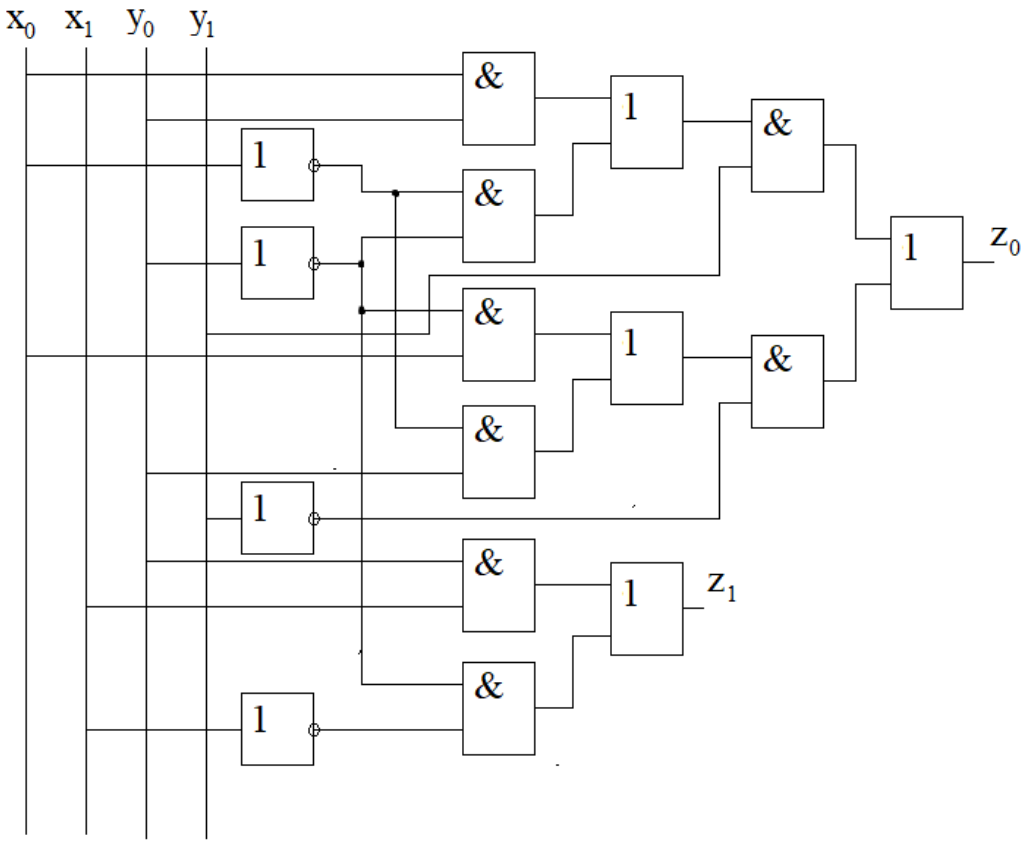


Рисунок 2.29 – Структурна схема латинського квадрату LS_4

Латинський квадрат LS_5 представлено на рис. 2.30:

x \ y	0	1	2	3
0	0	1	2	3
1	2	3	0	1
2	3	2	1	0
3	1	0	3	2

Рисунок 2.30 – Латинський квадрат LS_5

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.6 – Таблиця істинності латинського квадрату LS_5

N	x_1	x_0	y_1	y_0	z_1	z_0
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	1	0
5	0	1	0	1	1	1
6	0	1	1	0	0	0
7	0	1	1	1	0	1
8	1	0	0	0	1	1
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	0
12	1	1	0	0	0	1
13	1	1	0	1	0	0
14	1	1	1	0	1	1
15	1	1	1	1	1	0

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.31:

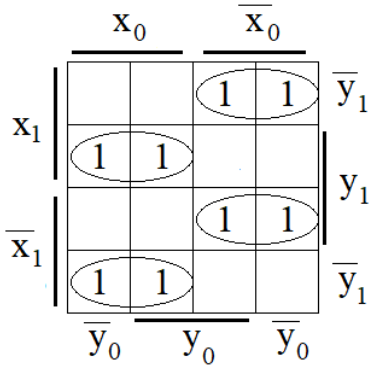


Рисунок 2.31 – Розмічена діаграми Вейча для функції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = \overline{x_0 y_1} x_1 + x_0 x_1 y_1 + \overline{x_0} y_1 \overline{x_1} + x_0 \overline{x_1 y_1} = x_0(x_1 y_1 + \overline{x_1 y_1}) + \overline{x_0}(\overline{y_1} x_1 + y_1 \overline{x_1})$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 21$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.32:

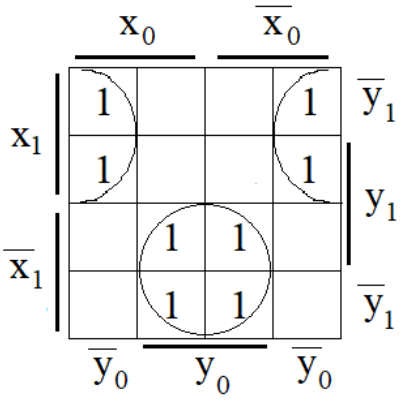


Рисунок 2.32 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = x_1 \overline{y_0} + y_0 \overline{x_1}$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 8$.

Загальна складність латинського квадрату $S_5 = S_{z0} + S_{z1} = 8 + 21 = 29$.

На рисунку 2.33 зображено структурну схему латинського квадрату.

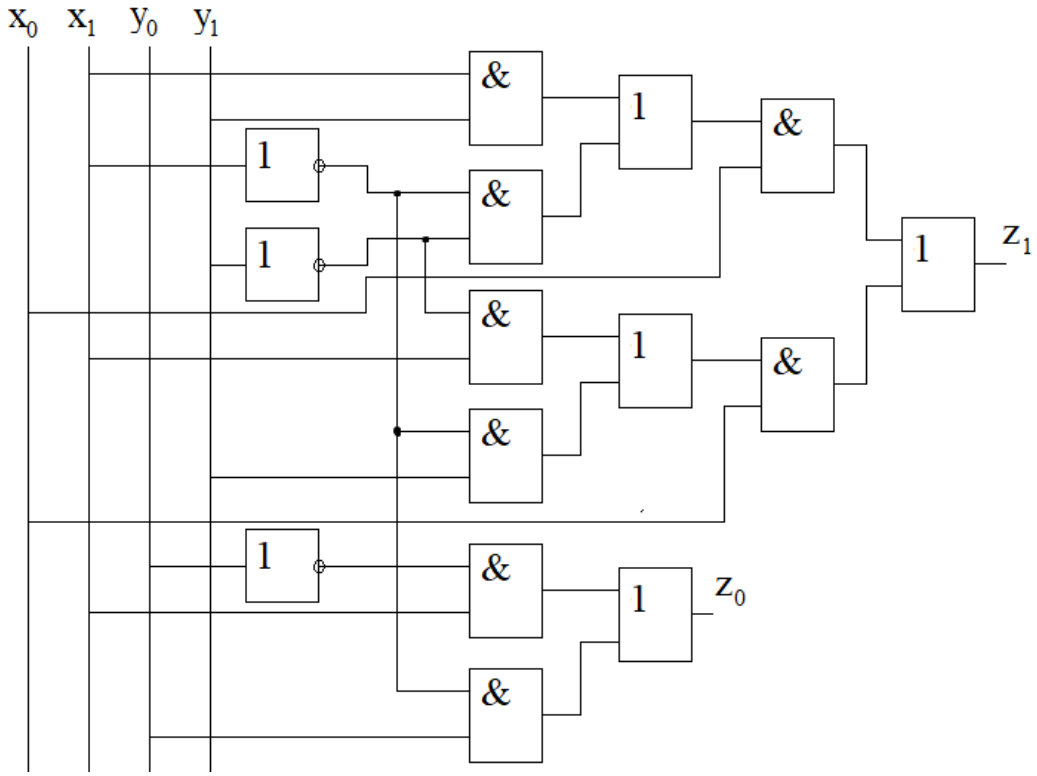


Рисунок 2.33 – Структурна схема латинського квадрату LS_5

Латинський квадрат LS_6 представлено на рис. 2.34:

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

Рисунок 2.34 – Латинський квадрат LS_6

Побудуємо таблицю істинності для даного латинського квадрату:

Таблиця 2.7 – Таблиця істинності латинського квадрату LS_6

N	x_1	x_0	y_1	y_0	z_1	z_0
0	0	0	0	0	1	1
1	0	0	0	1	0	0
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	1	0
6	0	1	1	0	0	0
7	0	1	1	1	1	1
8	1	0	0	0	0	0
9	1	0	0	1	1	1
10	1	0	1	0	0	1
11	1	0	1	1	1	0
12	1	1	0	0	1	0
13	1	1	0	1	0	1
14	1	1	1	0	1	1
15	1	1	1	1	0	0

Наступним кроком проведемо мінімізацію функції z_0 та z_1. Для цього розмітимо діаграму Вейча для кожної з функцій.

Розмічена діаграма Вейча для функції z_1 представлено на рис. 2.35:

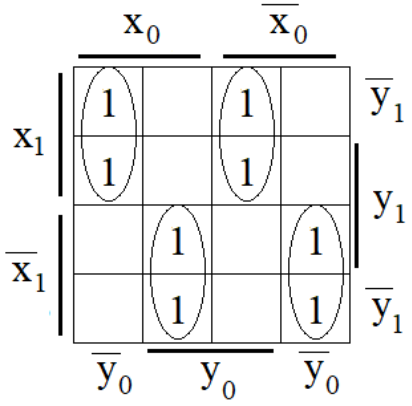


Рисунок 2.35 – Розмічена діаграма Вейча для фукнції z_1

З діаграми маємо наступну функцію та її спрощення:

$$z_1 = x_1x_0\overline{y_0} + x_1\overline{x_0}y_0 + \overline{x_1}y_0x_0 + \overline{x_1y_0x_0} = x_1(x_0\overline{y_0} + \overline{x_0}y_0) + \overline{x_1}(y_0x_0 + \overline{y_0x_0})$$

Отже, за методом Квайна, складність даної функції $S_{z1} = 21$.

Розмічена діаграма Вейча для функції z_0 представлено на рис. 2.36:

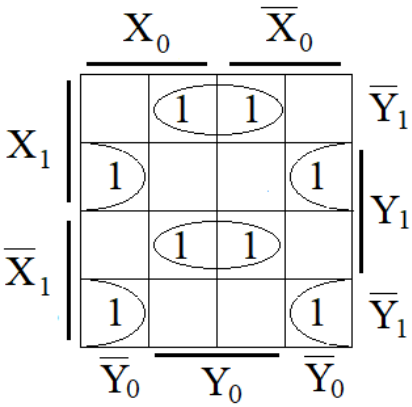


Рисунок 2.36 – Розмічена діаграма Вейча для фукнції z_0

З діаграми маємо наступну функцію та її спрощення:

$$z_0 = \overline{x_1y_0y_1} + \overline{x_1}y_0y_1 + x_1\overline{y_0}y_1 + x_1\overline{y_0}y_1 = \overline{x_1}(\overline{y_0y_1} + y_0y_1) + x_1(\overline{y_0}y_1 + y_0\overline{y_1})$$

Отже, за методом Квайна, складність даної функції $S_{z0} = 21$.

Загальна складність латинського квадрату $S_6 = S_{z0} + S_{z1} = 21 + 21 = 42$.

На рисунку 2.37 зображено структурну схему латинського квадрату.

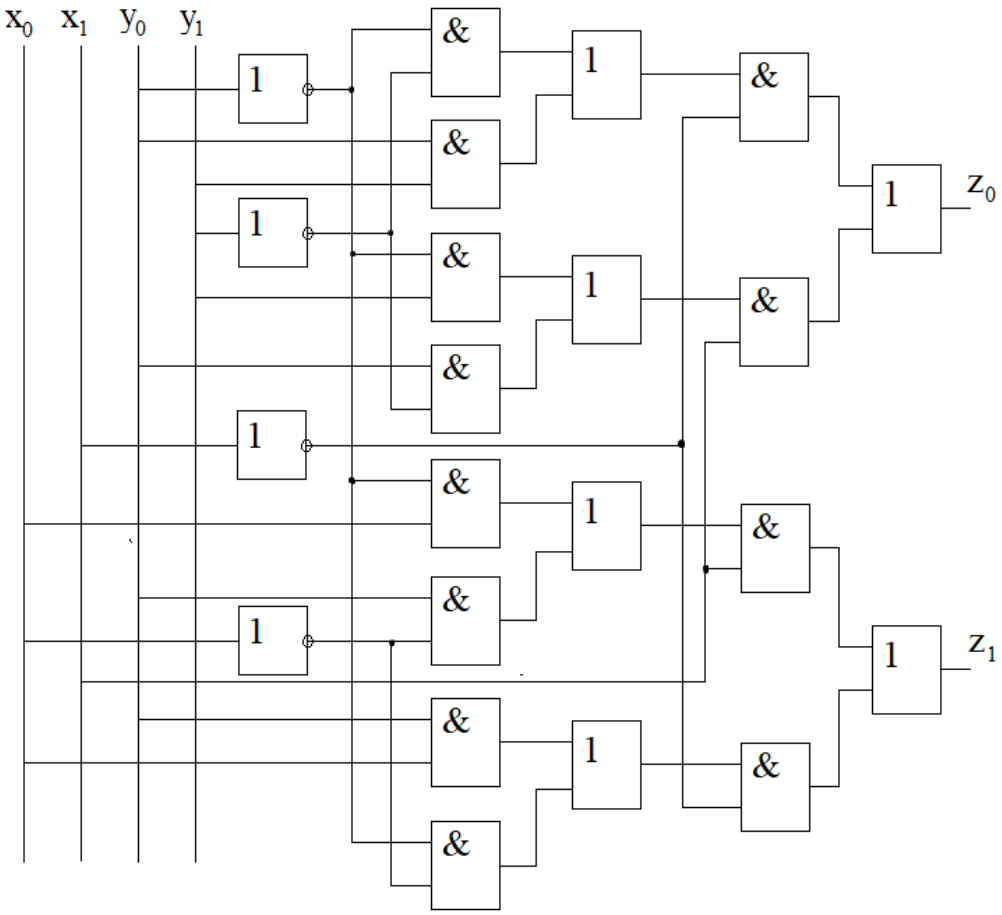


Рисунок 2.37 – Структурна схема латинського квадрату LS_6

Тепер, оцінивши складність всіх латинських квадратів, які використовуються у генеруванні псевдовипадкових чисел, підраховуємо загальну апаратну складність пристрою генерування псевдовипадкових чисел.

В сумі всі сім латинських квадратів мають складність реалізації:

$$S_з = S_0 + S_1 + S_2 + S_3 + S_4 + S_5 + S_6 = 232 \text{ (у. о.)}$$

Так як, за структурною схемою пристрою (див. рис. 2.7), маємо регістр для секретного ключа розрядністю 64, тобто ще 64 регістри з одним входом, то:

$$S_з = 242 + 64 = 306 \text{ (у. о.)} \qquad (2.6)$$

При реалізації усіх латинських квадратів за допомогою сучасних мікроелектронних технологій потрібно 68 елементів «І», 34 елемента «АБО» та 28 інверторів. Також, для реалізації генератору ПВЧ необхідний регістр зсуву, а це ще 64 тригера.

3 РОЗРОБКА ПОТОКОВОГО ШИФРУ

3.1 Розробка методу потокового шифру

Шифрування одного біту повідомлення у звичайних потокових шифрах виконується шляхом накладання 1 біту гами на 1 біт повідомлення використовуючи операцію додавання за модулем 2. У нашому випадку шифр буде обробляти по 2 біта використовуючи правило описане латинським квадратом. На рис. 3.1 представлено структурну схему пристрою для потокового шифрування:

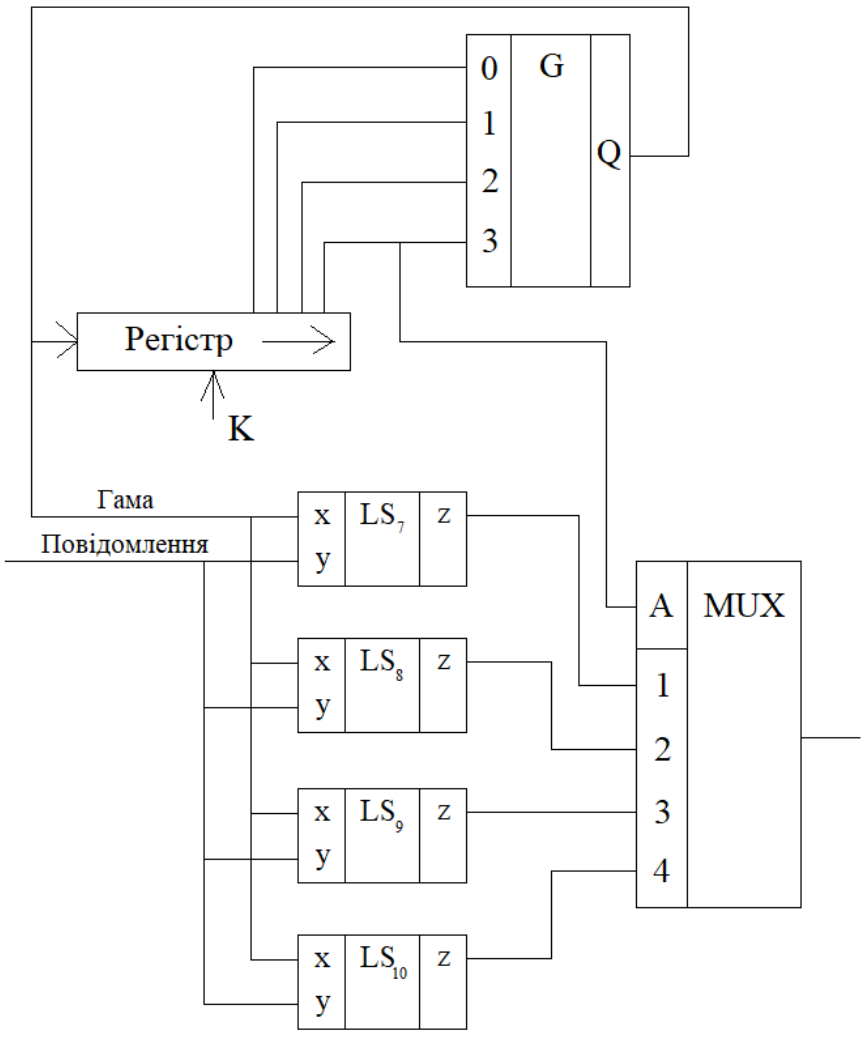


Рисунок 3.1 – Структурна схема пристрою для потокового шифрування

Пристрій складається з генератору гами, 4 латинських квадрата **$LS_{7\text{-}10}$** та мультиплексору **MUX**. В регістрі маємо ключ розрядністю 64 біт. 2-розрядне

вхідне повідомлення та 2-розрядний результат генератору гами надходять одночасно на всі латинські квадрати де виконується накладання гами на повідомлення. Мультиплексор визначає результат якого саме латинського квадрату буде передано на вихід. Для визначення на мультиплексор подається 2-розрядне значення, яке надходить з регістру на 3 вхід генератору.

Нехай, маємо вхідне повідомлення $m = 3 \,||\, 2 \,||\, 0 \,||\, 1 \,||\, 2 \,||\, 1 \,||\, 3 \,||\, 0$ та ключ $k = 1 \,||\, 2 \,||\, 3 \,||\, 0 \,||\, 0 \,||\, 2 \,||\, 3 \,||\, 1$. Відповідно до структурної схеми (див. рис. 3.1) необхідно побудувати чотири латинські квадрати для елементів $LS_{7\text{-}10}$ (рис. 3.2):

k \ m	0	1	2	3
0	2	3	1	0
1	1	0	2	3
2	0	1	3	2
3	3	2	0	1

LS_7

k \ m	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

LS_8

k \ m	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

LS_9

k \ m	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

LS_{10}

Рисунок 3.2 – Латинські квадрати для елементів $LS_{7\text{-}10}$

Далі необхідно записати ключ **k** в регістр зсуву для подання з нього по два розряди у генератор ПВЧ, тому представимо ключ у двійковому вигляді:

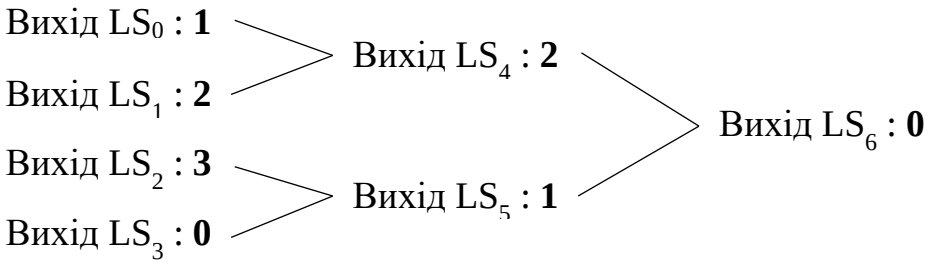
$$k = 01\ 10\ 11\ 00\ \mathbf{00\ 10\ 11\ 01}$$

Також, вхідне повідомлення **m** буде подаватись на латинський квадрат для накладання гами по два розряди, тому представимо вхідне повідомлення у двійковому вигляді:

$$m = 11\ 10\ 00\ 01\ 10\ 01\ 11\ 00$$

Отже, перейдемо до першого кроку. Згенеруємо псевдовипадкове число яке буде виступати гамою для накладання на першу складову вхідного повідомлення. Початкове значення регістру рівне ключу **k**, тому, відповідно до структурної схеми пристрою (див. рис. 3.1), на входи генератору **G** подаються дворозрядні значення ключа починаючи з молодших розрядів. На вхід **0** буде

подано значення **00**, на вхід **1** буде подано **10**, на вхід **2** буде подано **11** та на вхід **3** буде подано **01**. Використовуючи латинські квадрати $LS_{0\text{-}6}$, які описані для генератора ПВЧ (див. рис. 2.5), маємо наступний результат:



Отже, маємо першу складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_1 = 3}$. Маємо наступний результат:

Вихід LS_7 : **0**

Вихід LS_8 : **0**

Вихід LS_9 : **2**

Вихід LS_{10} : **3**

Так як, із чотирьох квадратів береться результат одного з них, тому, за допомогою дворозрядної складової ключа який подається на **3** вхід генератору **G** визначимо результат якого саме латинського квадрату буде виходом. Таким чином, на вхід **3** подається значення $10_2 = 2_{10}$, тобто виходом буде результат з латинського квадрату LS_8.

Отже маємо складову зашифрованого повідомлення $c_1 = 0$.

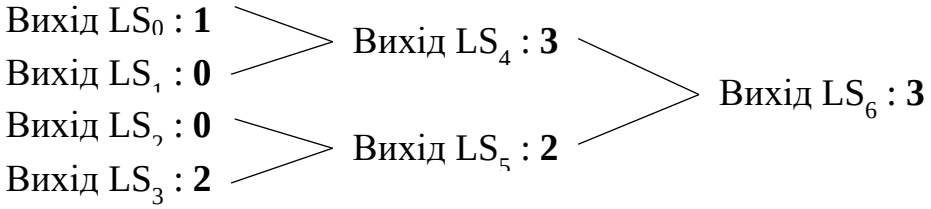
Паралельно з зашифруванням складової вхідного повідомлення **m** відбувається запис згенерованої складової гами у регістр тим самим виконуючи його зсув праворуч. Завдяки такому зсуву на вхід генератору **G** будуть подаватись нові значення. Тому, тепер маємо значення ключа:

$$k = 00\ 01\ 10\ 11\ \mathbf{00\ 00\ 10\ 11}$$

Аналогічно, виконаємо зашифрування решти складових вхідного повідомлення.

Перейдемо до зашифрування складової відкритого повідомлення $m_2 = 2$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_2}$. Маємо наступний результат:

Вихід LS 7 : **0** | Вихід LS 8 : **1** | Вихід LS 9 : **2** | Вихід LS10 : **3**

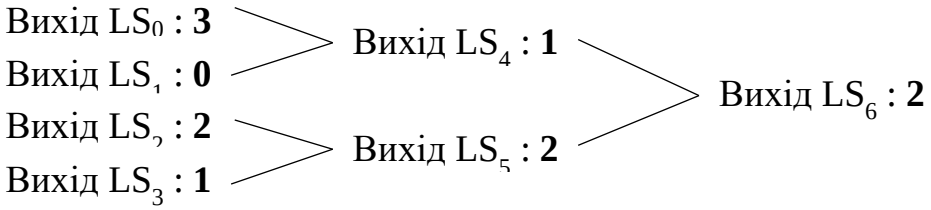
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_2 = 3$.

Після запису складової гами у регістр маємо:

$$k = 11\ 00\ 01\ 10\ \mathbf{11\ 00\ 00\ 10}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_3 = 0$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_3}$. Маємо наступний результат:

Вихід LS 7 : **0** | Вихід LS 8 : **2** | Вихід LS 9 : **2** | Вихід LS10 : **3**

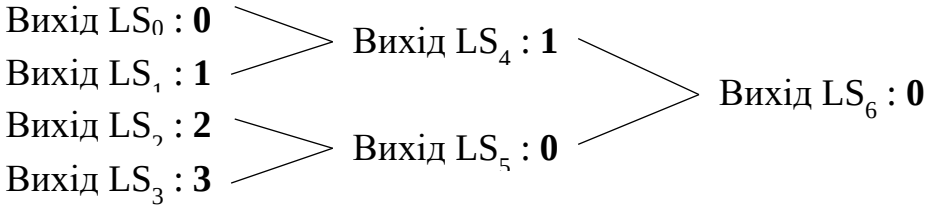
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_3 = 2$.

Після запису складової гами у регістр маємо:

$$k = 10\ 11\ 00\ 01\ \mathbf{10\ 11\ 00\ 00}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_4 = 1$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_4}$. Маємо наступний результат:

Вихід LS $_7$: **3** | Вихід LS $_8$: **2** | Вихід LS $_9$: **0** | Вихід LS_{10} : **1**

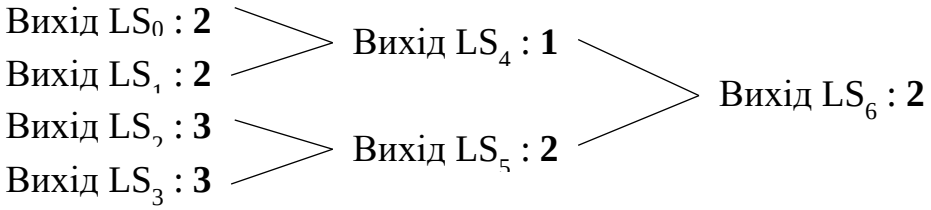
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_4 = 3$.

Після запису складової гами у регістр маємо:

$$k = 00\ 10\ 11\ 00\ \mathbf{01\ 10\ 11\ 00}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_5 = 2$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_5}$. Маємо наступний результат:

Вихід LS $_7$: **3** | Вихід LS $_8$: **0** | Вихід LS $_9$: **0** | Вихід LS_{10} : **1**

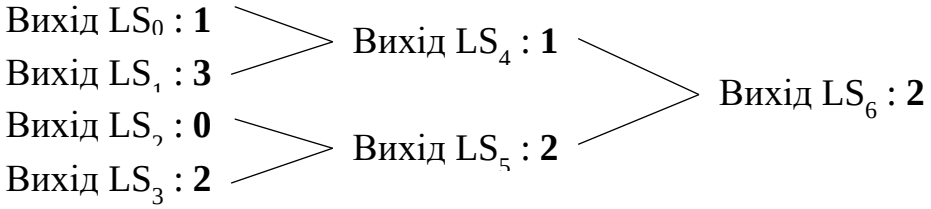
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_5 = 3$.

Після запису складової гами у регістр маємо:

$$k = 10\ 00\ 10\ 11\ \mathbf{00\ 01\ 10\ 11}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_6 = 1$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_6}$. Маємо наступний результат:

Вихід LS 7 : **1** | Вихід LS 8 : **1** | Вихід LS 9 : **1** | Вихід LS10 : **0**

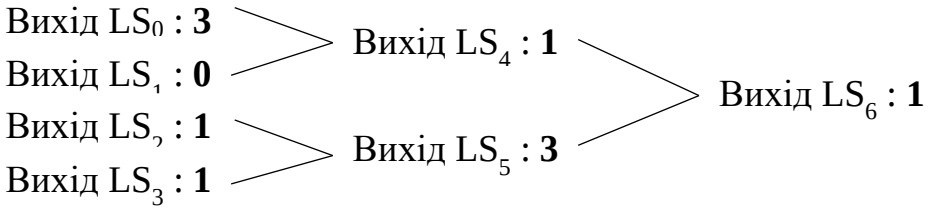
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_6 = 0$.

Після запису складової гами у регістр маємо:

$$k = 10\ 10\ 00\ 10\ \mathbf{11\ 00\ 01\ 10}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_7 = 3$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_7}$. Маємо наступний результат:

Вихід LS 7 : **3** | Вихід LS 8 : **1** | Вихід LS 9 : **0** | Вихід LS10 : **1**

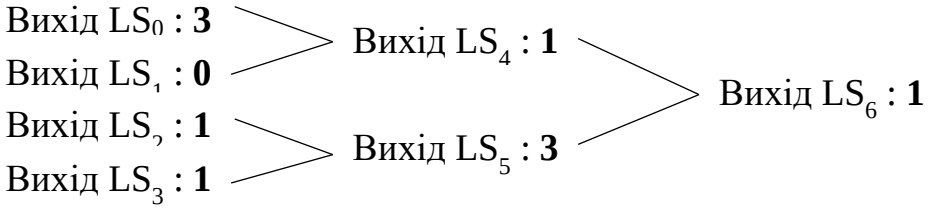
Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_7 = 0$.

Після запису складової гами у регістр маємо:

$$k = 10\ 10\ 00\ 10\ \mathbf{11\ 00\ 01\ 10}$$

Перейдемо до зашифрування складової відкритого повідомлення $m_8 = 0$.

Для генерування складової гами подаємо по два розряди з регістру на всі входи генератору. Маємо наступний результат:



Отже, маємо складову гами яку далі передаємо на чотири латинські квадрати $LS_{7\text{-}10}$ разом зі складовою відкритого повідомлення $\mathbf{m_8}$. Маємо наступний результат:

Вихід LS $_7$: **1** | Вихід LS $_8$: **0** | Вихід LS $_9$: **1** | Вихід LS_{10} : **0**

Відповідно до значення, яке подається на **3** вхід генератору **G**, маємо складову зашифрованого повідомлення $c_8 = 1$.

Після запису складової гами у регістр маємо:

$$k = 10\ 10\ 00\ 10\ \mathbf{11\ 00\ 01\ 10}$$

Отже, після зашифрування вхідного повідомлення **m**, порівняємо його з отриманим результатом **c**:

$$m = 3\ ||\ 2\ ||\ 0\ ||\ 1\ ||\ 2\ ||\ 1\ ||\ 3\ ||\ 0$$

$$\mathrm{c}\ = 0\ ||\ 3\ ||\ 2\ ||\ 3\ ||\ 3\ ||\ 0\ ||\ 0\ ||\ 1$$

Оцінимо складність апаратної реалізації пристрою для потокового шифрування.

Відповідно до структурної схеми пристрою (див. рис. 3.1), необхідно один генератор ПВЧ, реалізація чотирьох окремих латинських квадратів та один мультиплексор. Максимально можлива кількість логічних елементів для реалізації одного латинського квадрату становить 12 елементів «І», 6 елементів «АБО» та 4 інвертора. Тому для реалізації чотирьох квадратів нам потрібно 48 елементів «І», 24 елементів «АБО» та 16 інверторів. Відповідно до оцінки складності реалізації криптографічного генератору псевдовипадкових чисел маємо 84 елемента «І», 42 елемента «АБО», 28 інверторів та 64 тригера. Також,

для реалізації одного мультиплексора необхідно 8 елементів «І» та 2 елемента «АБО».

Складність пристрою, що реалізує потокове шифрування є вирішальним показником що говорить про малоресурсність. Для того щоб отримати ці дані та мати змогу зрівняти їх з іншими даними, вже відомих потокових шифрів, потрібно обрахувати складність кожного елементу, що реалізується у пристрої.

Складність кожного логічного елементу позначається спеціальними умовними одиницями що мають назву gate equivalent (GE).

Gate equivalent (еквівалентний логічний елемент) – одиниця виміру, яка дозволяє визначити виробничу складність технології незалежно від складності цифрових електронних схем. У табл. 3.1 [31] представлені короткі відомості за складністю реалізації різних логічних елементів.

Таблиця 4.1 – Складність логічних елементів

Умовне позначення елементу	Складність, GE
NOT	0,67
NAND	1
NOR	1
AND	1,33
OR	1,33
MUX	2,33
XOR (2)	2,67
XOR (3)	4,67
D flip flop	5,33

Як було описано вище, пристрій складається з 140 елементів «І», 68 елемента «АБО», 64 тригерів та 44 інверторів. Складність одного елемента «І» рівна складності одного елемента «АБО» та становить 1,33 GE, одного елемента «НІ» - 0,67 GE та одного тригера – 5,33 GE.

З отриманих даних, складність пристрою S обчислюється за формулою:

$$S = 64\ \text{Тг} + 140\ \text{"І"} + 68\ \text{"АБО"} + 44\ \text{"НІ"}\ \text{, де} \qquad (4.1)$$

Отже:

$$S = 64 * 5{,}33 + 140 * 1{,}33 + 68 * 1{,}33 + 44 * 0{,}67 = 647\ GE$$

У табл. 4.2 [32] наведено оцінки складності апаратної реалізації розробленого пристрою та відомих потокових шифрів.

Таблиця 4.2 – Оцінки складності апаратної реалізації потокових шифрів

Алгоритм	**Складність, GE**
Enocoro-80	2700
Grain v.1	11350
Trivium	749
WG-7	1097
Шифр, що пропонується	647

Порівнявши шифр, що пропонується, з рядом відомих потокових шифрів можна зробити висновок, що запропонований шифр обходить відомі потокові шифри у складності реалізації. Найближчим шифром по складності апаратної реалізації виявився Trivium зі складністю 749 GE, хоча є більш складні реалізації цього шифру.

Отже, мету дослідженнь досягнуто.

4 РОЗРОБКА БЛОКОВОГО ШИФРУ

4.1 Розробка методу блокового шифрування

Блокові шифри виконують шифрування повідомлення розбиваючи його на блоки однакової розрядності. Практично всі існуючі алгоритми використовують певний набір оборотних математичних перетворень. У нашому випадку, перетворення будуть виконуватись на основі правил описаних латинськими квадратами.

Нехай $\mathbf{M = m_1 \,||\, m_2 \,||\, \dots \,||\, m_n}$ – відкрите повідомлення, що складається з блоків $\mathbf{m_i}$ розрядністю 64 біт. Нехай **K –** секретний ключ розрядністю 64 біт. Кожний блок відкритого повідомлення, секретний ключ та блок зашифрованого повідомлення подаються як сукупність дворозрядних частин:

$$m_i = m_{i,1} \,||\, m_{i,2} \,||\, \dots \,||\, m_{i,32} \tag{4.1}$$

$$K = K_1 \,||\, K_2 \,||\, \dots \,||\, K_{32} \tag{4.2}$$

$$C_i = C_{i,1} \,||\, C_{i,2} \,||\, \dots \,||\, C_{i,32} \tag{4.3}$$

Кожна складова частина блоку зашифрованого повідомлення обчислюється як значення функції:

$$C_{i,j} = f_j\left(f_j\left(K_j,\, m_{i,j}\right),\, K_j\right),\;\; i = 1, 2, \dots, 32 \tag{4.4}$$

де f_i – функція, що описує **j**-й латинський квадрат. При шифруванні блоку $\mathbf{m_1}$ як **K** буде використовуватись секретний ключ, а при шифруванні блоку $\mathbf{m_i}$ як **K** буде використовуватись блок m_{i-1}. Частина $\mathbf{K_i}$ приймає 4 значення 0, 1, 2 і 3, тому використання вкладеної функції забезпечує реалізацію чотирьох різних латинських квадратів на основі одного. Оскільки $\mathbf{K_j}$ є секретним, то для зловмисника не буде відомо яка саме функція реалізується для дворозрядного повідомлення.

Розглянемо приклад реалізації блокового шифрування на основі латинських квадратів. Нехай $M = m_1||m_2||m_3||m_4$ – вхідне повідомлення та:

$$m_1 = 2 \,||\, 0 \,||\, 3 \,||\, 1$$
$$m_2 = 1 \,||\, 2 \,||\, 3 \,||\, 0$$
$$m_3 = 1 \,||\, 1 \,||\, 3 \,||\, 2$$
$$m_4 = 3 \,||\, 3 \,||\, 0 \,||\, 2$$

Нехай ключ $K = 0 \,||\, 3 \,||\, 1 \,||\, 2$.

Для реалізації функцій перетворення використовуються такі чотири латинські квадрати (рис. 4.1):

F_4

m \ k	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

F_3

m \ k	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

F_2

m \ k	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

F_1

m \ k	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

Рисунок 4.1 – Латинські квадрати для реалізації функцій перетворення

Для блоку повідомлення m_1 виконаємо перетворення відповідно за формулою 4.4:

$$\mathrm{C}_{1,1} = \mathrm{f}_1\left(\mathrm{f}_1\left(\mathrm{K}_1, \mathrm{m}_{1,1}\right), \mathrm{K}_1\right) = \mathrm{f}_1(\mathrm{f}_1(0,2),0) = \mathrm{f}_1(0,0) = \mathbf{1}$$

$$\mathrm{C}_{1,2} = \mathrm{f}_2\left(\mathrm{f}_2\left(\mathrm{K}_2, \mathrm{m}_{1,2}\right), \mathrm{K}_2\right) = \mathrm{f}_2(\mathrm{f}_2(3,0),3) = \mathrm{f}_2(1,3) = \mathbf{3}$$

$$\mathrm{C}_{1,3} = \mathrm{f}_3\left(\mathrm{f}_3\left(\mathrm{K}_3, \mathrm{m}_{1,3}\right), \mathrm{K}_3\right) = \mathrm{f}_3(\mathrm{f}_3(1,3),1) = \mathrm{f}_3(2,1) = \mathbf{0}$$

$$\mathrm{C}_{1,4} = \mathrm{f}_4\left(\mathrm{f}_4\left(\mathrm{K}_4, \mathrm{m}_{1,4}\right), \mathrm{K}_4\right) = \mathrm{f}_4(\mathrm{f}_4(2,1),2) = \mathrm{f}_4(2,2) = \mathbf{3}$$

Таким чином, для $m_1 = 2 \,||\, 0 \,||\, 3 \,||\, 1$ маємо на виході зашифрований блок повідомлення $\mathrm{c}_1 = 1 \,||\, 3 \,||\, 0 \,||\, 3$.

Якщо порівняти вхідний та вихідний (зашифрований) блок повідомлення у двійковому вигляді, то можна побачити, що всі символи були змінені окрім одного, що відповідає шифруванню:

$$m_1 = 10.00.11.0\mathbf{1}$$

$$\mathrm{c}_1 = 01.11.00.1\mathbf{1}$$

Перед переходом до кроку 2, ключ K приймає значення блоку повідомлення m_1. Тобто тепер ключ $K = m_1 = 2 \,||\, 0 \,||\, 3 \,||\, 1$.

По аналогії виконаємо шифрування решти трьох блоків вхідного повідомлення.

Для блоку повідомлення m_2 виконаємо перетворення:

$$C_{2,1} = f_1(f_1(K_1, m_{1,1}), K_1) = f_1(f_1(2,1),2) = f_1(1,2) = \mathbf{1}$$

$$C_{2,2} = f_2(f_2(K_2, m_{2,2}), K_2) = f_2(f_2(0,2),0) = f_2(2,0) = \mathbf{2}$$

$$C_{2,3} = f_3(f_3(K_3, m_{2,3}), K_3) = f_3(f_3(3,3),3) = f_3(0,3) = \mathbf{3}$$

$$C_{2,4} = f_4(f_4(K_4, m_{2,4}), K_4) = f_4(f_4(1,0),1) = f_4(2,1) = \mathbf{1}$$

Таким чином, для $m_2 = 1 \,||\, 2 \,||\, 3 \,||\, 0$ маємо на виході зашифрований блок повідомлення $c_2 = 1 \,||\, 2 \,||\, 3 \,||\, 1$.

Для блоку повідомлення m_3 виконаємо перетворення:

$$C_{2,1} = f_1(f_1(K_1, m_{1,1}), K_1) = f_1(f_1(2,1),2) = f_1(1,2) = \mathbf{3}$$

$$C_{2,2} = f_2(f_2(K_2, m_{2,2}), K_2) = f_2(f_2(0,2),0) = f_2(2,0) = \mathbf{2}$$

$$C_{2,3} = f_3(f_3(K_3, m_{2,3}), K_3) = f_3(f_3(3,3),3) = f_3(0,3) = \mathbf{3}$$

$$C_{2,4} = f_4(f_4(K_4, m_{2,4}), K_4) = f_4(f_4(1,0),1) = f_4(2,1) = \mathbf{2}$$

Таким чином, для $m_3 = 1 \,||\, 2 \,||\, 3 \,||\, 0$ маємо на виході зашифрований блок повідомлення $c_3 = 1 \,||\, 2 \,||\, 3 \,||\, 1$.

Для блоку повідомлення m_4 виконаємо перетворення:

$$C_{2,1} = f_1(f_1(K_1, m_{1,1}), K_1) = f_1(f_1(2,1),2) = f_1(1,2) = \mathbf{0}$$

$$C_{2,2} = f_2(f_2(K_2, m_{2,2}), K_2) = f_2(f_2(0,2),0) = f_2(2,0) = \mathbf{3}$$

$$C_{2,3} = f_3(f_3(K_3, m_{2,3}), K_3) = f_3(f_3(3,3),3) = f_3(0,3) = \mathbf{0}$$

$$C_{2,4} = f_4(f_4(K_4, m_{2,4}), K_4) = f_4(f_4(1,0),1) = f_4(2,1) = \mathbf{1}$$

Таким чином, для $m_4 = 1 \,||\, 2 \,||\, 3 \,||\, 0$ маємо на виході зашифрований блок повідомлення $c_4 = 1 \,||\, 2 \,||\, 3 \,||\, 1$.

Отже, після проходження всіх кроків, число яких рівне кількості вхідних блоків повідомлення, маємо зашифровані блоки повідомлення:

$$c_1 = 1 \,||\, 3 \,||\, 0 \,||\, 3$$

$$c_2 = 1 \,||\, 2 \,||\, 3 \,||\, 1$$

$$c_3 = 3 \,||\, 2 \,||\, 3 \,||\, 2$$

$$c_4 = 0 \,||\, 3 \,||\, 0 \,||\, 1$$

Для реалізації описаних процедур запропоновано таку структуру (рис. 4.2):

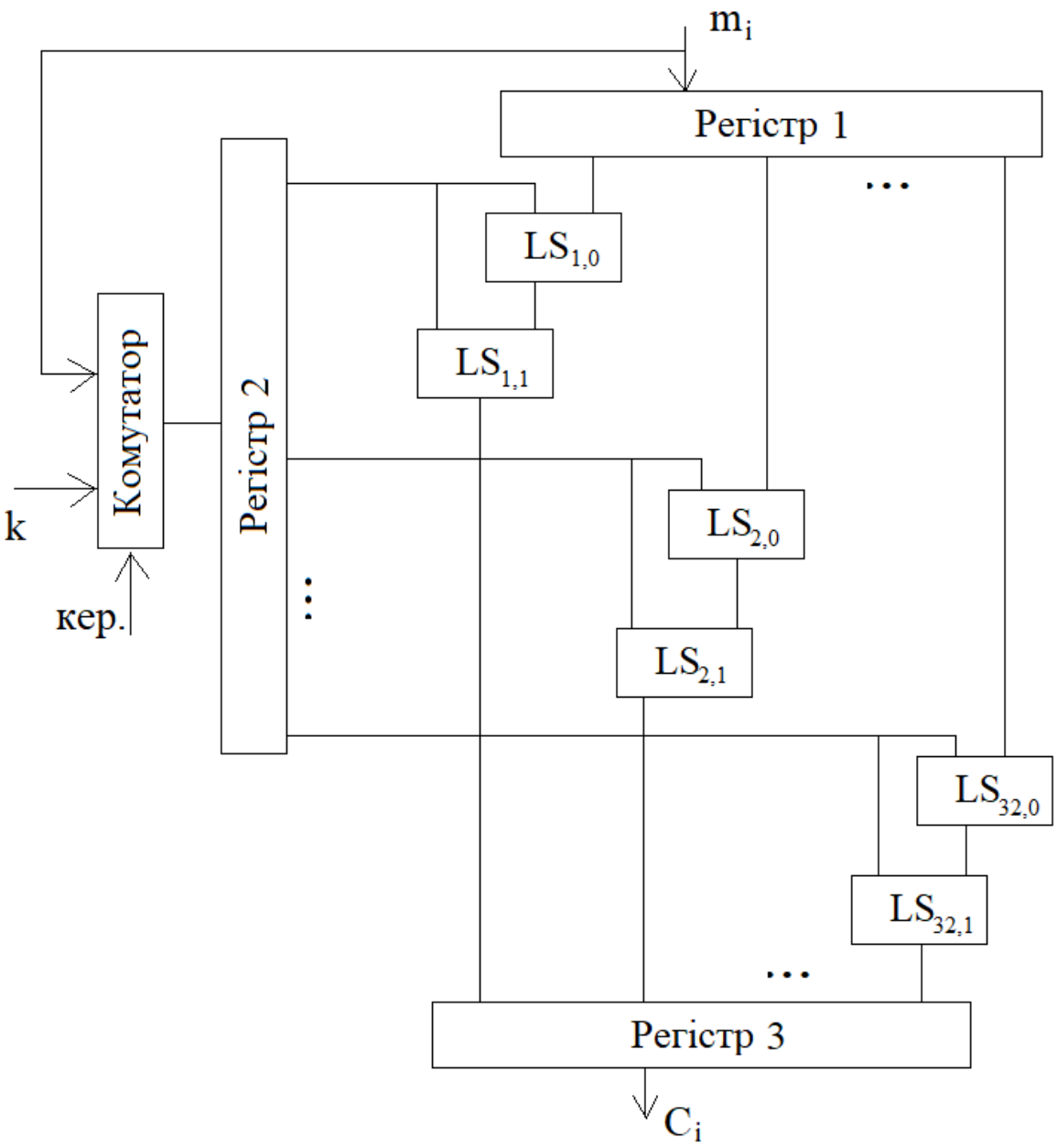


Рисунок 4.1 – Структурна схема пристрою для блокового шифрування

У регістр 1 буде подаватись вхідне повідомлення $\mathbf{m_i}$. На регістр 2 буде подаватись ключ **k** при першій ітерації та блок вхідного повідомлення над яким були виконані перетворення $\mathbf{m_{i-1}}$ для наступних ітерацій перетворень. Подача ключа або блоку вхідного повідомлення буде керуватись комутатором. Далі складові блоку вхідного повідомлення та складові ключа будуть подаватись на латинські квадрати. Так як, розрядність блоку повідомлення та ключа рівна 64, а на один латинський квадрат подається дворозрядне значення, то для реалізації перетворення усього блоку вхідного повідомлення необхідно 64 латинських квадрати (по два латинських квадрати на одну ітерацію). Зашифроване повідомлення буде записуватись у регістр 3.

Відповідно до структурної схеми пристрою необхідно три регістра (один для секретного ключа, другий для вхідного повідомлення та третій для вихідного зашифрованого повідомлення), реалізація 64 латинських квадратів та один комутатор.

Максимально можлива кількість логічних елементів для реалізації одного латинського квадрату становить 12 елементів «І», 6 елементів «АБО» та 4 інвертора. Тому для реалізації 64 латинських квадратів потрібно 768 елементів «І», 384 елемента «АБО» та 256 інверторів. Також маємо три регістра розрядністю 64, тому необхідно 192 тригера. Також маємо комутатор який виходить на 64-розрядний регістр, тому маємо ще 64 елемента «І».

Складність пристрою, що реалізує блокове шифрування є вирішальним показником що говорить про малоресурсність. Для того щоб отримати ці дані та мати змогу зрівняти їх з іншими даними, вже відомих блокових шифрів, потрібно обрахувати складність кожного елементу, що реалізується у пристрої.

Складність кожного логічного елементу позначається спеціальними умовними одиницями що мають назву gate equivalent (GE).

Як було описано вище, пристрій складається з 832 елементів «І», 384 елемента «АБО», 192 тригерів та 256 інверторів. Відповідно до табл. 3.1, складність одного елемента «І» рівна складності одного елемента «АБО» та становить 1,33 GE, одного елемента «НІ» - 0,67 GE та одного тригера – 5,33 GE.

З отриманих даних, складність пристрою S обчислюється за формулою:

$$S = 192\ \text{Тг} + 832\ \text{"І"} + 384\ \text{"АБО"} + 256\ \text{"НІ"}\ \text{, де} \quad (4.1)$$

Отже:

$$S = 192 * 5{,}33 + 832 * 1{,}33 + 384 * 1{,}33 + 256 * 0{,}67 = 2812\ GE$$

У табл. 4.2 [32] наведено оцінки складності апаратної реалізації розробленого пристрою та відомих блокових шифрів.

Таблиця 4.2 – Оцінки складності апаратної реалізації блокових шифрів

Алгоритм	**Складність, GE**
AES-128	3488
CAMELLIA	11350
CLEFIA-128	5979
IDEA	44708
KASUMI	2990
Шифр, що пропонується	2812

Отже, провівши порівняння ряду відомих блокових шифрів з запропонованим, можна зробити висновок, що запропонований шифр є відносно легким у апаратній реалізації, і обходить по складності такі відомі шифри як AES, CAAMELLIA, IDEA тощо. Найближчим шифром по складності апаратної реалізації виявився KASUMI зі складністю 2990 GE, хоча є більш складні реалізації цього шифру.

Отже, мету дослідженнь досягнуто.

Для побудови блокового шифру пропонується використовувати такі латинські квадрати:

LS_1

m \ k	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

LS_2

m \ k	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

LS_3

m \ k	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

LS_4

m \ k	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

LS_5

m \ k	0	1	2	3
0	1	0	2	3
1	2	3	1	0
2	3	2	0	1
3	0	1	3	2

LS_6

m \ k	0	1	2	3
0	2	0	3	1
1	1	3	0	2
2	0	2	1	3
3	3	1	2	0

LS_7

m \ k	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

LS_8

m \ k	0	1	2	3
0	1	3	2	0
1	2	0	1	3
2	0	2	3	1
3	3	1	0	2

LS_9

m \ k	0	1	2	3
0	1	3	0	2
1	0	2	1	3
2	2	0	3	1
3	3	1	2	0

LS_{10}

m \ k	0	1	2	3
0	1	3	2	0
1	2	0	1	3
2	0	2	3	1
3	3	1	0	2

LS_{11}

m \ k	0	1	2	3
0	1	0	2	3
1	3	2	0	1
2	0	1	3	2
3	2	3	1	0

LS_{12}

m \ k	0	1	2	3
0	3	0	2	1
1	0	1	3	2
2	2	3	1	0
3	1	2	0	3

LS_{13}

m \ k	0	1	2	3
0	2	1	3	0
1	0	3	1	2
2	1	2	0	3
3	3	0	2	1

LS_{14}

m \ k	0	1	2	3
0	2	0	1	3
1	1	3	2	0
2	3	1	0	2
3	0	2	3	1

LS_{15}

m \ k	0	1	2	3
0	1	0	2	3
1	2	3	1	0
2	3	2	0	1
3	0	1	3	2

LS_{16}

m \ k	0	1	2	3
0	2	1	0	3
1	3	0	1	2
2	1	2	3	0
3	0	3	2	1

LS_{17}

m \ k	0	1	2	3
0	2	3	1	0
1	1	0	2	3
2	0	1	3	2
3	3	2	0	1

LS_{18}

m \ k	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

LS_{19}

m \ k	0	1	2	3
0	3	1	0	2
1	0	2	3	1
2	2	0	1	3
3	1	3	2	0

LS_{20}

m \ k	0	1	2	3
0	3	0	2	1
1	1	2	0	3
2	0	3	1	2
3	2	1	3	0

LS_{21}

m \ k	0	1	2	3
0	3	0	2	1
1	0	1	3	2
2	2	3	1	0
3	1	2	0	3

LS_{22}

m \ k	0	1	2	3
0	3	1	2	0
1	2	0	3	1
2	0	2	1	3
3	1	3	0	2

LS_{23}

m \ k	0	1	2	3
0	3	2	0	1
1	1	0	2	3
2	2	3	1	0
3	0	1	3	2

LS_{24}

m \ k	0	1	2	3
0	2	0	1	3
1	1	3	2	0
2	3	1	0	2
3	0	2	3	1

LS_{25}

m \ k	0	1	2	3
0	3	2	1	0
1	0	3	2	1
2	1	0	3	2
3	2	1	0	3

LS_{26}

m \ k	0	1	2	3
0	1	3	0	2
1	0	2	1	3
2	2	0	3	1
3	3	1	2	0

LS_{27}

m \ k	0	1	2	3
0	3	0	2	1
1	2	3	1	0
2	0	1	3	2
3	1	2	0	3

LS_{28}

m \ k	0	1	2	3
0	2	0	3	1
1	0	3	1	2
2	3	1	2	0
3	1	2	0	3

LS_{29}

m \ k	0	1	2	3
0	0	3	1	2
1	3	1	2	0
2	1	2	0	3
3	2	0	3	1

LS_{30}

m \ k	0	1	2	3
0	3	2	0	1
1	2	0	1	3
2	0	1	3	2
3	1	3	2	0

LS_{31}

m \ k	0	1	2	3
0	1	2	3	0
1	2	3	0	1
2	3	0	1	2
3	0	1	2	3

LS_{32}

m \ k	0	1	2	3
0	2	1	3	0
1	0	3	1	2
2	1	2	0	3
3	3	0	2	1

5 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини магістерської кваліфікаційної роботи є обґрунтування економічної доцільності розроблених методів та засобів криптографічного перетворення на основі латинських квадратів. Для виконання поставленої мети необхідно:

- оцінити комерційний потенціал розробки;
- оцінити витрати на виконання та впровадження результатів наукової роботи;
- розрахувати ціну та чистий прибуток реалізації результатів розробки;
- розрахувати період окупності наукової роботи та результатів розробки.

5.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Для проведення технологічного аудиту було залучено трьох незалежних експертів: Баришев Ю. В, Куперштейн Л. М., Войтович О. П. Кожен з експертів повинен ознайомитися з запропонованою розробкою та заповнити таблицю, яка визначає рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можливу оцінку в балах. Після виконання цього, підраховується середньоарифметична сума балів та визначається який рівень комерційного потенціалу має нова розробка. Здійснюємо оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в додатку. Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В,	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
1	3	2	3
2	3	3	2
3	4	3	3
4	3	2	2
5	3	2	2
6	3	2	2
7	2	3	3
8	3	4	3

Продовження таблиці 5.1

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В,	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
9	3	1	2
10	4	3	3
11	4	3	3
12	3	4	2
Сума балів	$СБ_1$=38	$СБ_2$=31	$СБ_3$=30
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{i} СБ_i}{i} = \frac{38+31+30}{3} = 33$		

За даними таблиці 5.1 робимо висновок щодо рівня комерційного потенціалу розробки методів та засобів криптографічного перетворення на основі латинських квадратів. При цьому використано рекомендації, що наведено у таблиці 5.2.

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів	Рівень комерційного потенціалу
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-50	Високий

Отже, відповідно до результатів оцінювання експертами рівень комерційного потенціалу розробки методів та засобів криптографічного перетворення на основі латинських квадратів вище середнього.

Оцінювання рівня якості розробки методів та засобів криптографічного перетворення на основі латинських квадратів здійснюється з метою порівняльного аналізу та визначення найбільш ефективного, з технічної точки зору, варіанта інженерного рішення.

Рівень якості – кількісна характеристика міри придатності певного виду продукції для задоволення конкретного попиту на неї при порівнянні з відповідними базовими показниками за фіксованих умов споживання.

Абсолютний рівень якості розробки методів та засобів криптографічного перетворення на основі латинських квадратів знаходимо обчисленням обраних для вимірювання показників, не порівнюючи їх із відповідними показниками аналогічних засобів.

Для визначення рівня якості розробки обрано систему параметрів: ймовірність дешифрування даних, складність апаратної реалізації, швидкість шифрування.

Визначаємо величину параметрів якості в балах та встановлюємо граничні значення (кращі, гірші, середні). Дані для кожного параметра представлено у таблиці 5.3.

Таблиця 5.3 – Основні параметри методів та засобів криптографічного перетворення на основі латинських квадратів

Параметри	Абсолютне значення параметра			Коефіцієнт вагомості параметра
	Краще +5…+4	Середнє +3	Гірше +1..+2	
ймовірність дешифрування даних	5			0.6
складність апаратної реалізації		3		0.3
швидкість шифрування			3	0.1

Із врахуванням коефіцієнтів вагомості відповідних параметрів можна визначити абсолютний рівень якості інноваційного рішення за формулою

$$K_{я.а.} = \sum_{i=1}^{n} Рн_i * a_i \,, \qquad (5.1)$$

де $Рн_i$ – числове значення i-го параметра інноваційного рішення, n – кількість параметрів інноваційного рішення, що прийняті для оцінювання, a_i – коефіцієнт вагомості відповідного параметра (сума коефіцієнтів вагомості всіх параметрів повинна дорівнювати 1).

Отже, абсолютний рівень якості методів та засобів криптографічного перетворення на основі латинських квадратів складає 3,7 балів.

Одночасно визначаємо відносний рівень якості розробленого методів та засобів криптографічного перетворення на основі латинських квадратів, шляхом порівнюючи показники з абсолютними показниками якості найліпших аналогів, що представлено у таблиці 5.4.

Таблиця 5.4 – Порівняння основних параметрів методів та засобів криптографічного перетворення на основі латинських квадратів та товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		
ймовірність дешифрування даних	0,16	0,13	2	0,6
складність апаратної реалізації	2600	2812	1,7	0,3
швидкість шифрування	3,46 Гбіт/с	2,25 Гбіт/с	0,98	0,1

Відносний рівень якості методів та засобів криптографічного перетворення на основі латинських квадратів визначаємо за формулою 5.2:

$$K_{я.в.} = \sum_{i=1}^{n} q_i * a_i \qquad (5.2)$$

За розрахунками відносний рівень якості методів та засобів криптографічного перетворення на основі латинських квадратів 1,8. Це означає, що нова розробка якісніша на 49% відносно товару-аналога.

У найширшому розумінні конкурентоспроможність товару – це можливість його успішного продажу на певному ринку і в певний проміжок часу.

Водночас конкурентоспроможною можна вважати лише однорідну продукцію з технічними параметрами і техніко-економічними показниками, що ідентичні аналогічним показникам уже проданого товару. Для того, щоб високоякісний товар був одночасно і конкурентоспроможним, він має відповідати критеріям оцінювання споживачів конкретного ринку в конкретний період часу.

Дані для розрахунку загального показника конкурентоспроможності розробки необхідно занести до таблиці 5.5.

Таблиця 5.5 – Нормативні, технічні та економічні параметри методів та засобів криптографічного перетворення на основі латинських квадратів та товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		
ймовірність дешифрування даних	0,16	0,13	2	0,6
складність апаратної реалізації	2600	2812	1,7	0,3
швидкість шифрування	3,46 Гбіт/с	2,25 Гбіт/с	0,98	0,1
Ціна за продукт, грн	17500	12000	0,68	-

Загальний показник конкурентоспроможності розробки (K) з урахуванням вищезазначених груп показників визначаємо за формулою 5.3:

$$K = \frac{I_{\text{т.п}}}{I_{\text{е.п.}}} = \frac{1{,}8}{0{,}68} = 2{,}65 \qquad (5.3)$$

де $I_{Т.П.}$ – індекс технічних параметрів (відносний рівень якості інноваційного рішення); $I_{Е.П.}$ – індекс економічних параметрів розрахований нижче за формулою 5.4:

$$I_{\text{Е.П.}} = \frac{\text{РН}_{\text{EI}}}{\text{РБ}_{\text{EI}}} = \frac{12000}{17500} = 0{,}68 \qquad (5.4)$$

де $РН_{EI}$, $РБ_{EI}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

Згідно розрахунків загальний показник конкурентоспроможності 0,68, що свідчить про більшу конкурентну спроможність методів та засобів криптографічного перетворення на основі латинських квадратів у порівнянні з товаром-аналогом на 104%.

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько-технологічної роботи

5.2.1 Розрахунок витрат, що стосуються виконавців розробки методів та засобів криптографічних перетворень

Команда розробки методів та засобів криптографічного перетворення на основі латинських квадратів складається з керівника, двох інженерів та тестувальника.

Основна заробітна плата для розробників (дослідників) *Зо*, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{M}{T_p} t \qquad (5.5)$$

де M – місячний посадовий оклад розробника;

T_p – кількість робочих днів у місяці, $T_p = 22$ дні; t – число днів роботи.

Розрахунки заробітної плати для розробників наведені в таблиці 5.6

Таблиця 5.6 – Розрахунки основної заробітної плати розробників

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Керівник	25000	1136,36	10	11363,6
Інженер	20000	909,09	15	13636,35
Всього:				24999,95

Основна заробітна плата робітників *Зр*, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_Р = \sum_{i=1}^{n} t_i \cdot C_i \,, \tag{5.6}$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{M_M * K_i}{T_p * T_{ЗМ}}, \tag{5.7}$$

де M_M – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., K_i – тарифний коефіцієнт робітника відповідного розряду, $T_р$ = 22 дні; $T_{зм}$ – тривалість зміни, $T_{зм}$ =8 годин.

Таблиця 5.7 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	55,767	1,51	446,14
Тестування	8	4	50,596	1,37	404,77
Впровадження	2	2	40,25	1,04	80,96
Всього					931,87

Додаткова заробітна плата $З_Д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 * (З_О + З_Р) = 0{,}1 * (24999{,}95 + 931{,}87) = 2593{,}182 \text{ (грн.)} \tag{5.8}$$

Нарахування на заробітну плату $Н_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{ЗП} = (З_О + З_Р + З_Д) * \frac{\beta}{100} = (24999{,}95 + 931{,}87 + 2593{,}182) * 0{,}22 = 6275{,}5 \text{ (грн.)} \tag{5.9}$$

де $З_О$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата розробників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне страхування.

Розрахунок амортизаційних відрахувань виконується за такою формулою:

$$A = \frac{Ц}{t_B} \times \frac{T}{12} \quad (5.10)$$

де $Ц$ – балансова вартість обладнання, грн;

T – термін використання (T=22 дні= 0,73 місяців);

t_B – корисний час використання (t_B для комп'ютера становить 4 роки).

Під час виконання розробки використовувався ноутбук вартістю 30000 грн. Амортизаційні відрахування для ноутбуку представлені у таблиці 5.8.

Таблиця 5.8 - Амортизаційні відрахування

Найменування	Ціна, грн.	Корисний час використання, роки	Термін використання, міс.	Сума амортизації, грн.
Ноутбук	28000	4	0,73	425,83
Всього	425,83			

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B \times П \times Ф \times K_П \quad (5.11)$$

де B – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,74 кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$=22*8=176 год);

$K_П$ – коефіцієнт використання потужності ($K_П$ <1, $K_П$ = 0,8).

Відповідно до формули 5.11 витрати на силову електроенергію:

$B_E = 4{,}62 \times 0{,}74 \times 176 \times 0{,}8 = 481{,}36$ (грн.)

Інші витрати $B_{ін}$ можна прийняти як (100-300)% від суми основної заробітної плати розробників, які виконували роботу, тобто:

$$B_{\text{ін}} = 1 * (24999{,}95 + 931{,}87) = 25931{,}82 \text{ (грн.)} \quad (5.11)$$

Сума усіх попередніх витрат дає загальні витрати на виконання роботи. Усі витрати складають:

$$B = 24999{,}95 + 931{,}87 + 2593{,}182 + 481{,}36 + 6275{,}5 + 25931{,}82 + 425{,}83 = 61639{,}512 \text{ (грн.)}$$

Розрахунок загальної вартості наукової розробки $В_{заг}$ за формулою:

$$B_{заг} = \frac{B}{\alpha}, \quad (5.12)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях.

$\text{В}_{\text{заг}} = \frac{61639,512}{1} = 61639,512$ (грн.)

Прогнозування загальних витрат *ЗВ* на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{B_{заг}}{\beta} \quad (5.13)$$

Розрахунок прогнозованих загальних витрат:

$\text{ЗВ} = \frac{61639,512}{0,7} = 88056,446$ (грн.)

5.2.2 Розрахунок собівартості розробки методів та засобів криптографічних перетворень

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B \times П \times Ф \times K_П \quad (5.14)$$

де *В* – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

П – установлена потужність комп'ютеру (*П*=0,74 кВт);

Ф – фактична кількість годин роботи комп'ютеру (*Ф*=22*8=176 год);

$K_П$ – коефіцієнт використання потужності ($K_П$ <1, $K_П$ = 0,8).

Відповідно до формули 5.14 витрати на силову електроенергію:

$B_E = 4,62 \times 0,74 \times 176 \times 0,8 = 481,36$ (грн.)

Основна заробітна плата робітників *Зр*, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_Р = \sum_{i=1}^{n} t_i \cdot C_i \, , \quad (5.15)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{M_M * K_i}{T_p * T_{3M}}, \quad (5.16)$$

де M_M – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., K_i – тарифний коефіцієнт робітника відповідного розряду, T_p = 22 дні; $T_{зм}$ – тривалість зміни, $T_{зм}$ =8 годин.

Таблиця 5.9 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	55,767	1,51	446,14
Тестування	8	4	50,596	1,37	404,77
Впровадження	2	2	40,25	1,04	80,96
Всього					931,87

Додаткова заробітна плата $З_Д$ всіх робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 * (З_р) = 0{,}1 * 931{,}87 = 93{,}187 \text{ (грн.)} \quad (5.17)$$

Нарахування на заробітну плату $Н_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{зп} = (З_р + З_д) * \frac{\beta}{100} = (931{,}87 + 93{,}187) * 0{,}22 = 225{,}51 \text{ (грн.)} \quad (5.18)$$

де $З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне страхування.

Загальновиробничі витрати з рахунку на одиницю продукції можна розрахувати за нормативами відносно до основної заробітної плати основних робітників, які виготовляють продукцію :

$$ЗВВ = Н_В * З_О, \quad (5.19)$$

Норматив загальновиробничих витрат для програмних продуктів становить 230-270%.

ЗВВ = 2,7 ∗ 931,87 = 2516,05 (грн.)

Сума попередніх витрат утворює виробничу собівартість розробки.

$$S_B = 481,36 + 931,87 + 93,187 + 225,27 + 2516,05 = 4247,74 \text{ (грн.)} \quad (5.20)$$

5.3 Розрахунок мінімальної ціни та чистого прибутку від реалізації розробки методів та засобів криптографічних перетворень на основі латинських квадратів

Ціна – це грошовий вираз вартості товару (продукції, послуги). Вона завжди коливається навколо ціни виробництва (перетвореної форми вартості одиниці товару, що дорівнює сумі витрат виробництва й середнього прибутку) та відображає рівень суспільне необхідних витрат праці.

Виходячи з того, що розробки, як правило, приймаються та впроваджуються за завданням замовника, або коли результатом розробки є продукція, що підлягає державному регулюванню, то нижню межу ціни реалізації розробки можна розрахувати за формулою 5.21:

$$Ц = S_B \cdot (1 + \frac{P}{100}) \cdot (1 + \frac{\omega}{100}), \quad (5.21)$$

де S_B – виробнича собівартість інноваційного рішення, грн.;

P – норматив рентабельності узгоджений із замовником або встановлений державою, (Р=30…60%);

ω – ставка податку на додану вартість, % (з осені 2021 року $\omega = 20\%$).

$$Ц = 4247,74 * \left(1 + \frac{60}{100}\right) * \left(1 + \frac{20}{100}\right) = 8155,66 \text{ (грн.)} \quad (5.21)$$

Із врахуванням коефіцієнта якості ціна розробки становить 30176 грн.

Чистий прибуток від реалізації розробки можна розрахувати за формулою:

$$П = (Ц - \frac{(Ц - МР) \cdot f}{100} - S_B - \frac{q \cdot S_B}{100}) \cdot (1 - \frac{h}{100}) \cdot РП, \quad (5.22)$$

де *Ц* – ціна розробки, грн.; *МР* – вартість матеріальних та інших ресурсів, що були придбані виробником для виготовлення розробки (*МР*=(0,1…0,2) *Ц*), грн.; *f* – зустрічна ставка податку на додану вартість, %; S_B – виробнича собівартість розробки, грн.; *q* – норматив, який визначає величину адміністративних витрат, витрат на збут та інші операційні витрати, % (рекомендовано *q*=5…10%); *h* – ставка податку на прибуток, %, *РП* – прогнозований попит продажів.

$$П = \left(30176 - \frac{(30176-30176*0,2)*14}{100} - 4247,74 - \frac{5*4247,74}{100}\right) * \left(1 - \frac{18}{100}\right) * 2 = 36631 \text{ (грн.)}, \quad (5.23)$$

Прогнозований чистий прибуток від реалізації розробки складає 36631 грн.

5.4 Розрахунок терміну окупності коштів вкладених у наукову розробку методу та засобу автентифікації з нульовим знанням

Термін окупності вкладених у реалізацію наукового проекту інвестицій розраховано за формулою 5.24:

$$Т_{ок} = \frac{ЗВ}{П} = \frac{88056,45}{36631} = 2,4 \text{ (роки)} \quad (5.24)$$

Оскільки $T_{ОК} < 3$ років, то фінансування наукової розробки методу та засобу автентифікації користувачів із нульовим знанням є доцільним.

5.5 Висновки до розділу

Отже, у цьому розділі виконано обґрунтування економічної доцільності проведення наукового дослідження та розробки методів та засобів криптографічних перетворень на основі латинських квадратів.

Рівень комерційного потенціалу розробки методів та засобів криптографічних перетворень на основі латинських квадратів вище середнього.

На основі параметрів засобів криптографічних перетворень визначено абсолютний рівень якості методів та засобів криптографічних перетворень на основі латинських квадратів який складає 3,7 бали.

Відносний рівень якості розробки, що складає 1,8. Це означає, що нова розробка якісніша на 49% відносно товару-аналога. Загальний показник конкурентоспроможності становить 0,68, що свідчить про більшу конкурентну

спроможність засобу автентифікації з нульовим знанням у порівнянні з товаром-аналогом на 104%.

Загальні витрати, що стосуються виконавців розробки склали 88056,45 грн, а собівартість розробки – 4247,74 грн.

Розраховано мінімальну ціну та прогнозований чистий річний прибуток від реалізації розробки, які склали 30176 грн. та 36631 грн. відповідно. Термін окупності продукції вкладених інвестицій складає 2,4 роки, що свідчить про доцільність фінансування розробки.

ВИСНОВКИ

Серед галузей дослідження в системах, мережах та пристроях IT можна відмітити одну з найважливіших, а саме, дослідження та розробка нових методів захисту інформації для забезпечення безпеки інформації, що передається каналами зв’язку.

Проведений аналіз відомих підходів до побудови блокових та потокових шифрів показав, що одним із перспективних є підхід, що базується на використанні латинських квадратів. Зокрема, латинські квадрати були використані у таких відомих методах шифрування як Edon80 – потоковий шифр, представлений у 2005 році, який добрався до третього туру конкурсу ESTREAM; IDEA – блоковий шифр, який вперше був представлений у 1990 році; сімейство геш-функцій Edon-R, яке брало участь у конкурсі SHA-3 на новий американський стандарт. Також латинські квадрати знайшли своє застосування у побудові схем поділу секрету та протоколу з нульовим розголошенням.

Саме тому в даній роботі запропоновано нові методи блокового та потокового шифрування, для реалізації яких передбачається використання латинських квадратів 4-го порядку.

Представлено структурні схеми пристроїв криптографічного генератору псевдовипадкових чисел, блокового та потокового шифрування.

Проведено синтез апаратних засобів, що реалізують набір латинських квадратів 4-го порядку, який забезпечив можливість отримати оцінки складності апаратної реалізації запропонованих генератора та шифрів.

Порівняльний аналіз складності запропонованих апаратних засобів і відомих апаратних реалізацій блокових та потокових шифрів показав, що запропоновані рішення мають меншу складність, ніж відомі.

Отже, мету досліджень досягнуто.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. NIST 800-38G. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. [Чинний від 2016-01-22]. National Institute of Standards and Technology: Dworkin M. Gaithersburg. 2016. 172p.

2. Stream Ciphers vs. Block Ciphers: веб-сайт. URL: https://www.jscape.com/blog/stream-cipher-vs-block-cipher (дата звернення: 17.10.2021)

3. Biham E., Shamir A. Differential cryptanalysis of the Data Encryption Standard. Springer Verlag, 1993. 188 p.

4. Гатченко Н.А., Исаев А.С., Яковлев А.Д. Криптографическая защита информации. Учебное пособие. СПб: НИУ ИТМО, 2012. 142 с.

5. Глинчук Л. Я. Криптологія: навчально-методичний посібник. Луцьк: Вежа- Друк, 2014. 164 с.

6. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. Триумф, 2002. 816 с.

7. Полуяненко Н. А. Анализ современных тенденций развития генераторов потокового шифрования. Харьков: ХНУ им. В. Н. Каразина, 2017. 10 с.

8. Rueppel R.A. Analysis and Design of Stream Ciphers. Springer communications and control engineering series. Berlin, 1986. 244 p.

9. The eSTREAM Project. Technicsl Background: веб-сайт. URL: https://www.ecrypt.eu.org/stream/technical.html (дата звернення: 05.05.2021)

10. Babbage S. Some thoughts on Trivium. UK. Newbury, 2007. 5p.

11. Good T., Chelton W., Benaissa M. Review of stream cipher candidates from a low resource hardware perspective. UK: Department of Electrical & Electronic Engineering University of Sheffield, 2006. 24p.

12. Bjorstad T. E. Cryptanaltsis of Grain using Time, Memory, Data Tradeoffs. Norway: The Selmer Center Department of Informatics University of Bergen, 2008. 10 p.

13. Feldhofer M. Comparison of Low-Power Implementations of Trivium and Grain. Austria: Graz University of Technology Institute for Applied Information Processing and Communications, 2007. 11 p.

14. Boesgaard M., Vesterager M., Christensen T., Zenner E. The Stream Cipher Rabbit. Denmark. 2009. 30p.

15. Aumasson J.-P. On a bias of Rabbit. Switzerland, 2006. 10p.

16. О функциональном задании латинских квадратов: веб-сайт. URL: http://intsys.msu.ru/magazine/archive/v12(1-4)/nosov-317-332 (дата звернення: 23.09.2021).

17. М. Э. Тужилин. Латинские квадраты и их применение в криптографии. ПДМ, 2012. 87р.

18. McKay B.D., Wanless I. M. On the number of Latin Squares. 2005. 421p.

19. Построение латинских квадратов в булевской параметризации: веб-сайт. URL: http://www.intsys.msu.ru/magazine/archive/v9(1-4)/alashkevich-551-554 (дата звернення: 21.05.2020).

20. Bellaso G.B. Il vero modo di scrivere in cifra con facilit`a, prestezza, et securezza di Misser Giovan Battista Bellaso, gentil'huomo bresciano. Bressa, 1564. 560p.

21. Shannon C. Communication Theory of Secrecy Systems. Bell System Technical J. 1949. 715p.

22. Gligoroski D., Markovski S., Kocarev L., Gusev M. Edon80: веб-сайт. URL: http://www.ecrypt.eu. org/stream/edon80p3.html (дата звернення: 25.05.2020).

23. Lai X., Massey J. A. Proposal for a New Block EncryptionStandard Adv. Cryptology— EUROCRYPT'90. New York: Springer Verlag. 1991. 70p.

24. Gligoroski D., Ødeg˚ard R.S., Mihova M., et al. Cryptographic Hash Function Edon-R. Proc. IWSCN. Trondheim, 2009. 54p.

25. Cooper J., Donovan D., Seberry J. Secret Sharing Schemes Arising From Latin Squares Bulletin of the ICA. 1994. 142p.

26. D`enes J., D`enes T. Non-associative algebraic system in cryptology. Protection against meet in the middle attack. Quasigroups and Related Systems. 2001. 78p.

27. Gligoroski D. Candidate One-Way Functions and One-Way Permutations Based on Quasigroup String Transformations: веб-сайт. URL: http://eprint.iacr.org/2005/352 (дата звернення: 27.05.2020).

28. D`enes J., Keedwell A.D. Anew Authentication Scheme based in Latin Squares. Discrete Math. 1992. 162p.

29. Тихонова В.В., Р.И. Усманов, Н.Л. Додонова. Латинские квадраты и их применение в криптографии. Студенческая наука: современные реалии: материалы VII Междунар. студенч. науч.-практ. конф. (Чебоксары, 17 мая 2019 г.) / редкол.: О.Н. Широков [и др.] – Чебоксары: ЦНС «Интерактив плюс», 2019. – С. 72-78. – ISBN 978-5-6042714-7-6.

30. GE веб-сайт. URL:http://cryptowiki.net/index.php?title=GE (дата звернення: 29.10.2021).

ДОДАТКИ

Додаток А.Технічне завдання

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра захисту інформації

ЗАТВЕРДЖУЮ
завідувач кафедри ЗІ, д.т.н, проф.
_____ В. А.Лужецький
_____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему

«Методи та засоби криптографічних перетворень на основі латинських квадратів»

08-20.МКР.001.00.000 ТЗ

Розробив студент групи 1БС-20м
_____Абрамюк О. А.
Керівник магістерської кваліфікаційної роботи д. т. н., проф., зав. кафедри ЗІ
_____ Лужецький В. А.
«___»_____ 2021 р.

Вінниця 2021

1 Підстави для проведення робіт

Робота проводиться на підставі наказу ректора ВНТУ від 24 вересня 2021 року №277.

Дата початку роботи 01.09.2021 р.

Дата закінчення роботи 21.12.2021 р.

2 Мета та призначення МКР

Метою роботи є зменшення апаратних витрат на реалізацію пристроїв шифрування.

Предметом дослідження є засоби криптографічних перетворень.

Актуальність дослідження, проведеного у магістерській кваліфікаційній роботі полягає у тому, що існуючі методи захисту інформації не є досконалими, оскільки не дозволяють у повній мірі забезпечити безпеку інформації, що передається каналами зв'язку.

3 Джерела розробки

3.1 Dowkin M.: Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. NIST Special Publication 800-38G, Gaithersburg (2016)

3.2 Biham E., Shamir A. Differential cryptanalysis of the Data Encryption Standard. Springer Verlag, 1993. 188 p.

3.3 Глинчук Л. Я. Криптологія: навчально-методичний посібник. Луцьк: Вежа- Друк, 2014. 164 с.

3.4 Rueppel R.A. Analysis and Design of Stream Ciphers. Springer communications and control engineering series. Berlin, 1986. 244 p.

3.5 Good T., Chelton W., Benaissa M. Review of stream cipher candidates from a low resource hardware perspective. UK: Department of Electrical & Electronic Engineering University of Sheffield, 2006. 24 p.

3.6 Feldhofer M. Comparison of Low-Power Implementations of Trivium and Grain. Austria: Graz University of Technology Institute for Applied Information Processing and Communications, 2007. 11 p.

4 Вимоги до пристрою

- криптографічні перетворення – блоковий та потоковий шифри;
- розрядність ключа – 64;
- розрядність блоку – 64;

– період генератора псевдовипадкових послідовностей – не менше 2^{32};
– порядок латинських квадратів – 4;

5 Виконавці МКР

Студент групи 1БС-20м Абрамюк Олег Андрійович.

6 Вимоги до виконання МКР

Для реалізації поставленої мети необхідно розв’язати такі задачі:
- провести огляд методів блокового шифрування;
- провести огляд методів потокового шифрування;
- провести огляд генераторів псевдовипадкових послідовностей;
- проаналізувати використання латинських квадратів у криптографії;
- розробити метод та алгоритм блокового шифрування;
- розробити метод та алгоритм генерування псевдовипадкових послідовностей;
- розробити метод та алгоритм потокового шифрування;
- розробити структурні схеми пристроїв для шифрування.

7 Вимоги до супровідної документації

Графічна і текстова документація повинна відповідати діючим стандартам України.

8 Етапи МКР

№	Назва етапів роботи	Строк виконання	Результат
1	Аналіз завдання	до 04.09.2021	Вступ, технічне завдання
2	Аналіз інформаційних джерел.	до 05.10.2021	Звіт з аналізу інформаційних джерел
3	Розробка методів криптографічних перетворень: блокового та потокового шифрування, методу генерації псевдовипадкових послідовностей	до 24.10.2021	Методи криптографічних перетворень
4	Розробка структурних схем пристроїв криптографіних перетворень	до 29.11.2021	Структурні схеми пристроїв

9 Очікувані результати та порядок реалізації МКР

Передбачається розробка нових засобів криптографічних перетворень на основі латинських квадратів.

10 Матеріали які подаються після закінчення МКР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

11 Порядок приймання МКР та її етапів

Попередній захист та доопрацювання МКР грудень 2021 р.

Представлення МКР до захисту 15 грудня 2021р.

Захист МКР 21-23 грудня 2021 р.

12 Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп'ютерного набору у відповідності вимог ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання».

13 Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв'язку з тим, що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

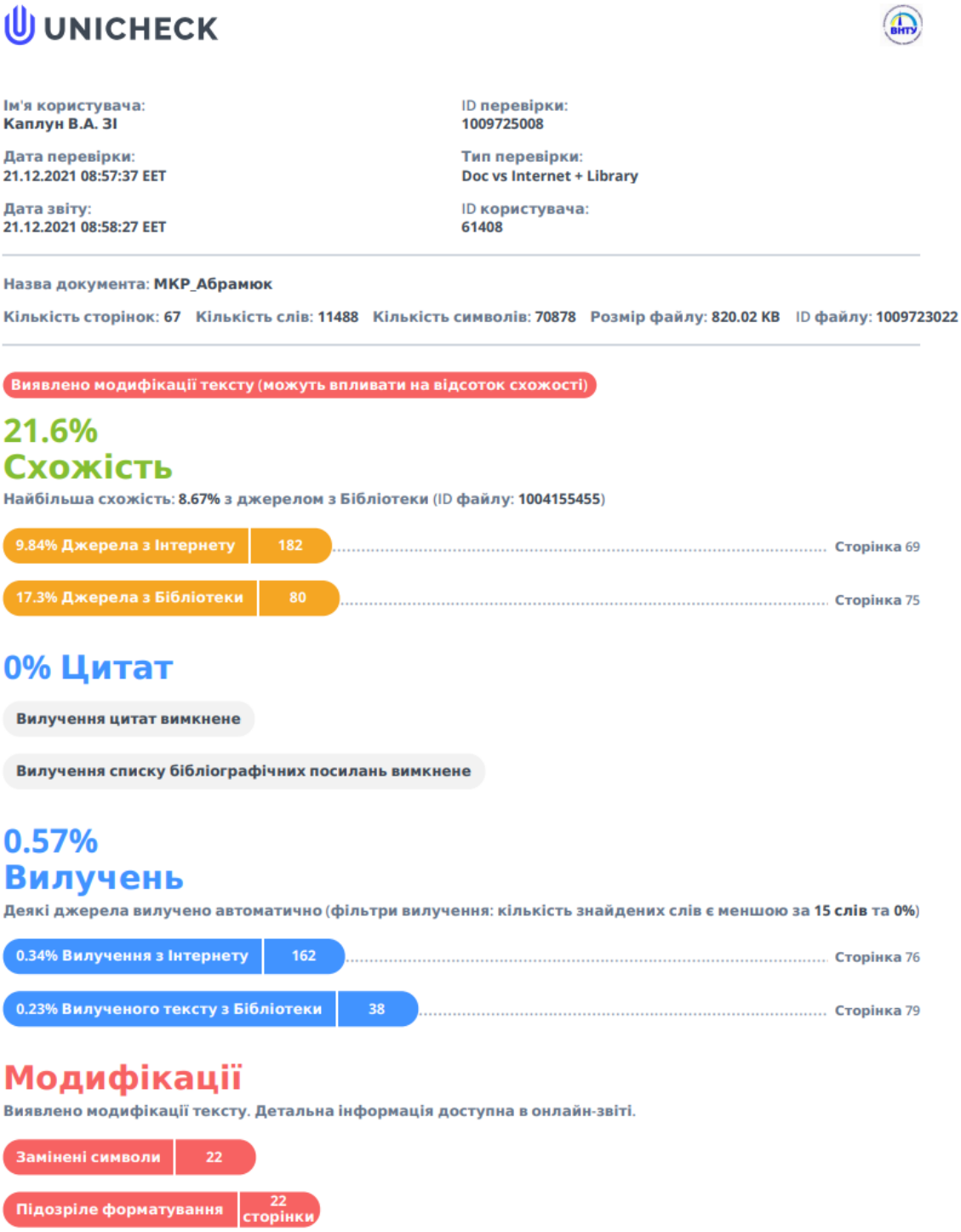
Розробив студент групи 1БС-20м _____Абрамюк О. А.

Додаток Б. Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	1	2	3	4	5
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	1	2	3	4	5
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Додаток В. Результат перевірки роботи на плагіат



UNICHECK

Ім'я користувача:
Каплун В.А. ЗІ

ID перевірки:
1009725008

Дата перевірки:
21.12.2021 08:57:37 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.12.2021 08:58:27 EET

ID користувача:
61408

Назва документа: МКР_Абрамюк

Кількість сторінок: 67 Кількість слів: 11488 Кількість символів: 70878 Розмір файлу: 820.02 KB ID файлу: 1009723022

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

21.6%
Схожість

Найбільша схожість: 8.67% з джерелом з Бібліотеки (ID файлу: 1004155455)

9.84% Джерела з Інтернету 182 ..... Сторінка 69

17.3% Джерела з Бібліотеки 80 ..... Сторінка 75

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.57%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 15 слів та 0%)

0.34% Вилучення з Інтернету 162 ..... Сторінка 76

0.23% Вилученого тексту з Бібліотеки 38 ..... Сторінка 79

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 22

Підозріле форматування 22 сторінки

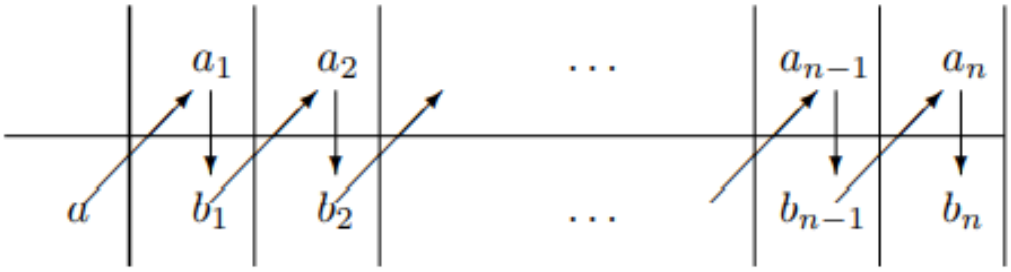
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

ПІДХІД ЩОДО ГЕНЕРУВАННЯ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ ЛАТИНСЬКИХ КВАДРАТІВ

$\bullet$	0	1	2	3
0	2	1	0	3
1	3	0	1	2
2	1	2	3	0
3	0	3	2	1

$\backslash$	0	1	2	3
0	2	1	0	3
1	1	2	3	0
2	3	0	1	2
3	0	3	2	1

$/$	0	1	2	3
0	3	1	0	2
1	2	0	1	3
2	0	2	3	1
3	1	3	2	0



					08-20.МКР.001.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Підхід щодо генерування псевдовипадкових чисел на основі латинськких квадратів.	Літ.	Арк. 1 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

ГЕНЕРАТОР ПОСЛІДОВНОСТІ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

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

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

x \ y	0	1	2	3
0	1	2	0	3
1	3	0	2	1
2	2	1	3	0
3	0	3	1	2

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

x \ y	0	1	2	3
0	3	0	1	2
1	1	2	3	0
2	2	1	0	3
3	0	3	2	1

x \ y	0	1	2	3
0	1	2	3	0
1	0	3	2	1
2	2	1	0	3
3	3	0	1	2

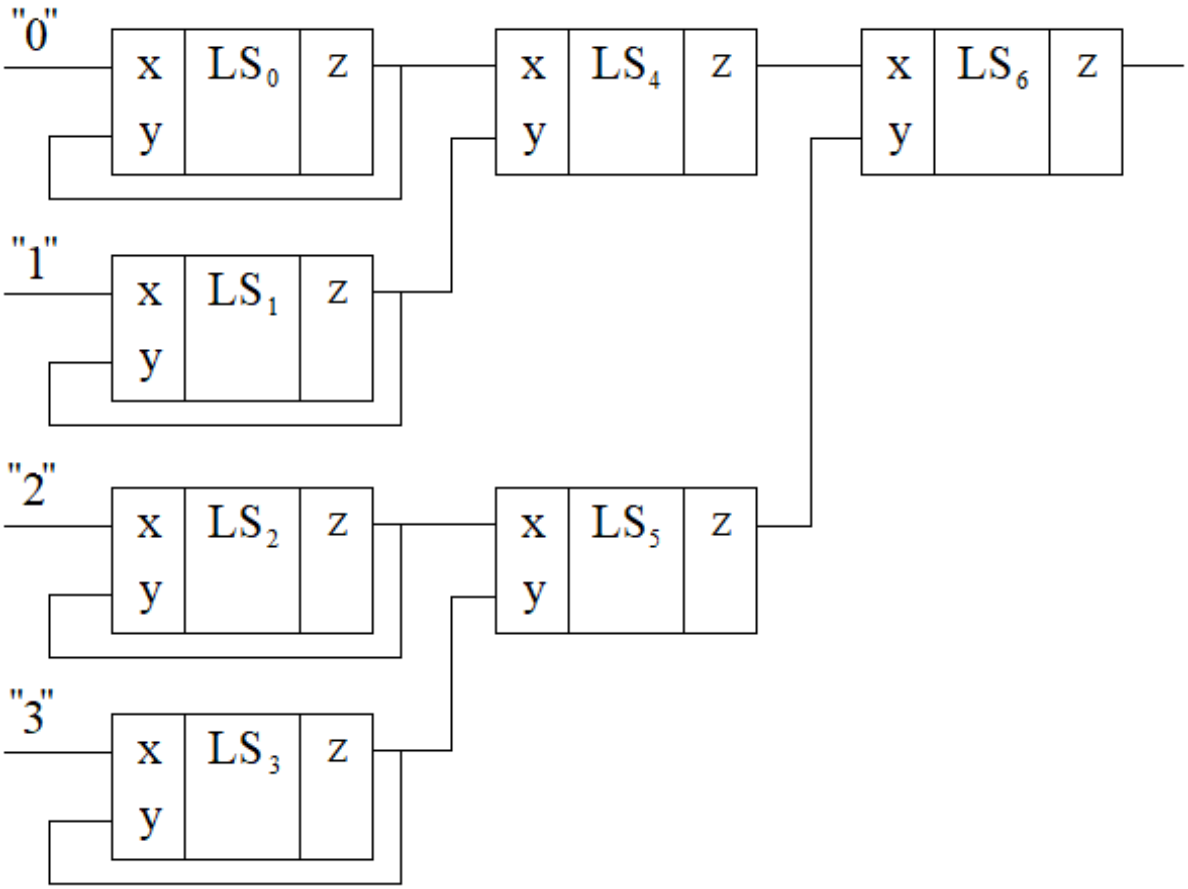
x \ y	0	1	2	3
0	2	1	3	0
1	3	0	2	1
2	0	3	1	2
3	1	2	0	3

x \ y	0	1	2	3
0	0	1	2	3
1	2	3	0	1
2	3	2	1	0
3	1	0	3	2

x \ y	0	1	2	3
0	2	1	0	3
1	0	3	2	1
2	3	0	1	2
3	1	2	3	0

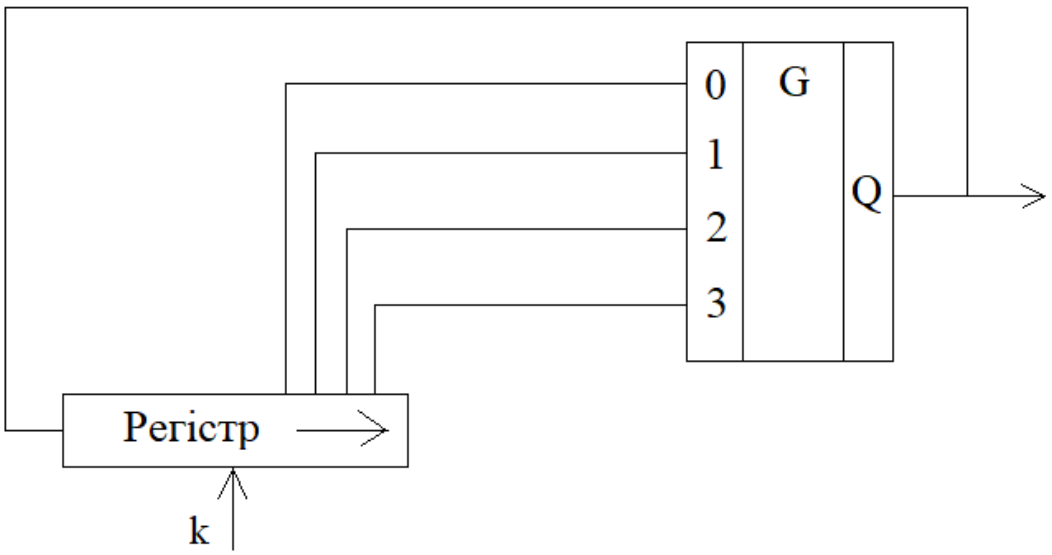
					08-20.МКР.001.00.000 І42		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Генератора послідовності псевдовипадкових чисел.	Літ.	Арк. 2 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

СТРУКТУРНА СХЕМА ГЕНЕРАТОРУ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ



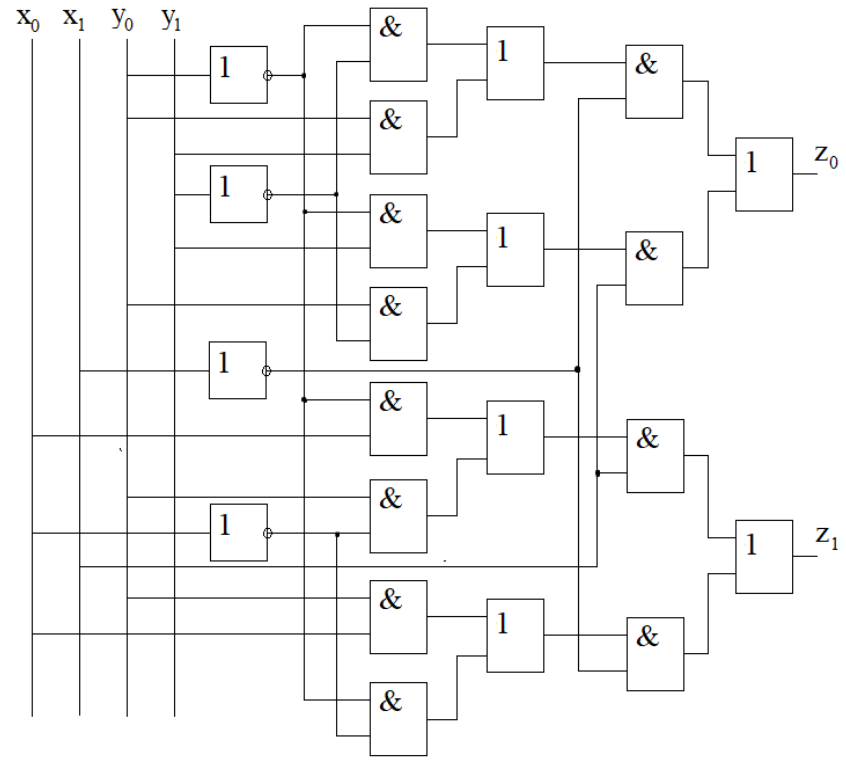
					08-20.МКР.001.00.000 ІЧЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів.Структурна схема генератору псевдовипадкових чисел.	Літ.	Арк. 3 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

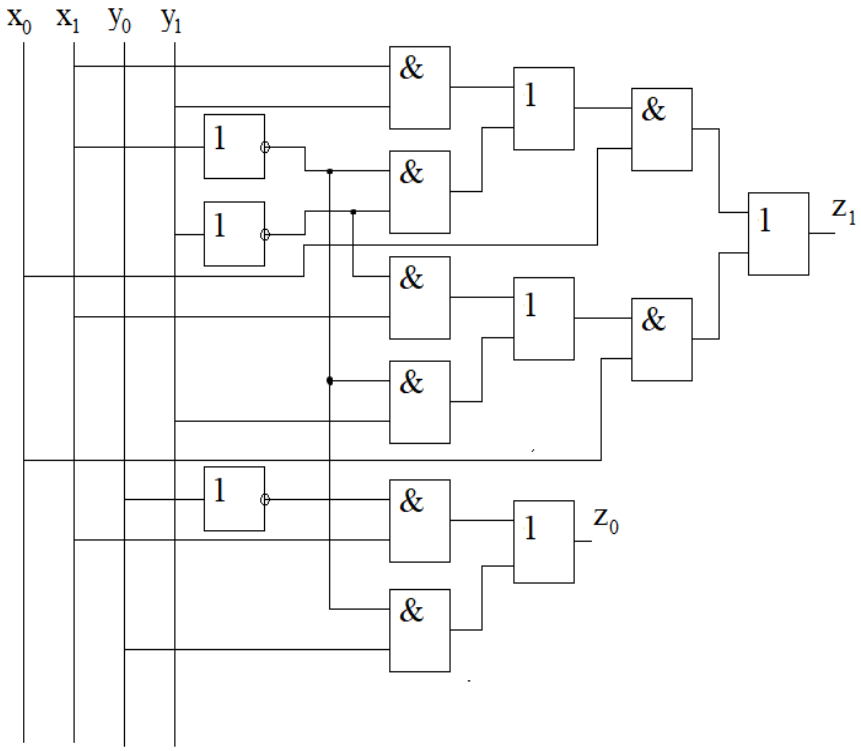
СТРУКТУРНА СХЕМА КРИПТОГРАФІЧНОГО ГЕНЕРАТОРУ ПСЕВДОПВИПАДКОВИХ ЧИСЕЛ



					08-20.МКР.001.00.000 І44		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів.Структурна схема криптографічного генератору псевдовипадкових чисел.	Літ.	Арк. 4 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

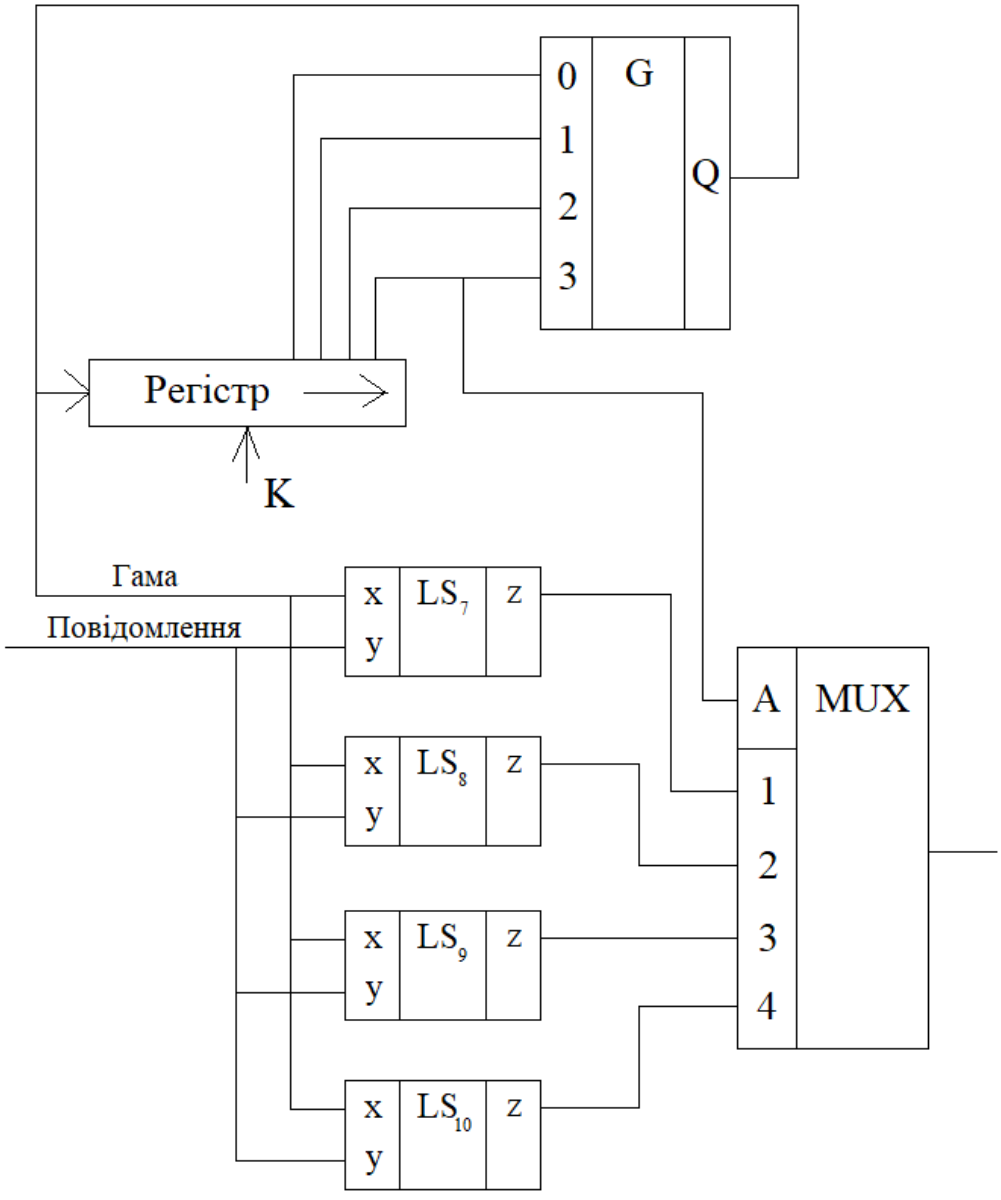
АПАРАТНА РЕАЛІЗАЦІЯ ЛАТИНСЬКИХ КВАДРАТІВ





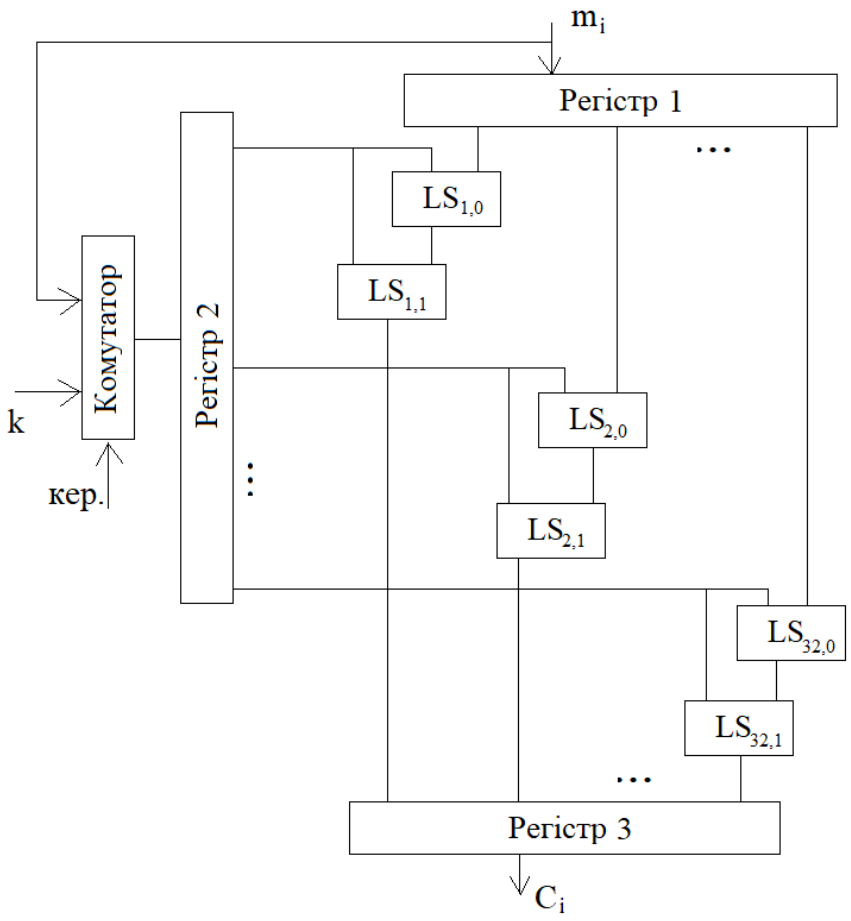
					08-20.МКР.001.00.000 І45		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Апаратна реалізація латинських квадратів.	Літ.	Арк. 5 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

СТРУКТУРНА СХЕМА ПРИСТРОЮ ДЛЯ ПОТОКОВОГО ШИФРУВАННЯ



					08-20.МКР.001.00.000 І46		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Структурна схема пристрою для потокового шифрування.	Літ.	Арк. 6 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

СТРУКТУРНА СХЕМА ПРИСТРОЮ ДЛЯ БЛОКОВОГО ШИФРУВАННЯ



					08-20.МКР.001.00.000 І47		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Структурна схема пристрою для блокового шифрування.	Літ.	Арк. 7 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

ОЦІНКА СКЛАДНОСТІ АПАРАТНОЇ РЕАЛІЗАЦІЇ ПОТОКОВИХ ШИФРІВ

Алгоритм	**Складність, GE**
Enocoro-80	2700
Grain v.1	11350
Trivium	749
WG-7	1097
Шифр, що пропонується	**647**

					08-20.МКР.001.00.000 І48		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Оцінка складності апаратної реалізації потокових шифрів.	Літ.	Арк. 8 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					

ОЦІНКА СКЛАДНОСТІ АПАРАТНОЇ РЕАЛІЗАЦІЇ ПОТОКОВИХ ШИФРІВ

Алгоритм	**Складність, GE**
AES-128	3488
CAMELLIA	11350
CLEFIA-128	5979
IDEA	44708
KASUMI	2990
Шифр, що пропонується	**2812**

					08-20.МКР.001.00.000 І49		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Абрамюк О.А.			Методи та засоби криптографічних перетворень на основі латинських квадратів. Оцінка складності апаратної реалізації блокових шифрів.	Літ.	Арк. 9 / Аркушів 9
Перевір.		Лужецький В.А.					
Реценз.		Азарова А.О.				ВНТУ, 1БС-20м	
Н. Контр.		Лужецький В.А.					
Затверд.		Лужецький В.А.					