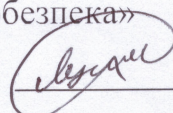


Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

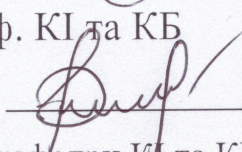
до випускної кваліфікаційної роботи бакалавра
на тему: «Проект бібліотеки та програмного
застосунку для демонстрації роботи національного
стандарту шифрування "Калина"»

Виконала студентка 4-го курсу групи КБ-19-2
спеціальності 125 «Кібербезпека»



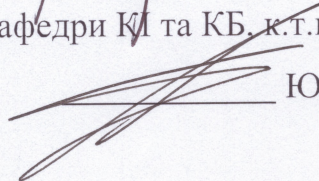
Марта Луцевич

Керівник, ст.в. каф. КІ та КБ



Єлизавета Байлюк

Рецензент, доцент кафедри КІ та КБ, к.т.н., доцент



Юрій Бродський

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

1 березня 2023 р.

ЗАВДАННЯ

на випускну кваліфікаційну роботу бакалавра

Студент: Луцевич Марта Олександрівна

Тема роботи: «Проект бібліотеки та програмного застосунку для демонстрації роботи національного стандарту шифрування "Калина"»

затверджена Наказом університету від 1 березня 2023 р. № 98/с

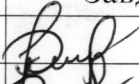
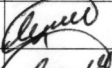
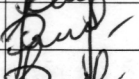
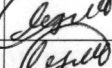
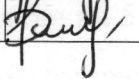
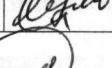
Термін здачі студентом закінченої роботи 29 червня 2023 р.

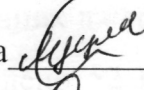
Вихідні дані роботи: (зазначається предмет і об'єкт дослідження):
Об'єктом дослідження є процес розробки програмного застосунку алгоритму шифрування «Калина». Предметом дослідження є методи та алгоритми шифрування «Калина».

Програмне забезпечення:

1. PyCharm2023
2. Python 3.0

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

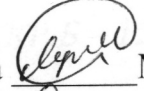
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Байлюк Є. М.	 24.04.2023	 24.04.2023
2	Байлюк Є. М.	 16.05.2023	 16.05.2023
3	Байлюк Є. М.	 08.06.2023	 08.06.2023

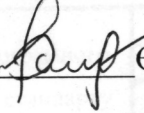
Студентка  Марта Луцевич

Керівник  Єлизавета Байлюк

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	01.03.2023-07.03.2023	
2	Аналіз літературних джерел	08.3.2023-25.03.2023	
3	Аналіз об'єкта дослідження	26.03.2023-04.04.2023	
4	Аналіз аналогічних розробок	05.04.2023-19.04.2023	
5	Вибір мови програмування	20.04.2023-26.04.2023	
6	Вибір середовища розробки	27.04.2023-6.05.2023	
7	Написання основних функцій	7.05.2023-23.05.2023	
8	Написання шифруючої функції	24.05.2023-31.05.2023	
9	Написання дешифруючої функції	01.06.2023-04.06.2023	
10	Написання пояснювальної записки	05.06.2023-20.06.2023	
11	Отримання відгуку та рецензії на кваліфікаційну роботу	21.06.2023-27.06.2023	
12	Захист	29.06.2023	

Студентка  Марта Луцевич

Керівник  Єлизавета Байлюк

РЕФЕРАТ

Пояснювальна записка на тему «Проект бібліотеки та програмного застосунку для демонстрації роботи національного стандарту шифрування "Калина"» складається зі вступу, трьох розділів, висновку, списку використаної літератури та додатків

Текстова частина викладена на 51 сторінках друкованого тексту.

Пояснювальна записка має список використаних джерел, який містить 16 найменувань і займає 3 сторінки. В роботі наведено 52 рисунки. Загальний обсяг роботи — 60 сторінок.

Завданням кваліфікаційної роботи було:

- аналіз структури алгоритму шифрування;
- аналіз безпеки та стійкості алгоритму;
- порівняння з іншими алгоритмами шифрування;
- опис основних функцій;
- обґрунтування мови програмування та середовища розробки;
- розробка програмного застосунку та бібліотеки;
- опис роботи бібліотеки.

В першому розділі було наведено теоретичні відомості та розглянуто основні кроки шифрування та дешифрування алгоритму «Калина»

У другому розділі було описано кодову реалізацію основних функцій.

У третьому розділі було розглянуто опис роботи з бібліотекою, наведено простий приклад її використання та проведене мінімальне тестування.

Висновок містить в собі результати розробки проєкту програмної реалізації алгоритму шифрування «Калина».

КЛЮЧОВІ СЛОВА: АЛГОРИТМ ШИФРУВАННЯ, ДСТУ 7624:2014, КАЛИНА

					KKIK.KP.B – 125 – 9 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Луцевич М. О.		27.06.23	Розробка бібліотеки та програмного				
Керівник		Байлюк Є. М.		27.06.23					
Реценз.		Бродський Ю. Б.		27.06.23	Застосунку національного стандарту				
Н. контр.					шифрування "Калина"				
Затверд.		Єфіменко А. А.		27.06.23	ФІКТ, Гр. КБ-19-2				

ANNOTATION

Explanatory note on the topic "Project of a library and software application to demonstrate the work of the national encryption standard "Viburnum"" consists of an introduction, three chapters, a conclusion, a list of references and appendices

The text part is laid out on 51 pages of printed text.

The explanatory note has a list of sources used, which contains 16 names and occupies 3 pages. There are 52 drawings in the work. The total volume of work is 61 pages.

The task of the qualification work was:

- analysis of the encryption algorithm structure,
- analysis of security and stability of the algorithm,
- comparison with other encryption algorithms,
- a description of the main functions,
- substantiation of the programming language and development environment,
- software application and library development,
- description of library work.

In the first section, theoretical information was provided and the main steps of encryption and decryption of the "Kalyna" algorithm were considered

The second section described the code implementation of the main functions.

In the third section, a description of working with the library was considered, a simple example of its use was given, and minimal testing was carried out

The conclusion contains the results of the development project of the software implementation of the "Kalyna" encryption algorithm

KEYWORDS: ENCRYPTION ALGORITHM, DSTU 7624:2014, KALINA

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АЛГОРИТМУ ШИФРУВАННЯ	9
1.1 Структура алгоритму шифрування ДСТУ 7624:2014 («Калина»)	9
1.2 Опис алгоритму розгортання ключів	11
1.3 Опис алгоритму шифрування	13
1.4 Опис алгоритму дешифрування.....	14
1.5 Аналіз безпеки та стійкості алгоритму	16
1.6 Порівняння з іншими алгоритмами шифрування	16
Висновки до першого розділу	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ	18
2.1 Опис основних функцій шифрування на прикладах	18
2.2 Опис основних функцій дешифрування на прикладах	27
2.3 Обґрунтування вибору мови програмування	31
2.4 Обґрунтування вибору середовища розробки.....	32
2.5 Програмна реалізація	33
Висновки до другого розділу	37
РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОПИС РОБОТИ.....	38
3.1 Опис роботи бібліотеки та додатка	38
3.2 Тестування	43
Висновки до третього розділу.....	46
ВИСНОВКИ.....	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	48
ДОДАТКИ.....	51

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

XOR — Виключна диз'юнкція (Exclusive or)

AES — Advanced Encryption Standard

DES — Data Encryption Standard

3DES — Triple DES

СЧ — Система числення

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. Впровадження надійних алгоритмів шифрування є ключовим елементом забезпечення інформаційної безпеки в сучасному світі, де передача даних через відкриті мережі є невід'ємною частиною багатьох галузей. Алгоритм «Калина» — перспективний алгоритм шифрування, розроблений у співпраці Держспецзв'язку, ПАТ «Інститут інформаційних технологій» та провідних українських вчених під керівництвом Горбенка І.Д.

Тому реалізація алгоритму шифрування Калина є актуальною темою кваліфікаційної роботи, оскільки цей алгоритм може бути застосований у різних сферах, пов'язаних з інформаційною безпекою, ставши важливим кроком у забезпеченні кібербезпеки в Україні. Така робота корисна студентам, які цікавляться криптографією, програмуванням і безпекою даних, а також викладачам, які хочуть залучити студентів до актуальних тем сучасного інформаційного світу.

Метою дослідження є розробка бібліотеки та програмного застосунку для демонстрації роботи для алгоритму шифрування «Калина»

Об'єктом дослідження є процес розробки програмного застосунку алгоритму шифрування «Калина».

Предметом дослідження є методи та алгоритми шифрування «Калина».

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АЛГОРИТМУ ШИФРУВАННЯ

1.1 Структура алгоритму шифрування ДСТУ 7624:2014 («Калина»)

Алгоритм шифрування Калина — це блоковий шифр з ключем довжиною 128, 256 або ж 512 біт. Його названо на честь квітки калини, що символізує національність розробників алгоритму — українців.

Алгоритм Калина складається з наступних етапів[3]:

1. Initialization - налагодження даних: таблиці заміни, константи.

2. Key Expansion - розширення ключа: генерація раундових ключів для кожного раунду шифрування. Для цього використовується алгоритм розширення ключа на основі симетричних шифрів.

3. Encryption Rounds - раунди шифрування, які включають нелінійні заміни та лінійні перестановки бітів з використанням раундових ключів. Кількість раундів залежить від довжини ключа та рівня безпеки, яку необхідно забезпечити.

4. Output Formatting - упорядкування кінцевих даних у певний формат.

Стандарт набрав чинності з 1 липня 2015 року наказом Мінекономрозвитку від 2 грудня 2014 року №1484. Текст стандарту у вільному доступі.[16]

Стандарт ДСТУ 7624:2014 розроблено у співпраці Держспецзв'язку та провідних українських науковців і враховує досвід та результати проведення міжнародних і відкритого національного конкурсу криптографічних алгоритмів. Він призначений для поступової заміни міждержавного стандарту ДСТУ ГОСТ 28147:2009[15], (введений в дію з 1 липня 2015 р.).

Згідно з чинним наказом Мінцифри від 30 вересня 2020 року №140/614 після 1 січня 2022 року разом з функцією гешування Купина є обов'язковим для використання при накладанні та перевірці електронного цифрового підпису за ДСТУ 4145-2002 замість криптографічного перетворення за ДСТУ ГОСТ 28147:2009.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Перетворення зашифрування описується формулою 1.1:

$$T_{l,k}^{(K)} = \eta_l^{(K_t)} \circ \psi_l \circ \tau_l \circ \pi'_l \circ \prod_{v=1}^{t-1} (\kappa^{(K_v)} \circ \psi_l \circ \tau_l \circ \pi'_l) \circ \eta_l^{(K_t)} \quad 1.1$$

де K — ключ шифрування k -бітної довжини,

$\eta_l^{(K_t)}$ — функція додавання до матриці стану раундового ключа за модулем 2^{64} ,

π'_l — заміна байтів у матриці стану,

τ_l — циклічний зсув вправо,

ψ_l — лінійне перетворення елементів стану над скінченним полем $x^8 + x^4 + x^3 + x^2 + 1$,

$\kappa^{(K_v)}$ — додавання до матриці стану раундового ключа за модулем 2 (виключна диз'юнкція).

Перетворення розшифрування описується формулою 1.2:

$$U_{l,k}^{(K)} = {}_{-1}\eta_l^{(K_t)} \circ \prod_{v=1}^{t-1} ({}_{-1}\pi'_l \circ {}_{-1}\tau_l \circ {}_{-1}\psi_l \circ {}_{-1}\kappa^{(K_v)}) \circ {}_{-1}\pi'_l \circ {}_{-1}\tau_l \circ {}_{-1}\psi_l \circ {}_{-1}\eta_l^{(K_t)} \quad 1.2$$

де K — ключ шифрування k -бітної довжини,

${}_{-1}\eta_l^{(K_t)}$ — функція віднімання від матриці стану раундового ключа за модулем 2^{64} ,

${}_{-1}\pi'_l$ — зворотня заміна байтів,

${}_{-1}\tau_l$ — зворотня перестановка елементів внутрішнього стану шифру, циклічний зсув вліво,

${}_{-1}\psi_l$ — зворотнє лінійне перетворення елементів стану над кінцевим полем,

$\kappa^{(K_v)}$ — додавання до матриці стану раундового ключа за модулем 2 (виключна диз'юнкція).

Щодо криптостійкості, було виявлено деякі атаки[11][13] на скорочені варіанти шифру, але вони не є практичними

Узагальнюючи, структура алгоритму шифрування Калина складається з комбінації лінійних та нелінійних операцій, що забезпечує захист даних.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

1.2 Опис алгоритму розгортання ключів

Опис алгоритму розгортання ключів в ДСТУ 7624:2014 («Калина») включає наступні кроки:

1. На основі ключа шифрування (K) формується допоміжний ключ (K_t), з допомогою якого будуть формуватися майбутні раундові ключі (рис. 1.1). K_t формується на основі 3-х раундів зашифрування і дорівнює розміру блоку — $64 \times N_b$:

- початковими даними є значення $N_b + N_k + 1$. Сума трьох доданків дорівнює найпершому числу в 16-ковій СЧ. Інші байти заповнюються значеннями 0x00;
- після кожного раунду використовуються ключ шифрування K , який і буде використовуватися замість раундових ключів;

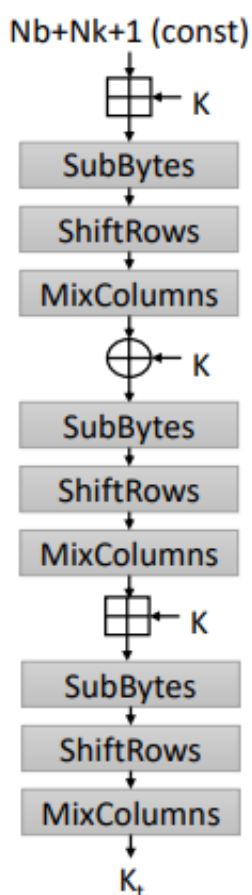


Рисунок 1.1. Формування допоміжного ключа

2. Після отримання K_t на основі ключа K , формуються раундові ключі[12] (рис. 1.2), які будуть брати участь при шифруванні та дешифруванні:

- базуючись на допоміжному ключі K_t , і ключі K формується парні раундові ключі K_{2i} з використанням двох раундів;
- після кожного раунду використовуються результат додавання по модулю 2^{64} між допоміжним ключем і tmv_i . Значення tmv_i формуються на основі індексу раундового ключа. $tmv_{i+2} = tmv_i \ll 1, tmv_0 = 0x0100\ 0100 \dots 0100$;
- після сформування парних раундових ключів, формуються не парні раундові ключі. Відбувається циклічний зсув попереднього ключа з парним індексом вліво на $2 \times N_b + 3$ байт (рис. 1.3);

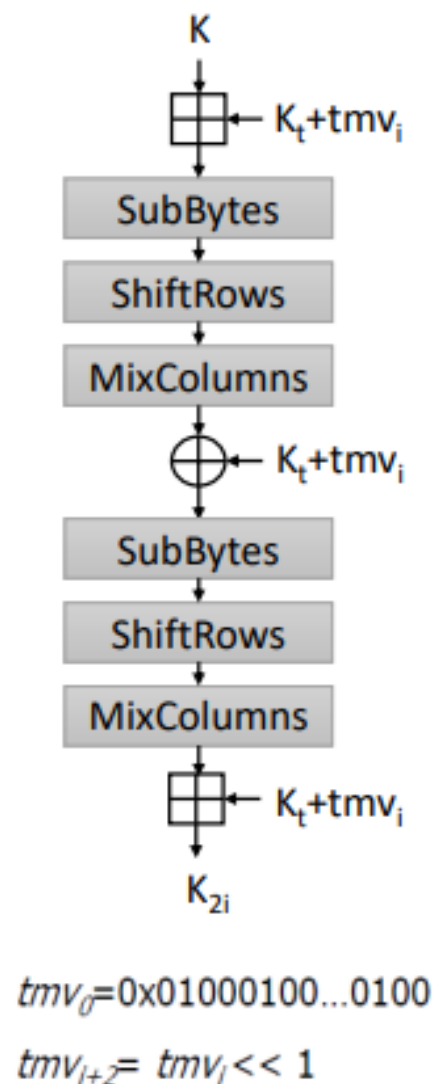


Рисунок 1.2. Формування парних раундових ключів

<i>Розмір блоку, біт (байт)</i>	<i>Зсув вліво (байт)</i>
128(16)	7
256(32)	11
512(64)	19

Рисунок 1.3. Правило формування непарних ключів

1.3 Опис алгоритму шифрування

Алгоритм шифрування Калина[2] використовує ключі розміром 128, 256 або 512 біт і повідомлення, розбиті на блоки фіксованого розміру.

Після розширення ключа використовуються чотири основні функції: SubBytes (підстановка байтів), ShiftRows (зсув рядків), MixColumn (перемішування стовпців), XOR[5] та додавання по модулю 2^{64} . Ці функції виконуються у циклі, кількість ітерацій якого залежить від довжини ключа.

На кожній ітерації виконуються такі кроки:

- підстановка байтів: вхідний блок даних проходить через чотири S-блоки ($\pi_0, \pi_1, \pi_2, \pi_3$), кожен з яких замінює частину блока на нове значення за таблицею заміни;
- зсув рядків: вхідний блок даних проходить через лінійне перетворення, яке змінює його структуру[6];
- перемішування стовпців: перетворення стовпців матриці стану шляхом виконання операцій множення і додавання в скінченному полі $GF(2^8)$ за модулем незвідного многочлена $x^8 + x^4 + x^3 + x^2 + 1$;
- XOR: побітове додавання раундового ключа до матриці стану за модулем 2;

Під час останнього раунду, в кінці використовується додавання по модулю 2^{64} , замість XOR.

Загальну структуру шифрування зображено на рисунку 1.4 схематично.

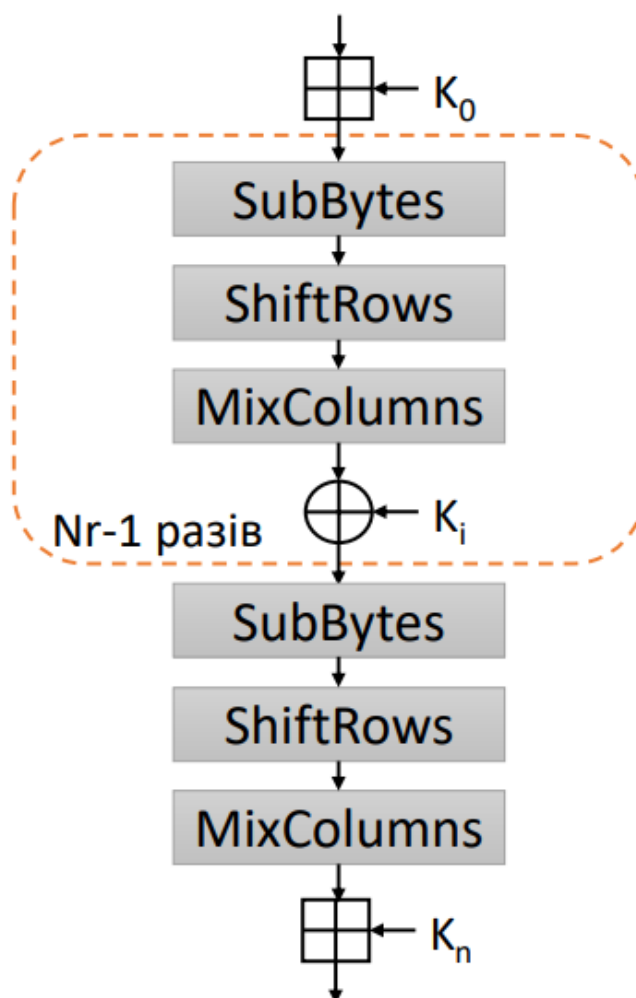


Рисунок 1.4. Структурна схема алгоритму «Калина» в режимі зашифрування

1.4 Опис алгоритму дешифрування

Аналогічно шифруванню, підготовка дешифрування передбачає отримання раундових ключів на основі початкового ключа шифрування. Для цього використовується алгоритм ітераційного розширення ключа, який забезпечує генерацію того ж самого набору раундових ключів, що і при шифруванні.

1. Виконуються операції віднімання по модулю 2^{64} , InvMixColumns, InvShiftRows, InvSubBytes. В якості K_n використовується останній раундовий ключ з шифрування.

2. Далі використовуються функції: XOR, InvMixColomn (операція обернена до операції перемішування стовпців), InvShiftRows (операція обернена до операції зсуву рядків), InvSubBytes (операція обернена до операції

підстановка байтів). Ці функції виконуються у циклі, а кількість ітерацій якого залежить від довжини ключа.

На кожній ітерації виконуються такі кроки:

- XOR: побітове додавання попереднього кроку до раундового за модулем 2;
- зворотна операція до перемішування стовпців: виконуються обернені операції для відновлення початкових значень даних, використовується інша матриця стану;
- зсув рядків у зворотному порядку: кожен рядок матриці зсувається вправо на певну кількість позицій, в залежності від розміру блоку;
- обернена операція до підстановки байтів: кожен байт матриці стану замінюється відповідно до оберненої таблиці зворотної заміни;

Загальну структуру шифрування зображено на рисунку 1.5 схематично.

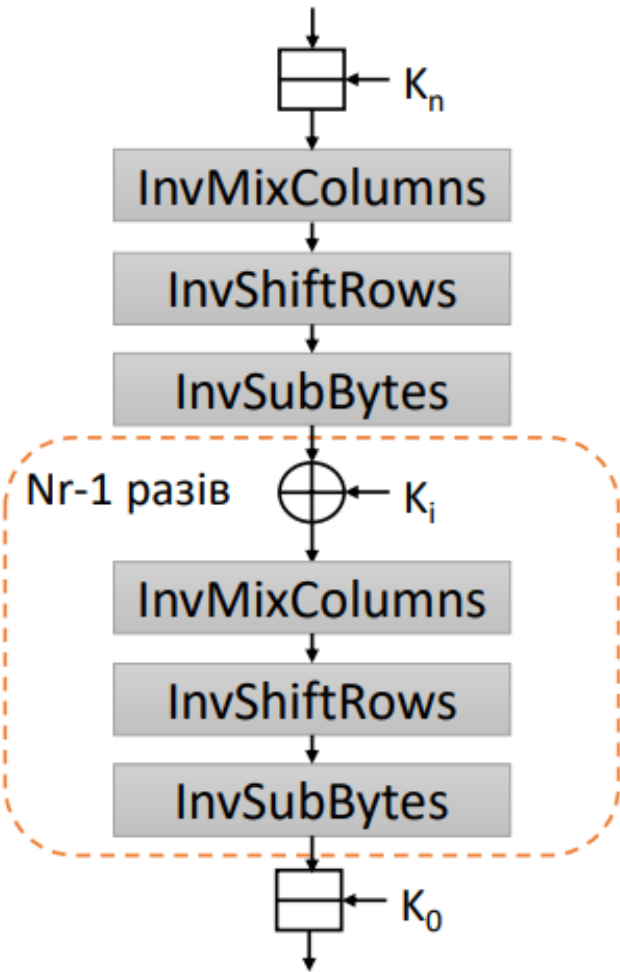


Рисунок 1.5. Структурна схема алгоритму «Калина» в режимі розшифрування

1.5 Аналіз безпеки та стійкості алгоритму

Однією з головних переваг "Калини" є велика довжина ключа, яка може сягати аж до 512 біт. Це дозволяє збільшити стійкість шифрування і зробити його більш відповідним до сучасних вимог безпеки.

Алгоритм "Калина" використовує операції шифрування, такі як перестановки, змішування та підстановки. Він має складну структуру, яка ускладнює атаки на нього. Крім того, алгоритм відповідає вимогам стандартів безпеки, таких як ISO/IEC 18033-3 та NESSIE, і успішно пройшов багато різних тестів на стійкість і продемонстрував свою високу стійкість до різних типів атак.

Однак, як і у випадку з будь-яким криптографічним алгоритмом, існує можливість знайти уразливості, які можуть бути використані для злому шифрування. Тому, хоча алгоритм "Калина" є досить безпечним, його необхідно регулярно оновлювати та тестувати на наявність можливих уразливостей.

Не можна стверджувати на 100%, що алгоритм "Калина" є абсолютно недоступним для атак. Існують деякі можливі атаки на алгоритм, такі як атака з використанням бічного каналу, деякі атаки на блокові шифри, а також можливість атак методом перебору, якщо ключ недостатньо довгий. Було виявлено деякі атаки на скорочені варіанти шифру, але вони не є практичними.

Таким чином, можна зробити висновок, що алгоритм Калина є одним з найбільш прогресивних і безпечних блокових шифрів на сьогодні. Його можна використовувати для шифрування даних в різних сферах, включаючи фінанси, медицину, телекомунікації та багато іншого.

1.6 Порівняння з іншими алгоритмами шифрування

Алгоритм "Калина" доволі новий алгоритм (2014 рік), і його ще не так багато досліджували порівняно з іншими популярними алгоритмами шифрування, такими як AES, 3DES і DES.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Однією з основних переваг "Калини" є те, що вона підтримує ключі довжиною до 512 біт, що дозволяє збільшити стійкість шифрування та зробити його більш відповідним до сучасних вимог безпеки. Крім того, "Калина" використовує безпечні операції шифрування, що ускладнює атаки на нього.

Розглянемо основні аспекти алгоритмів:

Швидкодія: алгоритм має гарну швидкої порівнюючи з AES[10] та DES. Особливо це видно при використанні ключа на 256 біт. Проте якщо розглядати ключ на 128 біт, AES є набагато швидшим.

Розмір ключа: «Калина» допускає довжину ключа від 128 до 256 біт, що забезпечує високий рівень стійкості до атак. Однак інші алгоритми, такі як Blowfish/RSA, допускають довжину ключа до 448/4096 біт, що забезпечує вищий рівень стабільності.

Стійкість до атак: "Калина" забезпечує високий рівень стійкості до атак, особливо, при використанні ключів довжиною 256 та 512 біт. Водночас AES також є дуже стійким до атак.

Підбиваючи підсумок, обираючи алгоритм шифрування, варто звертати увагу на багато факторів: стійкість, швидкодія, доступність. «Калина» є стійким, безпечним алгоритмом та гарним вибором для захисту конфіденційної інформації.

Висновки до першого розділу

В даному розділі було описано структуру алгоритму шифрування та дешифрування, представлений опис алгоритму для шифрування та розшифрування, проведено аналіз безпеки та стійкості алгоритму, а також порівняння з іншими алгоритмами.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

РОЗДІЛ 2. ПРОЄКТУВАННЯ БІБЛІОТЕКИ ТА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ДЕМОНСТРАЦІЇ РОБОТИ НАЦІОНАЛЬНОГО СТАНДАРТУ ШИФРУВАННЯ "КАЛИНА"

2.1 Опис основних функцій шифрування на прикладах

Розглянемо основні етапи формування ключів, шифрування на прикладах[2]:

1. Формування початкової матриці стану

При формуванні допоміжного ключа, виникає потреба в роз'ясненні моменту формування початкової матриці стану.

Маючи початковий ключ і умову, почнемо розгляд.

Ключ: «довільний». Потрібно сформувати допоміжний ключ на основі іншого ключа, за довжиною 128 біт.

Як бачимо з таблиці 2.1 інші значення не доступні, тобто кожен тип розміру блоку має свої власні обмеження на розмір застосованого ключа для шифрування.

Згідно з таблицею 2.1, при ключі в 128 біт маємо значення $N_k = 2$. Тоді значення $N_b = 2$.

Таблиця 2.1

Кількість раундів шифрування алгоритму «Калина»

Розмір блоку	Кількість раундів шифрування для різних довжин ключа		
	Довжина ключа 128 бітів ($N_k = 2$)	Довжина ключа 256 бітів ($N_k = 4$)	Довжина ключа 512 бітів ($N_k = 8$)
128 ($N_b = 2$)	10	14	-
256 ($N_b = 4$)	-	14	18
512 ($N_b = 8$)	-	-	18

По формулі $N_b + N_k + 1 = 2 + 2 + 1 = 5$ вирахуємо, що початкова матриця стану буде мати такий вигляд (таблиця 2.2).

Початкова матриця стану за заданих умов

05	00
00	00
00	00
00	00
00	00
00	00
00	00
00	00

2. SubByted (Підстановка байтів)

При операції підстановки байтів кожен байт матриці стану замінюється відповідно до заданої таблиці підстановки. Всього таблиць 4(рис. 2.1 - 2.4), вони використовуються послідовно.

Підстановка π_0 :																															
A8	43	5F	06	6B	75	6C	59	71	DF	87	95	17	F0	D8	09	6D	F3	1D	CB	C9	4D	2C	AF	79	E0	97	FD	6F	4B	45	39
3E	DD	A3	4F	B4	B6	9A	0E	1F	BF	15	E1	49	D2	93	C6	92	72	9E	61	D1	63	FA	EE	F4	19	D5	AD	58	A4	BB	A1
DC	F2	83	37	42	E4	7A	32	9C	CC	AB	4A	8F	6E	04	27	2E	E7	E2	5A	96	16	23	2B	C2	65	66	0F	BC	A9	47	41
34	48	FC	B7	6A	88	A5	53	86	F9	5B	DB	38	7B	C3	1E	22	33	24	28	36	C7	B2	3B	8E	77	BA	F5	14	9F	08	55
9B	4C	FE	60	5C	DA	18	46	CD	7D	21	B0	3F	1B	89	FF	EB	84	69	3A	9D	D7	D3	70	67	40	B5	DE	5D	30	91	B1
78	11	01	E5	00	68	98	A0	C5	02	A6	74	2D	0B	A2	76	B3	BE	CE	BD	AE	E9	8A	31	1C	EC	F1	99	94	AA	F6	26
2F	EF	E8	8C	35	03	D4	7F	FB	05	C1	5E	90	20	3D	82	F7	EA	0A	0D	7E	F8	50	1A	C4	07	57	B8	3C	62	E3	C8
AC	52	64	10	D0	D9	13	0C	12	29	51	B9	CF	D6	73	8D	81	54	C0	ED	4E	44	A7	2A	85	25	E6	CA	7C	8B	56	80

Рисунок 2.1. Таблиця π_0 для підстановки

Підстановка π_1 :

```

CE BB EB 92 EA CB 13 C1 E9 3A D6 B2 D2 90 17 F8
42 15 56 B4 65 1C 88 43 C5 5C 36 BA F5 57 67 8D
31 F6 64 58 9E F4 22 AA 75 0F 02 B1 DF 6D 73 4D
7C 26 2E F7 08 5D 44 3E 9F 14 C8 AE 54 10 D8 BC
1A 6B 69 F3 BD 33 AB FA D1 9B 68 4E 16 95 91 EE
4C 63 8E 5B CC 3C 19 A1 81 49 7B D9 6F 37 60 CA
E7 2B 48 FD 96 45 FC 41 12 0D 79 E5 89 8C E3 20
30 DC B7 6C 4A B5 3F 97 D4 62 2D 06 A4 A5 83 5F
2A DA C9 00 7E A2 55 BF 11 D5 9C CF 0E 0A 3D 51
7D 93 1B FE C4 47 09 86 0B 8F 9D 6A 07 B9 B0 98
18 32 71 4B EF 3B 70 A0 E4 40 FF C3 A9 E6 78 F9
8B 46 80 1E 38 E1 B8 A8 E0 0C 23 76 1D 25 24 05
F1 6E 94 28 9A 84 E8 A3 4F 77 D3 85 E2 52 F2 82
50 7A 2F 74 53 B3 61 AF 39 35 DE CD 1F 99 AC AD
72 2C DD D0 87 BE 5E A6 EC 04 C6 03 34 FB DB 59
B6 C2 01 F0 5A ED A7 66 21 7F 8A 27 C7 C0 29 D7

```

Рисунок 2.2. Таблиця π_1 для підстановки

Підстановка π_2 :

```

93 D9 9A B5 98 22 45 FC BA 6A DF 02 9F DC 51 59
4A 17 2B C2 94 F4 BB A3 62 E4 71 D4 CD 70 16 E1
49 3C C0 D8 5C 9B AD 85 53 A1 7A C8 2D E0 D1 72
A6 2C C4 E3 76 78 B7 B4 09 3B 0E 41 4C DE B2 90
25 A5 D7 03 11 00 C3 2E 92 EF 4E 12 9D 7D CB 35
10 D5 4F 9E 4D A9 55 C6 D0 7B 18 97 D3 36 E6 48
56 81 8F 77 CC 9C B9 E2 AC B8 2F 15 A4 7C DA 38
1E 0B 05 D6 14 6E 6C 7E 66 FD B1 E5 60 AF 5E 33
87 C9 F0 5D 6D 3F 88 8D C7 F7 1D E9 EC ED 80 29
27 CF 99 A8 50 0F 37 24 28 30 95 D2 3E 5B 40 83
B3 69 57 1F 07 1C 8A BC 20 EB CE 8E AB EE 31 A2
73 F9 CA 3A 1A FB 0D C1 FE FA F2 6F BD 96 DD 43
52 B6 08 F3 AE BE 19 89 32 26 B0 EA 4B 64 84 82
6B F5 79 BF 01 5F 75 63 1B 23 3D 68 2A 65 E8 91
F6 FF 13 58 F1 47 0A 7F C5 A7 E7 61 5A 06 46 44
42 04 A0 DB 39 86 54 AA 8C 34 21 8B F8 0C 74 67

```

Рисунок 2.3. Таблиця π_3 для підстановки

Підстановка π_3 :

```

68 8D CA 4D 73 4B 4E 2A D4 52 26 B3 54 1E 19 1F
22 03 46 3D 2D 4A 53 83 13 8A B7 D5 25 79 F5 BD
58 2F 0D 02 ED 51 9E 11 F2 3E 55 5E D1 16 3C 66
70 5D F3 45 40 CC E8 94 56 08 CE 1A 3A D2 E1 DF
B5 38 6E 0E E5 F4 F9 86 E9 4F D6 85 23 CF 32 99
31 14 AE EE C8 48 D3 30 A1 92 41 B1 18 C4 2C 71
72 44 15 FD 37 BE 5F AA 9B 88 D8 AB 89 9C FA 60
EA BC 62 0C 24 A6 A8 EC 67 20 DB 7C 28 DD AC 5B
34 7E 10 F1 7B 8F 63 A0 05 9A 43 77 21 BF 27 09
C3 9F B6 D7 29 C2 EB C0 A4 8B 8C 1D FB FF C1 B2
97 2E F8 65 F6 75 07 04 49 33 E4 D9 B9 D0 42 C7
6C 90 00 8E 6F 50 01 C5 DA 47 3F CD 69 A2 E2 7A
A7 C6 93 0F 0A 06 E6 2B 96 A3 1C AF 6A 12 84 39
E7 B0 82 F7 FE 9D 87 5C 81 35 DE B4 A5 FC 80 EF
CB BB 6B 76 BA 5A 7D 78 0B 95 E3 AD 74 98 3B 36
64 6D DC F0 59 A9 4C 17 7F 91 B8 C9 57 1B E0 61

```

Рисунок 2.4. Таблиця π_4 для підстановки

Отже, припустимо, що маємо довільний початковий ключ.

Ключ: 0D0C 1004 181C 2024 282C 3034 383C 0004

До байтів одного рядка поточного стану застосовується аналогічна підстановка. Правило підстановки байтів представлено на рисунку 2.5

$S_{0,i}$	0-рядок	π_0
$S_{1,i}$	1-рядок	π_1
$S_{2,i}$	2-рядок	π_2
$S_{3,i}$	3-рядок	π_3
$S_{4,i}$	4-рядок	π_0
$S_{5,i}$	5-рядок	π_1
$S_{6,i}$	6-рядок	π_2
$S_{7,i}$	7-рядок	π_3

Рисунок 2.5. Правило підстановки байтів

Перетворимо перший байт 0D за таблицею π_0 . Отже, на перехресті 0 та D стоїть значення F0. Саме це значення і буде перенесено.

	π_0															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A8	43	5F	06	6B	75	6C	59	71	DF	87	95	17	F0	D8	09
1	6D	F3	1D	CB	C9	4D	2C	AF	79	E0	97	FD	6F	4B	45	39
2	3E	DD	A3	4F	B4	B6	9A	0E	1F	BF	15	E1	49	D2	93	C6
3	92	72	9E	61	D1	63	FA	EE	F4	19	D5	AD	58	A4	BB	A1
4	DC	F2	83	37	42	E4	7A	32	9C	CC	AB	4A	8F	6E	04	27
5	2E	E7	E2	5A	96	16	23	2B	C2	65	66	0F	BC	A9	47	41
6	34	48	FC	B7	6A	88	A5	53	86	F9	5B	DB	38	7B	C3	1E
7	22	33	24	28	36	C7	B2	3B	8E	77	BA	F5	14	9F	08	55
8	9B	4C	FE	60	5C	DA	18	46	CD	7D	21	B0	3F	1B	89	FF
9	EB	84	69	3A	9D	D7	D3	70	67	40	B5	DE	5D	30	91	B1
A	78	11	01	E5	00	68	98	A0	C5	02	A6	74	2D	0B	A2	76
B	B3	BE	CE	BD	AE	E9	8A	31	1C	EC	F1	99	94	AA	F6	26
C	2F	EF	E8	8C	35	03	D4	7F	FB	05	C1	5E	90	20	3D	82
D	F7	EA	0A	0D	7E	F8	50	1A	C4	07	57	B8	3C	62	E3	C8
E	AC	52	64	10	D0	D9	13	0C	12	29	51	B9	CF	D6	73	8D
F	81	54	C0	ED	4E	44	A7	2A	85	25	E6	CA	7C	8B	56	80

Рисунок 2.6. Таблиця π_0

Другий байт 0C перетворюється за таблицею π_1 . Маємо результат D2.

Третій байт, відповідно, буде перетворюється за таблицею π_2 , четвертий — π_3 . П'ятий байт буде перетворюватися за таблицею π_0 , згідно з рисунком 2.6

Таким чином, отримуємо кінцевий результат зображений у таблиці 2.3.

Кінцевий результат за заданих умов

0D	28	→	F0	1F
0C	2C		D2	DF
10	30		4A	A6
04	34		73	40
18	38		79	F4
1C	3C		F5	54
20	00		49	93
24	04		ED	73

3. ShiftPows (Зсув рядків)

Рядки стану циклічно зсувають праворуч на різну кількість байтів, залежно від розміру блоку. Все залежить від типу блоку: 128, 256 чи 512-бітовий блок

Блок 128-бітів:

0D	28	→	0D	28
0C	2C		0C	2C
10	30		10	30
04	34		04	34
18	38		38	18
1C	3C		3C	1C
20	00		00	20
24	04		04	24

Рисунок 2.7. Результат зсуву рядків

Блок 256-бітів:

0D	28	F0	1F	→	0D	28	F0	1F
0C	2C	D2	DF		0C	2C	D2	DF
10	30	4A	A6		30	10	4A	A6
04	34	73	40		34	04	73	40
18	38	79	F4		79	38	18	F4
1C	3C	F5	54		F5	3C	1C	54
20	00	49	93		93	00	49	20
24	04	ED	73		73	04	ED	24

Рисунок 2.8. Результат зсуву рядків

Блок 512-бітів:

0D	28	F0	1F	01	08	06	07	→	0D	28	F0	1F	01	08	06	07
0C	2C	D2	DF	05	01	08	06		2C	0C	D2	DF	05	01	08	06
10	30	4A	A6	01	05	01	08		4A	30	10	A6	01	05	01	08
04	34	73	40	01	01	05	01		40	34	73	04	01	01	05	01
18	38	79	F4	04	01	01	05		04	38	79	F4	18	01	01	05
1C	3C	F5	54	07	04	01	01		04	3C	F5	54	07	1C	01	01
20	00	49	93	06	07	04	01		04	00	49	93	06	07	04	01
24	04	ED	73	08	06	07	04		04	04	ED	73	08	06	07	24

Рисунок 2.9. Результат зсуву рядків

4. MixColumn (Перемішування стовпців)

Під час виконання операції перемішування стовпців, виконуються такі основні операції: XOR, ділення по модулю.

Для того, щоб отримати лише перше число фінальної матриці (0x62) потрібно виконати XOR добутку кожного елементу рядка фіксований многочлен з кожним елементом стовпчика матриці стану.

Якщо ж добуток двох чисел, після того, як скоротили однакові доданки, перевищує допустиме значення (255 в десятковій системі числення або 0xff в 16-ковій системі числення або x^7), то потрібно робити ділення на незвідний многочлен ($x^8 + x^4 + x^3 + x^2 + 1$). Розглянемо на прикладі (рис. 2.10)

01	01	05	01	08	06	07	04		75	71		62	ED
04	01	01	05	01	08	06	07		BB	3A		C9	51
07	04	01	01	05	01	08	06		9A	DF		7C	31
06	07	04	01	01	05	01	08		4D	B3		6E	D6
08	06	07	04	01	01	05	01	.	17	6B	=	6A	24
01	08	06	07	04	01	01	05		90	CB		BF	C7
05	01	08	06	07	04	01	01		51	45		41	C1
01	05	01	08	06	07	04	01		1F	2A		33	82

Рисунок 2.10. Приклад отримання першого значення

Маємо перший доданок: $0x01 \cdot 0x75$. Представимо перший і другий множник в «х-форму»:

$$0x01_{16} = 0000\ 0001_2 = 0 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = 1$$

$$0x75_{16} = 0111\ 0101_2 = 0 \cdot x^7 + 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^6 + x^5 + x^4 + x^2 + 1$$

Виконаємо множення:

$$(0x01 \cdot 0x75) = 1 \cdot (x^6 + x^5 + x^4 + x^2 + 1) = x^6 + x^5 + x^4 + x^2 + 1 = 0111\ 0101$$

Як бачимо, добуток цих двох чисел не перевищує 7-му степінь, тому можна переводити значення у 2-кову СЧ.

Розглянемо інший приклад: $0x05 \cdot 0x9a$. Робимо аналогічні дії: переписуємо у двійкову СЧ, представляємо в форматі x , множимо, перевіряємо:

$$0x05_{16} = 0000\ 0101_2 = 0 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^2 + 1$$

$$0x9a_{16} = 1001\ 1010_2 = 1 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = x^7 + x^4 + x^3 + x$$

Множимо:

$$(0x05 \cdot 0x9a) = (x^2 + 1) \cdot (x^7 + x^4 + x^3 + x) = x^9 + x^6 + x^5 + x^3 + x^7 + x^4 + x^3 + x = x^9 + x^7 + x^6 + x^5 + x^4 + x$$

Як бачимо, в результаті маємо степінь 9, що перевищує значення 7. Тому потрібно ділити на незвідний многочлен — $x^8 + x^4 + x^3 + x^2 + 1$. Якщо є однакові доданки, скорочуємо їх.

$$\begin{array}{r|l} x^9 + x^7 + x^6 + x^5 + x^4 + x & x^8 + x^4 + x^3 + x^2 + 1 \\ \hline x^9 + x^5 + x^4 + x^3 + x & x \\ \hline x^7 + x^6 + x^3 & \end{array}$$

Таким чином проводимо усі додавання та отримуємо наступне:

$$\begin{aligned} & (0x01 \cdot 0x75) \oplus (0x01 \cdot 0xbb) \oplus (0x05 \cdot 0x9a) \oplus (0x01 \cdot 0x4d) \oplus \\ & \oplus (0x08 \cdot 0x17) \oplus (0x06 \cdot 0x90) \oplus (0x07 \cdot 0x51) \oplus (0x04 \cdot 0x1f) = \\ & = 0111\ 0101_2 \oplus 1011\ 1011_2 \oplus 1100\ 1000_2 \oplus 0100\ 1101_2 \oplus 1011\ 1000_2 \oplus \\ & \oplus 0100\ 0111_2 \oplus 1010\ 1010_2 \oplus 0111\ 1100_2 = 0110\ 0010_2 = 62_{16} \end{aligned}$$

Для отримання наступного значення $0xed$, слід використовувати перший рядочок фіксованої матриці та сусідній стовпчик матриці стану. Приклад подальшої реалізації представлений на рисунку 2.11.

01	01	05	01	08	06	07	04		75	71		62	ED
04	01	01	05	01	08	06	07		BB	3A		C9	51
07	04	01	01	05	01	08	06		9A	DF		7C	31
06	07	04	01	01	05	01	08		4D	B3		6E	D6
08	06	07	04	01	01	05	01		17	6B		6A	24
01	08	06	07	04	01	01	05		90	CB		BF	C7
05	01	08	06	07	04	01	01		51	45		41	C1
01	05	01	08	06	07	04	01		1F	2A		33	82

Рисунок 2.11. Покрокова реалізація множення матриць

5. Додавання по модулю 2^{64} ($\oplus$)

При додаванні по модулю 2^{64} виконується ряд простих операцій: інвертування, побітове додавання. Розглянемо приклад (рис. 2.12):

$$\begin{pmatrix} C3 & 5A \\ 1E & 21 \\ E8 & A4 \\ A8 & 35 \\ 7E & FD \\ 2C & B2 \\ ED & 5B \\ 24 & 92 \end{pmatrix} \boxplus \begin{pmatrix} 95 & B7 \\ CD & 26 \\ 56 & 53 \\ 60 & E1 \\ 91 & 71 \\ D3 & 80 \\ 27 & F3 \\ 65 & 81 \end{pmatrix} = \begin{pmatrix} 58 & 11 \\ EC & 48 \\ 3E & F7 \\ 09 & 16 \\ 10 & 6F \\ 00 & 33 \\ 15 & 4F \\ 8A & 14 \end{pmatrix}$$

Рисунок 2.12. Додавання по модулю 2^{64}

Розкладемо кожний доданок в стрічку:

C31E E8A8 7E2C ED24 5A21 A435 FDB2 5B92

95CD 5660 91D3 2765 B726 53E1 7180 F381

Переве́рнемо обидва доданки[8] і розділимо кожен навпіл (рис. 2.13, 2.14):

C31EE8A87E2CED245A21A435FDB25B92 (big endian)

925BB2FD35A4215A24ED2C7EA8E81EC3 (little endian)

Рисунок 2.13. Інверсія по 2 байти

95CD566091D32765B72653E17180F381 (big endian)

81F38071E15326B76527D3916056CD95 (little endian)

Рисунок 2.14. Інверсія по 2 байти

Над кожним доданком однакового кольору виконаємо побітове додавання (рис. 2.15):

$(925B\ B2FD\ 35A4\ 215A + 81F3\ 8071\ E153\ 26B7) \bmod 2^{64} = 1414F\ 336F\ 16F7\ 4811$

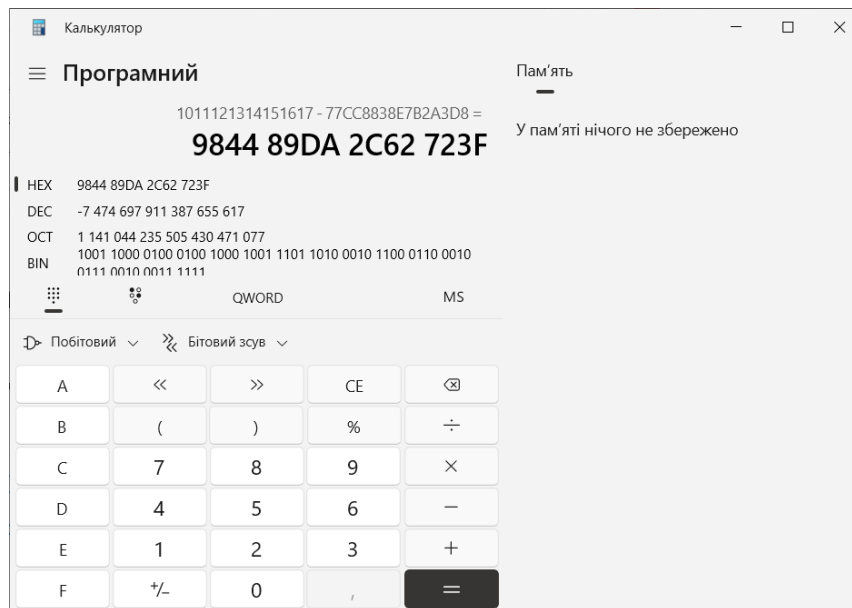


Рисунок 2.16. Результат у Windows Calculator

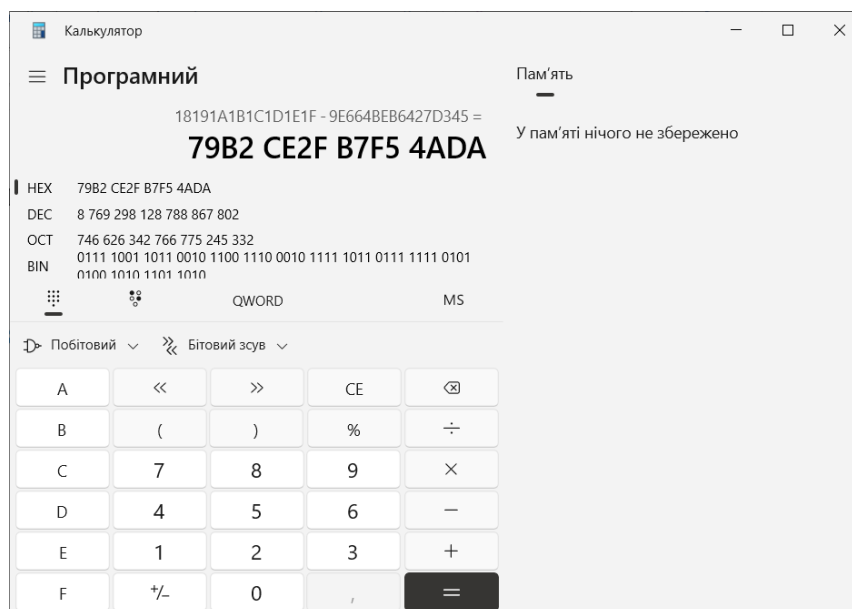


Рисунок 2.17. Результат у Windows Calculator

Як бачимо, від меншого віднімається більше, і результат повинен бути від'ємний. В таких випадках, від'ємний результат (-67bb 7625 d39d 8dc1) переводиться у двійкову систему числення, кожен біт якого, інвертується (0 стає 1, а 1 стає 0). Це правило застосовується до всього результату. Виключення становить останній біт останнього числа: до нього додається 1. Після цього результат переводиться в 16-кову СЧ (9844 89da 2c62 723f). Тільки після цих операції, ми маємо той самий результат, що і в Windows Calculator.

Провівши аналогічні операції з іншим результатом отримуємо: 9844 89DA 2C62 723F 79B2 CE2F B7F5 4ADA. Фінальний результат: DA4A F5B7 2FCE B279 3F72 622C DA89 4498

2. InvMixColumns

При операції, оберненій до операції перемішування стовпців, виконуються ті ж самі дії, тільки операції виконуються на фіксований многочлен $c^{-1}(x)$ (рис. 2.18).

$$\begin{pmatrix} AD & 95 & 76 & A8 & 2F & 49 & D7 & CA \\ CA & AD & 95 & 76 & A8 & 2F & 49 & D7 \\ D7 & CA & AD & 95 & 76 & A8 & 2F & 49 \\ 49 & D7 & CA & AD & 95 & 76 & A8 & 2F \\ 2F & 49 & D7 & CA & AD & 95 & 76 & A8 \\ A8 & 2F & 49 & D7 & CA & AD & 95 & 76 \\ 76 & A8 & 2F & 49 & D7 & CA & AD & 95 \\ 95 & 76 & A8 & 2F & 49 & D7 & CA & AD \end{pmatrix}$$

Рисунок 2.18. Фіксований многочлен $c^{-1}(x)$

3. InvShiftRows

При операції, оберненій до операції зсув рядків[4], виконується зсув, тільки за іншим правилом. Результат показаний на рис. 2.19 для 128-бітного та 256-бітного блоку, та на рис. 2.20 для 512-бітного блоку.

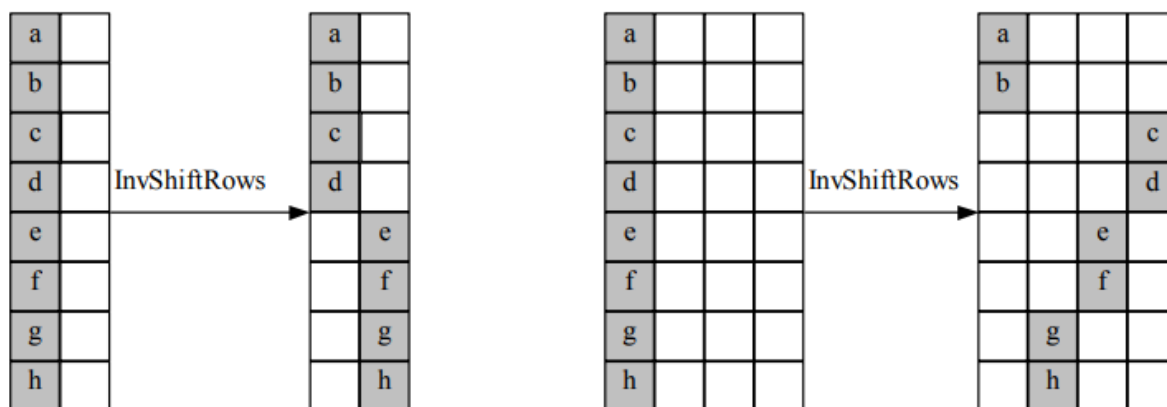


Рисунок 2.19. InvShiftRows для 128-бітного та 256-бітного блоку

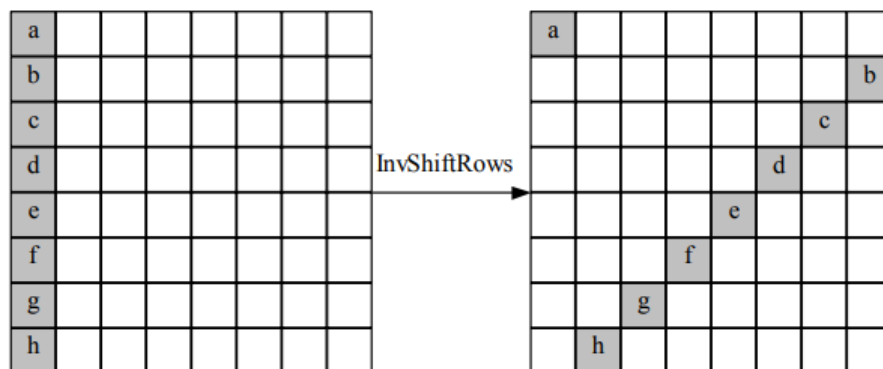


Рисунок 2.20. InvShiftRows для 512-бітного блоку

4. InvSubBytes

5. При операції, InvSubBytes, виконуються такі ж дії, що і для SubBytes, тільки заміна відбувається відносно чотирьох інших таблиць (рис. 2.21-2.24).

Підстановка $_{-1}\pi_0$:

A4	A2	A9	C5	4E	C9	03	D9	7E	0F	D2	AD	E7	D3	27	5B
E3	A1	E8	E6	7C	2A	55	0C	86	39	D7	8D	B8	12	6F	28
CD	8A	70	56	72	F9	BF	4F	73	E9	F7	57	16	AC	50	C0
9D	B7	47	71	60	C4	74	43	6C	1F	93	77	DC	CE	20	8C
99	5F	44	01	F5	1E	87	5E	61	2C	4B	1D	81	15	F4	23
D6	EA	E1	67	F1	7F	FE	DA	3C	07	53	6A	84	9C	CB	02
83	33	DD	35	E2	59	5A	98	A5	92	64	04	06	10	4D	1C
97	08	31	EE	AB	05	AF	79	A0	18	46	6D	FC	89	D4	C7
FF	F0	CF	42	91	F8	68	0A	65	8E	B6	FD	C3	EF	78	4C
CC	9E	30	2E	BC	0B	54	1A	A6	BB	26	80	48	94	32	7D
A7	3F	AE	22	3D	66	AA	F6	00	5D	BD	4A	E0	3B	B4	17
8B	9F	76	B0	24	9A	25	63	DB	EB	7A	3E	5C	B3	B1	29
F2	CA	58	6E	D8	A8	2F	75	DF	14	FB	13	49	88	B2	EC
E4	34	2D	96	C6	3A	ED	95	0E	E5	85	6B	40	21	9B	09
19	2B	52	DE	45	A3	FA	51	C2	B5	D1	90	B9	F3	37	C1
0D	BA	41	11	38	7B	BE	D0	D5	69	36	C8	62	1B	82	8F

Рисунок 2.21. Таблиця $_{-1}\pi_0$ для підстановки в InvSubBytes

Підстановка $_{-1}\pi_1$:

83	F2	2A	EB	E9	BF	7B	9C	34	96	8D	98	B9	69	8C	29
3D	88	68	06	39	11	4C	0E	A0	56	40	92	15	BC	B3	DC
6F	F8	26	BA	BE	BD	31	FB	C3	FE	80	61	E1	7A	32	D2
70	20	A1	45	EC	D9	1A	5D	B4	D8	09	A5	55	8E	37	76
A9	67	10	17	36	65	B1	95	62	59	74	A3	50	2F	4B	C8
D0	8F	CD	D4	3C	86	12	1D	23	EF	F4	53	19	35	E6	7F
5E	D6	79	51	22	14	F7	1E	4A	42	9B	41	73	2D	C1	5C
A6	A2	E0	2E	D3	28	BB	C9	AE	6A	D1	5A	30	90	84	F9
B2	58	CF	7E	C5	CB	97	E4	16	6C	FA	B0	6D	1F	52	99
0D	4E	03	91	C2	4D	64	77	9F	DD	C4	49	8A	9A	24	38
A7	57	85	C7	7C	7D	E7	F6	B7	AC	27	46	DE	DF	3B	D7
9E	2B	0B	D5	13	75	F0	72	B6	9D	1B	01	3F	44	E5	87
FD	07	F1	AB	94	18	EA	FC	3A	82	5F	05	54	DB	00	8B
E3	48	0C	CA	78	89	0A	FF	3E	5B	81	EE	71	E2	DA	2C
B8	B5	CC	6E	A8	6B	AD	60	C6	08	04	02	E8	F5	4F	A4
F3	C0	CE	43	25	1C	21	33	0F	AF	47	ED	66	63	93	AA

Рисунок 2.22. Таблиця $_{-1}\pi_1$ для підстановки в InvSubBytes

Підстановка $_{-1}\pi_2$:

45	D4	0B	43	F1	72	ED	A4	C2	38	E6	71	FD	B6	3A	95
50	44	4B	E2	74	6B	1E	11	5A	C6	B4	D8	A5	8A	70	A3
A8	FA	05	D9	97	40	C9	90	98	8F	DC	12	31	2C	47	6A
99	AE	C8	7F	F9	4F	5D	96	6F	F4	B3	39	21	DA	9C	85
9E	3B	F0	BF	EF	06	EE	E5	5F	20	10	CC	3C	54	4A	52
94	0E	C0	28	F6	56	60	A2	E3	0F	EC	9D	24	83	7E	D5
7C	EB	18	D7	CD	DD	78	FF	DB	A1	09	D0	76	84	75	BB
1D	1A	2F	B0	FE	D6	34	63	35	D2	2A	59	6D	4D	77	E7
8E	61	CF	9F	CE	27	F5	80	86	C7	A6	FB	F8	87	AB	62
3F	DF	48	00	14	9A	BD	5B	04	92	02	25	65	4C	53	0C
F2	29	AF	17	6C	41	30	E9	93	55	F7	AC	68	26	C4	7D
CA	7A	3E	A0	37	03	C1	36	69	66	08	16	A7	BC	C5	D3
22	B7	13	46	32	E8	57	88	2B	81	B2	4E	64	1C	AA	91
58	2E	9B	5C	1B	51	73	42	23	01	6E	F3	0D	BE	3D	0A
2D	1F	67	33	19	7B	5E	EA	DE	8B	CB	A9	8C	8D	AD	49
82	E4	BA	C3	15	D1	E0	89	FC	B1	B9	B5	07	79	B8	E1

Рисунок 2.23. Таблиця $_{-1}\pi_2$ для підстановки в InvSubBytes

Підстановка $_{-1}\pi_3$:

B2	B6	23	11	A7	88	C5	A6	39	8F	C4	E8	73	22	43	C3
82	27	CD	18	51	62	2D	F7	5C	0E	3B	FD	CA	9B	0D	0F
79	8C	10	4C	74	1C	0A	8E	7C	94	07	C7	5E	14	A1	21
57	50	4E	A9	80	D9	EF	64	41	CF	3C	EE	2E	13	29	BA
34	5A	AE	8A	61	33	12	B9	55	A8	15	05	F6	03	06	49
B5	25	09	16	0C	2A	38	FC	20	F4	E5	7F	D7	31	2B	66
6F	FF	72	86	F0	A3	2F	78	00	BC	CC	E2	B0	F1	42	B4
30	5F	60	04	EC	A5	E3	8B	E7	1D	BF	84	7B	E6	81	F8
DE	D8	D2	17	CE	4B	47	D6	69	6C	19	99	9A	01	B3	85
B1	F9	59	C2	37	E9	C8	A0	ED	4F	89	68	6D	D5	26	91
87	58	BD	C9	98	DC	75	C0	76	F5	67	6B	7E	EB	52	CB
D1	5B	9F	0B	DB	40	92	1A	FA	AC	E4	E1	71	1F	65	8D
97	9E	95	90	5D	B7	C1	AF	54	FB	02	E0	35	BB	3A	4D
AD	2C	3D	56	08	1B	4A	93	6A	AB	B8	7A	F2	7D	DA	3F
FE	3E	BE	EA	AA	44	C6	D0	36	48	70	96	77	24	53	DF
F3	83	28	32	45	1E	A4	D3	A2	46	6E	9C	DD	63	D4	9D

Рисунок 2.24. Таблиця $_{-1}\pi_3$ для підстановки в InvSubBytes

2.3 Обґрунтування вибору мови програмування

Однією з сильних сторін Python є його простота та легкість використання. Python має простий синтаксис і динамічний тип[14], що дозволяє швидко створювати прототипи та тестувати код. Це особливо корисно для алгоритмів шифрування, розробка яких може бути складною та трудомісткою.

Python також має велику кількість бібліотек, доступних для криптографії, таких як PyCrypto та Cryptography. Ці бібліотеки надають можливість виконувати криптографічні операції, такі як шифрування та дешифрування, генерація ключів тощо.

Python також підтримує кілька платформ, включаючи Windows, Linux і MacOS, що робить його ідеальним для розробки кросплатформних криптографічних алгоритмів. Крім того, Python є відкритою мовою

програмування, яка дозволяє розробникам робити внесок у вдосконалення мови та розширення її можливостей.

Python також має вбудовану підтримку бітових даних, що важливо для реалізації криптографічних алгоритмів. Також Python має вбудовані функції для роботи з хеш-функціями, що корисно для реалізації підтримки різних алгоритмів хешування.

2.4 Обґрунтування вибору середовища розробки

По-перше, PyCharm є потужним інструментом для розробки мовою програмування Python. З його допомогою можна швидко створювати, редагувати та налагоджувати Python-код, що є основою проєкту. PyCharm надає багато корисних функцій, таких як автодоповнення коду, перевірка синтаксису, рефакторинг та багато іншого, що сприяє покращенню продуктивності розробників.

Крім того, PyCharm має інтегровану підтримку системи контролю версій, зокрема Git. Це дозволяє легко відстежувати зміни в коді, робити коміти, гілки, злиття гілок та багато іншого. Така інтеграція спрощує спільну роботу над проєктом і забезпечує ефективне управління версіями коду.

PyCharm пропонує розширення і плагіни, що дозволяють налаштувати середовище розробки під особисті потреби. Це включає підтримку інструментів для тестування, аналізу коду, збірки та розгортання програмного продукту. Ми можемо використовувати ці розширення, щоб забезпечити високу якість нашого коду та зручність розробки.

Додатковою перевагою використання PyCharm є активна спільнота користувачів та розробників, яка надає підтримку та знання у випадку виникнення питань або проблем. Є можливість звертатися до форумів, документації та онлайн-ресурсів, щоб отримати відповіді на свої запитання та дізнатися про нові можливості PyCharm.

Загалом, обґрунтований вибір середовища розробки є ключовим етапом у процесі реалізації проєкту. Вибір PyCharm для проєкту забезпечує

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

потужність, гнучкість та зручність в розробці на мові Python, підтримку системи контролю версій та доступ до розширень для налаштування середовища розробки під наші потреби.

2.5 Програмна реалізація

Для ефективного написання коду, варто розділити задачу на підзадачі. Тому було прийняте рішення написати функції для основних операцій: SubBytes (підстановка байтів), ShiftRows (зсув рядків), MixColomn (перемішування стовпців), XOR, додавання по модулю 2^{64} , а далі «склеювати» потрібні функції як пазл.

Основні етапи реалізації шифрування для алгоритму зображено на рисунку 2.25

При проєктуванні модулів, було вирішено працювати з єдиним одним форматом введених даних. Оскільки в мові програмування Python, відсутній тип даних масив, то було обрано такий тип даних як список[9]. В Python можна під'єднати бібліотеку numpy, яка працює з масивами, проте дозволений формат — тільки десяткові числа, що абсолютно не підходить для даного випадку.

Одна з найскладніших функцій — додавання по модулю 2^{64} (Add2in64) [7]. При отриманні початкових даних, масиви інвертуються, діляться навпіл, з'єднуються, додаються і знову об'єднуються. Для реалізації цієї однієї операції були залучені додаткові 4 функції, що знімали навантаження з основної функції та виконували повторюванні операції. Найважчим було навчити програму правильно додавати в стовпчик (FirstFinalAddition, SecondFinalAddition): потрібно було прописати кожен варіант розвитку операцій у двійковій СЧ. Програмний код наведений в додатку А. Результат операції та початкові дані наведені на попередніх рисунках 2.25 і 2.26. У ролі змінної keyToHex виступав масив `['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']`. У ролі змінної StateMatrix виступав масив `['0x5', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0']`

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

```
print(f'(state[0].add_rkey): {Add2in64(keyToHex, StateMatrix)}')
```

Рисунок 2.25. Початкові дані

```
(state[0].add_rkey): [['0x5', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
```

Рисунок 2.26. Результат операції

При роботі з функцією ShiftRows потрібно було зробити правильний зсув рядків, який індивідуальний різного розміру ключа: 128, 256 та 512 біт. Програмний код представлений на рисунку 2.27.

```
# Зсув рядків
7 usages
def ShiftRows(arr): # ПРИЙМАЄ МАСИВ, НЕ СПИСОК!
    tmp = 0
    if len(arr[0]) == 2:
        for i in range(len(arr)): # пройдемося по рядках
            for j in range(len(arr[0])): # пройдемося по стовп
                if i >= 4 and j == 0: # 4/0
                    tmp = arr[i][j] # 6b
                    arr[i][j] = arr[i][j + 1] # 17 17
                    arr[i][j + 1] = tmp # 17 6b

    elif len(arr[0]) == 4:
        j = 0
        for i in range(len(arr)): # пройдемося по рядках
            if (i + 1) % 2 != 0:
                tmp = arr[i][j]
                arr[i][j] = arr[i][0]
                arr[i][0] = tmp
            elif (i + 1) % 2 == 0:
                tmp = arr[i][j]
                arr[i][j] = arr[i][0]
                arr[i][0] = tmp
                j += 1
                i += 1

    elif len(arr[0]) == 8:
        for i in range(len(arr)): # пройдемося по рядках
            for j in range(len(arr[0])): # пройдемося по стовп
                if i == j:
                    tmp = arr[i][j]
                    arr[i][j] = arr[i][0]
                    arr[i][0] = tmp

    return arr
```

Рисунок 2.27. Програмний код

Результат операції та початкові дані наведені на рисунках 2.28 та 2.29. У ролі змінної виступав результат попередньої дії: [['0x75', '0x71'], ['0xbb', '0x3a'],

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

```
['0x9a', '0xdf'], ['0x4d', '0xb3'], ['0x6b', '0x17'], ['0xcb', '0x90'], ['0x45', '0x51'],
['0x2a', '0x1f']]
```

```
ShiftRows_SybBytes_K0PlusKEY = ShiftRows(SybBytes_K0PlusKEY)
print(f'(state[0].s_row):{ShiftRows_SybBytes_K0PlusKEY}')
```

Рисунок 2.28. Початкові дані

```
(state[0].s_row):[['0x75', '0x71'], ['0xbb', '0x3a'], ['0x9a', '0xdf'], ['0x4d', '0xb3'],
['0x17', '0x6b'], ['0x90', '0xcb'], ['0x51', '0x45'], ['0x1f', '0x2a']]
```

Рисунок 2.29. Результат операції

При роботі над функцією MixColumn було залучено дві додаткові функції. Multiply() виконувала значне розвантаження основної функції multiplyArr() таке як: скорочення однакових доданків після множення многочленів, ділення по незвідному многочлену, повертає кінцевий проміжний результат. Функція xorElemMatrix() забезпечувала правильну виключну диз'юнкцію усіх елементів масиву.

Програмний код представлений у додатку Б. Результат операції та початкові дані наведені в рисунках 2.30 і 2.31. У ролі змінної ShiftRows_SybBytes_K0PlusKEY виступала матриця: [['0x75', '0x71'], ['0xbb', '0x3a'], ['0x9a', '0xdf'], ['0x4d', '0xb3'], ['0x17', '0x6b'], ['0x90', '0xcb'], ['0x51', '0x45'], ['0x1f', '0x2a']]

```
Mix_ShiftRows_SybBytes_K0PlusKEY = multiplyArr(stateMatrix, ShiftRows_SybBytes_K0PlusKEY)
print(f'(state[0].m_col):{Mix_ShiftRows_SybBytes_K0PlusKEY}')
```

Рисунок 2.30. Початкові дані

```
(state[0].m_col):[['0x62', '0xed'], ['0xc9', '0x51'], ['0x7c', '0x31'], ['0x6e', '0xd6'],
['0x6a', '0x24'], ['0xbf', '0xc7'], ['0x41', '0xc1'], ['0x33', '0x82']]
```

Рисунок 2.31.Результат операції

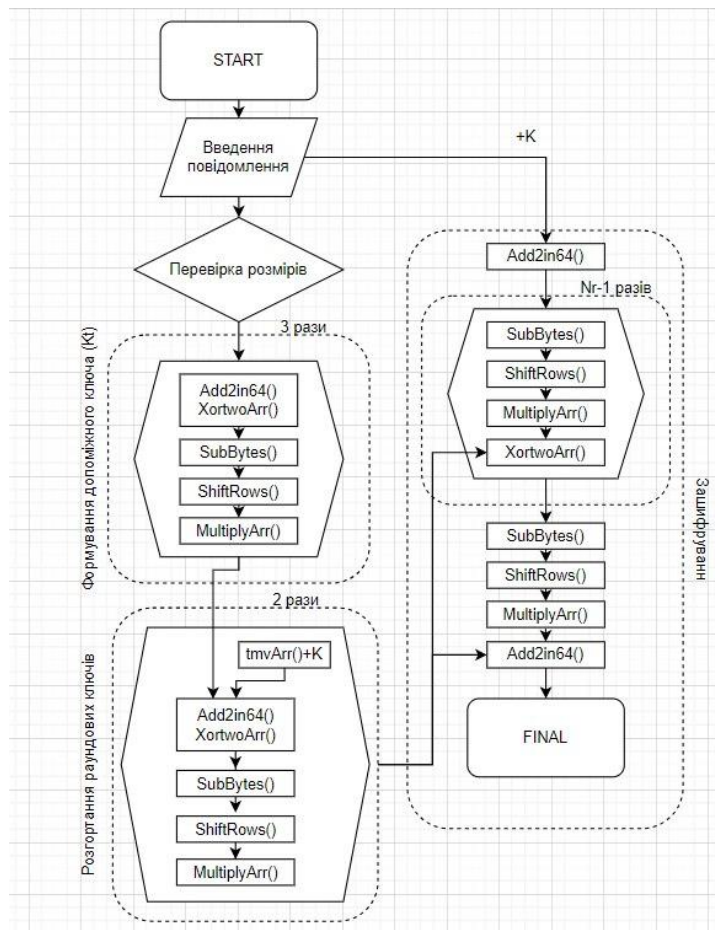


Рисунок 2.32. Процес проєктування алгоритму шифрування у вигляді алгоритмічної схеми

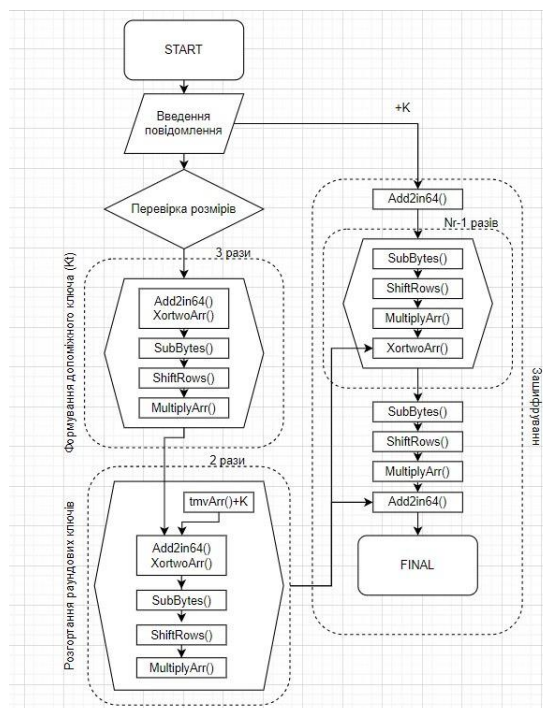


Рисунок 2.33. Процес проєктування алгоритму дешифрування у вигляді алгоритмічної схеми

Висновки до другого розділу

В другому розділі було детально розглянуто основні дії формування ключів, шифрування та дешифрування — формування початкової матриці, SubBytes, ShiftRows, MixColumns, додавання та віднімання по модулю 2^{64} , InvSubBytes, InvShiftRows, InvMixColumns. Обґрунтовано вибір мови програмування та середовища розробки. Наведено деталі розробки програмного коду.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОПИС РОБОТИ

3.1 Опис роботи бібліотеки та додатка

Бібліотека складається з базового набору функцій, необхідних функцій для шифрування:

ConvertToHex(): перетворює вхідні дані в 16-кову СЧ. Приймає в якості вхідного параметра рядок. Повертає масив з елементами в 16-ковій СЧ.

ContRounds(): підрахунок кількості раундів, і повернення значень N_b , N_k . Приймає вхідне повідомлення і ключ.

InitialStateMatrix(): створює початкову матрицю стану. Приймає значення N_b , N_k . Повертає масив.

ShiftRows()/InvShiftRows: зсув рядків вхідного масиву.

SubBytes()/InvSubBytes: підстановка байтів згідно з таблицею.

multiplyArr(): функція для виконання операції MixColumn та InvMixColumn. Приймає два значення: матриця стану (відповідно до шифрування чи дешифрування) та масив. Повертає масив.

XortwoArr(): допоміжна функція для розвантаження multiplyArr(). Виконує виключну диз'юнкцію для всіх результатів множення двох матриць.

tmvArr(): виконує генерацію раундової константи. Приймає число, рівне лічильнику.

JustAdd(): виконує просте додавання допоміжного ключа K_t з tmv_i при генерації парних раундових ключів.

cyclicRightShift(): виконує зсув парних ключів для створення непарних раундових ключів.

Subtraction2in64(): віднімання по модулю 2^{64} .

SubArr(): допоміжна функція Subtraction2in64(). Виконує проміжні результати: віднімання, заміна байтів.

FirstFinalAddition()/SecondFinalAddition(): виконує бітове додавання в стовпчик.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

multiply(): виконує значне розвантаження функції multiplyArr: скорочення однакових доданків після множення многочленів, ділення по незвідному многочлену. Повертає кінцевий проміжний результат.

Також доступні й дві основні функції: шифрування KalynaEncryption() та дешифрування KalynaDecryption() з вхідними параметрами: ключ та повідомлення.

Всі функції, змінні, константи доступні у головному файлі, де імпортується бібліотека. (рисунок. 3.1-3.4).

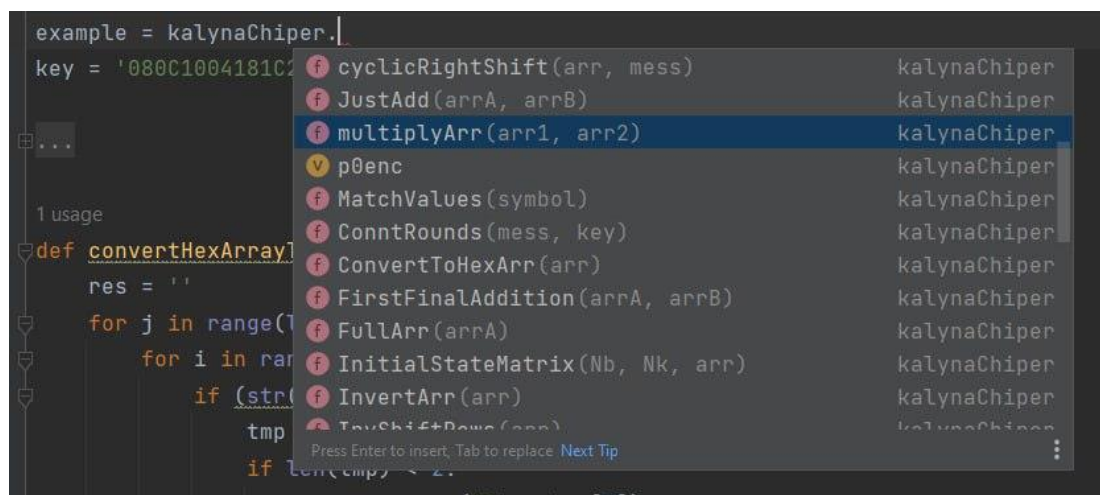


Рисунок 3.1. Доступність виклику внутрішніх функцій бібліотеки kalynaChiper

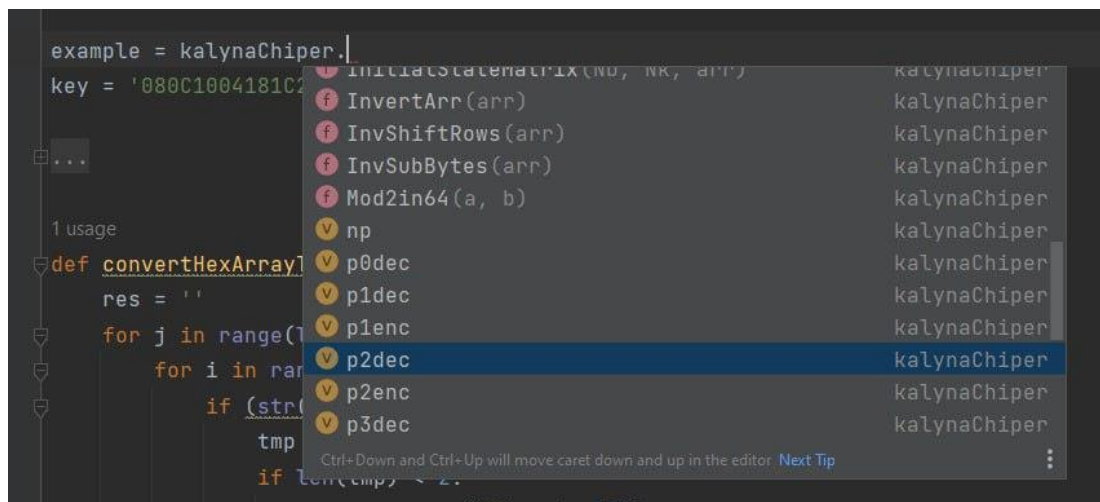


Рисунок 3.2. Доступність виклику внутрішніх функцій бібліотеки kalynaChiper

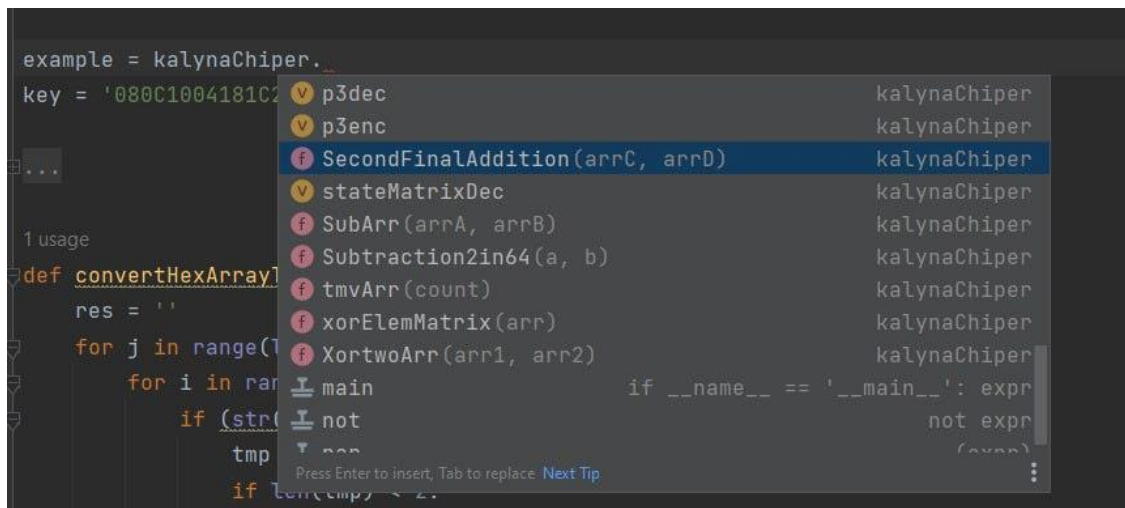


Рисунок 3.3. Доступність виклику внутрішніх функцій бібліотеки kalynaChiper

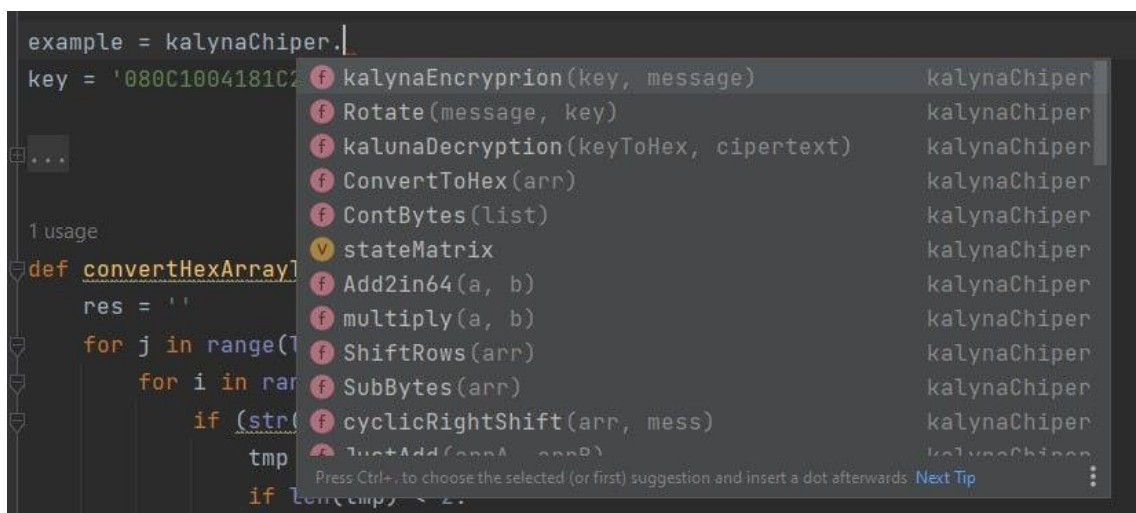


Рисунок 3.4. Доступність виклику внутрішніх функцій бібліотеки kalynaChiper

Користувач, який буде використовувати бібліотеку може частково використовувати чи перевикористовувати наявний в бібліотеці функціонал. Даний підхід дозволяє використовувати додаткові рішення для створення ключів в зовнішніх бібліотеках, зберігати їх в базах даних для зменшення загальної вразливості програмного рішення. Тобто, ключі та кодова реалізація алгоритму шифрування/дешифрування не знаходяться у спільному доступі. Така реалізація дозволяє використати алгоритм шифрування для забезпечення чутливих даних користувача від несанкціонованого доступу. На основі такого рішення може бути розроблений, наприклад, додаток для шифрування медичних даних пацієнта, карткових даних банків, аналогічні.

Для відображення результатів було застосовано пакет tkinter — стандартний інтерфейс Python, що встановлена типово. Tkinter не є окремою бібліотекою чи надбудовою, тому відсутнє додаткове навантаження на програму. На рисунку 3.5-3.6 зображено найпростіший графічний варіант використання кодової реалізації алгоритму шифрування «Калина».

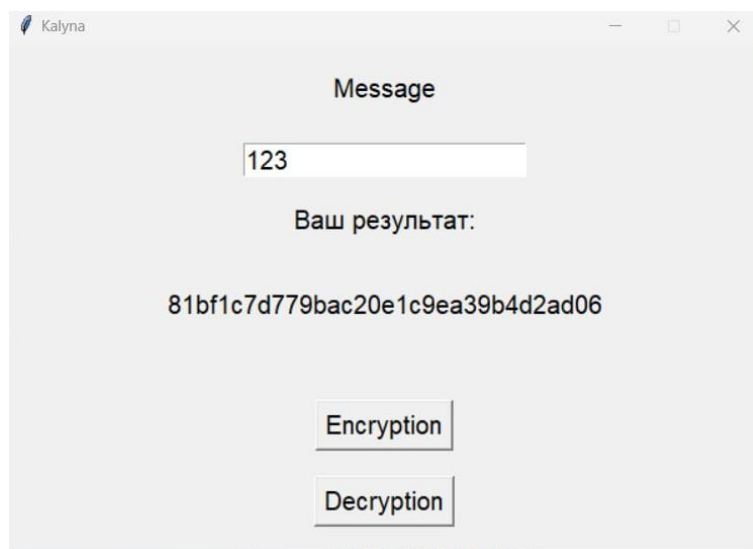


Рисунок 3.5. Найпростіший варіант використання кодової реалізації алгоритму шифрування «Калина» при шифруванні

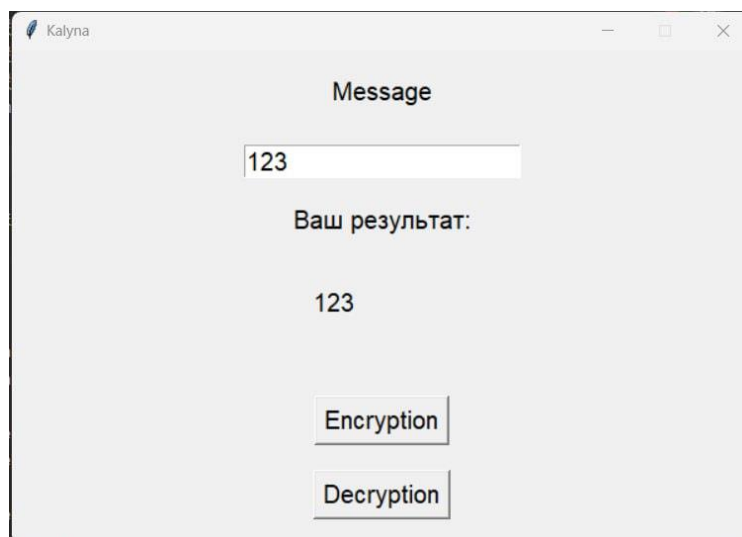


Рисунок 3.6. Найпростіший варіант використання кодової реалізації алгоритму шифрування «Калина» при дешифруванні

Функція отримує значення в 16-ковій СЧ, попередньо введений текст було закодовано у 16-ковий відповідник з можливістю розшифрування готового результату дешифрування

Також, у консоль виводиться покрокове виконання шифрування та дешифрування ведених в програму даних (рис. 3.7-3.8).

```
-----ШИФРУВАННЯ-----
KEY=[['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
PLAINTEXT=[['0x10', '0x18'], ['0x11', '0x19'], ['0x12', '0x1a'], ['0x13', '0x1b'], ['0x14', '0x1c'], ['0x15', '0x1d'], ['0x16', '0x1e'], ['0x17', '0x1f']]
(round[0].rkey):[['0x16', '0x86'], ['0x50', '0x5b'], ['0x5e', '0x77'], ['0x6b', '0xdc'], ['0x9b', '0xe0'], ['0x3a', '0x82'], ['0xb1', '0xa0'], ['0xe6', '0xf4']]
(round[0].add_rkey):[['0x26', '0x9e'], ['0x61', '0x74'], ['0x70', '0x91'], ['0x7e', '0xf7'], ['0xaf', '0xfc'], ['0x4f', '0x9f'], ['0xc7', '0xbe'], ['0xfd', '0x13']]
-----
(round[1].s_box):[['0x9a', '0x91'], ['0x2b', '0x4a'], ['0x1e', '0xcf'], ['0xac', '0x17'], ['0x76', '0x7c'], ['0xee', '0x98'], ['0x89', '0xdd'], ['0x1b', '0x3d']]
(round[1].s_row):[['0x9a', '0x91'], ['0x2b', '0x4a'], ['0x1e', '0xcf'], ['0xac', '0x17'], ['0x76', '0x7c'], ['0x98', '0xee'], ['0xdd', '0x89'], ['0x3d', '0x1b']]
(round[1].m_col):[['0x16', '0x25'], ['0xce', '0xb5'], ['0xde', '0x6'], ['0xe8', '0xf0'], ['0xd9', '0x42'], ['0x99', '0xd3'], ['0xf', '0xb3'], ['0x9e', '0x5']]
(round[1].rkey):[['0xe6', '0xf4'], ['0x86', '0x16'], ['0x5b', '0x50'], ['0x77', '0x5e'], ['0xdc', '0x6b'], ['0xe0', '0x9b'], ['0x82', '0x3a'], ['0xa0', '0xb1']]
(round[1].xor_key):[['0xf0', '0xd1'], ['0x48', '0xa3'], ['0x85', '0x56'], ['0x9f', '0xae'], ['0x5', '0x29'], ['0x79', '0x48'], ['0x8d', '0x89'], ['0x3e', '0xb4']]
-----
(round[2].s_box):[['0x81', '0xae'], ['0xd1', '0x4b'], ['0x3f', '0x55'], ['0xb2', '0x42'], ['0x75', '0xbf'], ['0x62', '0xd1'], ['0xed', '0xf7'], ['0xe1', '0x6f']]
(round[2].s_row):[['0x81', '0xae'], ['0xd1', '0x4b'], ['0x3f', '0x55'], ['0xb2', '0x42'], ['0xbf', '0x75'], ['0xd1', '0x62'], ['0xf7', '0xed'], ['0x6f', '0xe1']]
(round[2].m_col):[['0x32', '0x3b'], ['0xf1', '0x4d'], ['0x72', '0x13'], ['0xc7', '0x95'], ['0xe2', '0x8f'], ['0xd2', '0xb0'], ['0xe1', '0xe2'], ['0xc9', '0x8d']]
(round[2].rkey):[['0x7e', '0x8a'], ['0x70', '0xaa'], ['0x87', '0xa0'], ['0x6e', '0xa'], ['0xae', '0x7c'], ['0x49', '0x93'], ['0x84', '0xec'], ['0x76', '0x42']]
(round[2].xor_key):[['0x4c', '0xb1'], ['0x81', '0xe7'], ['0xf5', '0xb3'], ['0xa9', '0x9f'], ['0x4c', '0xf3'], ['0x9b', '0x2f'], ['0x65', '0xe'], ['0xbf', '0xcf']]
-----
(round[3].s_box):[['0x8f', '0xbe'], ['0xda', '0xa6'], ['0x86', '0x3a'], ['0x33', '0xb2'], ['0x8f', '0xed'], ['0x6a', '0x4d'], ['0x9c', '0x51'], ['0x7a', '0x39']]
(round[3].s_row):[['0x8f', '0xbe'], ['0xda', '0xa6'], ['0x86', '0x3a'], ['0x33', '0xb2'], ['0xed', '0x8f'], ['0x4d', '0x6a'], ['0x51', '0x9c'], ['0x39', '0x7a']]
(round[3].m_col):[['0x4', '0x13'], ['0x4e', '0xf2'], ['0x67', '0x41'], ['0x25', '0x97'], ['0x2', '0x77'], ['0xe9', '0x3d'], ['0x45', '0x45'], ['0xd3', '0x47']]
(round[3].rkey):[['0x76', '0x42'], ['0x8a', '0x7e'], ['0xaa', '0x70'], ['0xa0', '0x87'], ['0xa0', '0x6e'], ['0x7c', '0xae'], ['0x93', '0x84'], ['0xec', '0x84']]
(round[3].xor_key):[['0x72', '0x51'], ['0x4c', '0x8c'], ['0xcd', '0x33'], ['0x85', '0x10'], ['0x8', '0x19'], ['0x95', '0x93'], ['0xd6', '0xc'], ['0x3f', '0xc3']]
-----
(round[4].s_box):[['0x24', '0xe7'], ['0x9a', '0xe'], ['0x64', '0x2c'], ['0x8f', '0x22'], ['0x71', '0xe0'], ['0x47', '0xfe'], ['0x75', '0x9f'], ['0xd', '0xf']]
(round[4].s_row):[['0x24', '0xe7'], ['0x9a', '0xe'], ['0x64', '0x2c'], ['0x8f', '0x22'], ['0xe0', '0x71'], ['0xfe', '0x47'], ['0x9f', '0x75'], ['0xf', '0xd']]
(round[4].m_col):[['0x73', '0x7a'], ['0xec', '0x44'], ['0x52', '0x21'], ['0x1d', '0x24'], ['0xa9', '0x56'], ['0xba', '0xfe'], ['0x96', '0x8e'], ['0x77', '0x15']]
(round[4].rkey):[['0x45', '0x3e'], ['0xce', '0x72'], ['0xd4', '0x76'], ['0xc5', '0x82'], ['0x1e', '0xf'], ['0x91', '0xb'], ['0x40', '0xd9'], ['0xf5', '0xfe']]
(round[4].xor_key):[['0x36', '0x44'], ['0x22', '0x36'], ['0x86', '0x57'], ['0xd8', '0xa6'], ['0xb7', '0x59'], ['0x2b', '0xf5'], ['0xb9', '0xdb'], ['0x82', '0xeb']]
-----
```

Рисунок 3.7. Опис покрокового виконання алгоритму шифрування у консольному виводі

```
-----ДЕШИФРУВАННЯ-----
KEY=[['0x8', '0x28'], ['0xc', '0x2c'], ['0x10', '0x30'], ['0x4', '0x34'], ['0x18', '0x38'], ['0x1c', '0x3c'], ['0x20', '0x0'], ['0x24', '0x4']]
CHIPHERTEXT=[['0x81', '0xe1'], ['0xbf', '0xc9'], ['0x1c', '0xea'], ['0x7d', '0x39'], ['0x77', '0xb4'], ['0x9b', '0xd2'], ['0xac', '0xad'], ['0x20', '0x6']]
10
(round[10].rkey):[['0x57', '0xf6'], ['0x26', '0xb'], ['0xb1', '0xf3'], ['0xa8', '0xd5'], ['0x94', '0xe8'], ['0xdb', '0xd7'], ['0xc4', '0x48'], ['0x18', '0x61']]
(round[10].sub_rkey):[['0xd6', '0xeb'], ['0x66', '0x41'], ['0x6a', '0xf6'], ['0xd4', '0x63'], ['0xe2', '0xcb'], ['0xbf', '0xfa'], ['0xe7', '0x9a'], ['0xf7', '0xa4']]
(round[10].m_col):[['0x5c', '0x18'], ['0xe6', '0x8b'], ['0xea', '0xb'], ['0xf8', '0xb'], ['0x69', '0x1b'], ['0x96', '0xb0'], ['0xe7', '0x8e'], ['0x9e', '0x87']]
(round[10].s_row):[['0x5c', '0x18'], ['0xe6', '0x8b'], ['0xea', '0xb'], ['0xf8', '0xe4'], ['0x1b', '0x69'], ['0xb0', '0x96'], ['0x8e', '0xe7'], ['0x87', '0x9e']]
(round[10].s_box):[['0x84', '0x86'], ['0xad', '0xb0'], ['0xcb', '0x8'], ['0xa2', '0xaa'], ['0x8d', '0x92'], ['0x9e', '0x64'], ['0xab', '0xea'], ['0xd6', '0x26']]
9
(round[9].rkey):[['0xb8', '0xd8'], ['0xaa', '0xa'], ['0x87', '0x98'], ['0x9a', '0x72'], ['0x20', '0xe2'], ['0x86', '0x5c'], ['0xa6', '0xd2'], ['0xd', '0xb0']]
(round[9].xor_key):[['0x3c', '0x5e'], ['0x7', '0xba'], ['0x4c', '0x90'], ['0x38', '0xd8'], ['0xad', '0x70'], ['0x18', '0x38'], ['0xd', '0xb0'], ['0xb', '0x96']]
(round[9].m_col):[['0x4d', '0x18'], ['0x10', '0x29'], ['0xc7', '0xda'], ['0x44', '0xb7'], ['0xb7', '0x28'], ['0x2b', '0x35'], ['0x5e', '0xf2'], ['0x5a', '0xc9']]
(round[9].s_row):[['0x4d', '0x18'], ['0x10', '0x29'], ['0xc7', '0xda'], ['0x44', '0xb7'], ['0x28', '0xb7'], ['0x35', '0x2b'], ['0xf2', '0x5e'], ['0x9c', '0x5a']]
(round[9].s_box):[['0x15', '0x86'], ['0x3d', '0xfe'], ['0x88', '0x6e'], ['0x61', '0x1a'], ['0x73', '0x63'], ['0xd9', '0x61'], ['0xba', '0x7e'], ['0xd', '0x5e']]
8
(round[8].rkey):[['0xa', '0xaa'], ['0x98', '0x87'], ['0x72', '0x9a'], ['0xe2', '0x20'], ['0x5c', '0x86'], ['0xd2', '0xa6'], ['0xb0', '0xd6'], ['0xb8', '0xd8']]
(round[8].xor_key):[['0x1f', '0xf0'], ['0x2c', '0x79'], ['0xfa', '0xf4'], ['0x83', '0x3a'], ['0x2f', '0xe5'], ['0xb', '0x13'], ['0xd5', '0x3d']]
(round[8].m_col):[['0x9b', '0xf2'], ['0xd3', '0x48'], ['0xea', '0xf'], ['0xbe', '0x3'], ['0x82', '0x2b'], ['0x49', '0x2c'], ['0xba', '0x3e'], ['0x9e', '0x6b']]
(round[8].s_row):[['0x9b', '0xf2'], ['0xd3', '0x48'], ['0xea', '0xf'], ['0xbe', '0x3'], ['0x2b', '0x82'], ['0x2c', '0x49'], ['0x3e', '0xba'], ['0x6b', '0x9e']]
(round[8].s_box):[['0x80', '0x44'], ['0xca', '0x62'], ['0xcb', '0x95'], ['0x65', '0x11'], ['0x57', '0xcf'], ['0xe1', '0x59'], ['0x9c', '0x8'], ['0xe2', '0x26']]
7
(round[7].rkey):[['0x62', '0xb1'], ['0x51', '0x8c'], ['0x5f', '0x77'], ['0x66', '0xee'], ['0x32', '0x22'], ['0x5', '0x79'], ['0x60', '0x0'], ['0xc4', '0xc4']]
(round[7].xor_key):[['0xe2', '0xf0'], ['0x9b', '0xee'], ['0x94', '0xf2'], ['0x3', '0xff'], ['0x65', '0xed'], ['0xe4', '0x20'], ['0xfc', '0x8'], ['0x26', '0xe2']]
(round[7].m_col):[['0x2f', '0xe'], ['0x7b', '0x9'], ['0xfb', '0xc4'], ['0x26', '0x9d'], ['0x2e', '0x1d'], ['0x23', '0xff'], ['0x11', '0x71'], ['0x1c', '0xc9']]
(round[7].s_row):[['0x2f', '0xe'], ['0x7b', '0x9'], ['0xfb', '0xc4'], ['0x26', '0x9d'], ['0x1d', '0x2e'], ['0xff', '0x23'], ['0x71', '0x11'], ['0xc9', '0x1c']]
(round[7].s_box):[['0xc0', '0x27'], ['0x5a', '0x96'], ['0xb5', '0x32'], ['0xa', '0xd5'], ['0x12', '0x50'], ['0xaa', '0xba'], ['0x1a', '0x44'], ['0xfb', '0xca']]
6
(round[6].rkey):[['0x8c', '0x51'], ['0x77', '0x5f'], ['0xee', '0x66'], ['0x22', '0x32'], ['0x79', '0x5'], ['0x0', '0x60'], ['0xc4', '0xc4'], ['0x62', '0xb1']]
(round[6].xor_key):[['0x4c', '0x76'], ['0x2d', '0xc9'], ['0x5b', '0x54'], ['0x28', '0xe7'], ['0x6b', '0x55'], ['0xaa', '0xda'], ['0xde', '0x80'], ['0x99', '0x7b']]
-----
```

Рисунок 3.8. Опис покрокового виконання алгоритму дешифрування у консольному виводі

Було вирішено не виводити всі етапи шифрування та дешифрування у віконний додаток, оскільки це було б повним повторенням написаної офіційної документації[1] даного алгоритму. Тобто користувач бібліотеки має змогу, за бажанням, вивчити та розібрати кожен крок алгоритму. Перед остаточним

написанням функції для шифрування та дешифрування, попередньо було використано тестування за наданим, у документації, покроковим прикладом. Результати повністю збіглися, тому алгоритм вважається робочим та готовим до використання.

3.2 Тестування

Перш ніж використовувати фінальну функцію як бібліотеку для сторонніх програмних рішень, створене кодове рішення було перевірено на прикладах з офіційної документації (рис. 3.9, 3.11, 3.13) та перевірено програмно (рис. 3.10, 3.12, 3.14).

```
Round keys for Kalyna-128/128

round[0].rkey:
16505E6B9B3AB1E6865B77DCE082A0F4

round[1].rkey:
E6865B77DCE082A0F416505E6B9B3AB1

round[2].rkey:
7E70876EAE4984768AAA00A7C93EC42

round[3].rkey:
768AAA00A7C93EC427E70876EAE4984

round[4].rkey:
45CED4C51E9140F53E7276820F0BD9FE

round[5].rkey:
F53E7276820F0BD9FE45CED4C51E9140

round[6].rkey:
8C77EE227900C462515F66320560C4B1

round[7].rkey:
62515F66320560C4B18C77EE227900C4

round[8].rkey:
0A9872E25CD2B0B8AA879A2086A66DD8

round[9].rkey:
B8AA879A2086A66DD80A9872E25CD2B0

round[10].rkey:
5726B1A894DBC418F60BF3D5E8D74861
```

Рисунок 3.9. Раундові ключі в офіційній документації

```
resultRoundKeysArr[0]=['0x16', '0x86', ['0x50', '0x5b'], ['0x5e', '0x77'], ['0x6b', '0xdc'], ['0x9b', '0xe0'], ['0x3a', '0x82'], ['0xb1', '0xa0'], ['0xe6', '0xf4']]
resultRoundKeysArr[1]=['0xe6', '0xf4'], ['0x86', '0x16'], ['0x5b', '0x50'], ['0x77', '0x5e'], ['0xdc', '0x6b'], ['0xe0', '0x9b'], ['0x82', '0x3a'], ['0xa0', '0xb1']]
resultRoundKeysArr[2]=['0x7e', '0x8a'], ['0x70', '0xaa'], ['0x87', '0xa0'], ['0x6e', '0xa'], ['0xae', '0x7c'], ['0x49', '0x93'], ['0x84', '0xec'], ['0x76', '0x42']]
resultRoundKeysArr[3]=['0x76', '0x42'], ['0x8a', '0x7e'], ['0xaa', '0x70'], ['0xa0', '0x87'], ['0xa', '0x6e'], ['0x7c', '0xae'], ['0x93', '0x49'], ['0xec', '0x84']]
resultRoundKeysArr[4]=['0x45', '0x3e'], ['0xce', '0x72'], ['0xd4', '0x76'], ['0xc5', '0x82'], ['0x1e', '0xf'], ['0x91', '0xb'], ['0x40', '0xd9'], ['0xf5', '0xfe']]
resultRoundKeysArr[5]=['0xf5', '0xfe'], ['0x3e', '0x45'], ['0x72', '0xce'], ['0x76', '0xd4'], ['0x82', '0xc5'], ['0xf', '0x1e'], ['0xb', '0x91'], ['0xd9', '0x40']]
resultRoundKeysArr[6]=['0x8c', '0x51'], ['0x77', '0x5f'], ['0xee', '0x66'], ['0x22', '0x32'], ['0x79', '0x5'], ['0x0', '0x60'], ['0xc4', '0xc4'], ['0x62', '0xb1']]
resultRoundKeysArr[7]=['0x62', '0xb1'], ['0x51', '0x8c'], ['0x5f', '0x77'], ['0x66', '0xee'], ['0x32', '0x22'], ['0x5', '0x79'], ['0x60', '0x0'], ['0xc4', '0xc4']]
resultRoundKeysArr[8]=['0xa', '0xaa'], ['0x98', '0x87'], ['0x72', '0x9a'], ['0xe2', '0x20'], ['0x5c', '0x86'], ['0xd2', '0xa6'], ['0xb0', '0x6d'], ['0xb8', '0xd8']]
resultRoundKeysArr[9]=['0xb8', '0xd8'], ['0xaa', '0xa'], ['0x87', '0x98'], ['0x9a', '0x72'], ['0x20', '0xe2'], ['0x86', '0x5c'], ['0xa6', '0xd2'], ['0x6d', '0xb0']]
resultRoundKeysArr[10]=['0x57', '0xf6'], ['0x26', '0xb'], ['0xb1', '0xf3'], ['0xa8', '0xd5'], ['0x94', '0xe8'], ['0xdb', '0xd7'], ['0xc4', '0x48'], ['0x18', '0x61']]
```

Рисунок 3.10. Пораховані раундові ключі в програмі

```

state[0].k0:
000102030405060708090A0B0C0D0E0F

state[0].k1:
000102030405060708090A0B0C0D0E0F

state[0].add_rkey:
050102030405060708090A0B0C0D0E0F

state[0].s_box:
75BB9A4D6BCB452A713ADF831790511F

state[0].s_row:
75BB9A4D1790511F713ADF836BCB452A

state[0].m_col:
62C97C6E6ABF4133ED5131D624C7C182

state[0].xor_rkey:
62C87E6D6EBA4734E5583BDD28CACF8D

state[0].s_box:
FC4F5E9CC3232E40D98141FC1FD382BF

state[0].s_row:
FC4F5E9C1FD382BFD98141FCC3232E40

state[0].m_col:
53E85C8F02C0CA94B7578DD19C8B8A35

state[0].add_rkey:
53E95E9206C5D09BBF6097DCA8989844

state[0].s_box:
5A04E6B6C846B1D26E724A5C50B28E5

state[0].s_row:
5A04E6B6C50B28E526E724A56C846B1D

state[0].m_col:
862F1F653B775BA1D05CBC2F38E2D87D

KT:
862F1F653B775BA1D05CBC2F38E2D87D

```

Рисунок 3.11. Сформований допоміжний ключ в офіційній документації

```

-----РОЗГОРТАННЯ КЛЮЧІВ-----
K0 (Початковий ключ): [['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
Nb = 2, Nk = 2, rounds: 10
messageToHex[['0x10', '0x18'], ['0x11', '0x19'], ['0x12', '0x1a'], ['0x13', '0x1b'], ['0x14', '0x1c'], ['0x15', '0x1d'], ['0x16', '0x1e'], ['0x17', '0x1f']]
StateMatrix[['0x5', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0']]
(state[0])(Nb+Nk+1)(Початкова матриця стану): [['0x5', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0'], ['0x0', '0x0']]
(state[0].k0): [['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
(state[0].k1): [['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
(state[0].add_rkey): [['0x5', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
K0PlusKEY[['0x5', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
ROUND 1:
(state[0].s_box): [['0x75', '0x71'], ['0xbb', '0x3a'], ['0x9a', '0xdf'], ['0x4d', '0xb3'], ['0x6b', '0x17'], ['0xcb', '0x90'], ['0x45', '0x51'], ['0x2a', '0x1f']]
(state[0].s_row): [['0x75', '0x71'], ['0xbb', '0x3a'], ['0x9a', '0xdf'], ['0x4d', '0xb3'], ['0x17', '0x6b'], ['0x90', '0xcb'], ['0x51', '0x45'], ['0x1f', '0x2a']]
(state[0].m_col): [['0x62', '0xed'], ['0xc9', '0x51'], ['0x7c', '0x31'], ['0x6e', '0xd6'], ['0x6a', '0x24'], ['0xbf', '0xc7'], ['0x41', '0xc1'], ['0x33', '0x82']]
(state[0].xor_key): [['0x62', '0xe5'], ['0xc8', '0x58'], ['0x7e', '0x3b'], ['0x6d', '0xdd'], ['0x6e', '0x28'], ['0xba', '0xca'], ['0x47', '0xcf'], ['0x34', '0x8d']]
ROUND 2:
(state[0].s_box): [['0xfc', '0xd9'], ['0x4f', '0x81'], ['0x5e', '0x41'], ['0x9c', '0xfc'], ['0xc3', '0x1f'], ['0x23', '0xd3'], ['0x2e', '0x82'], ['0x40', '0xbf']]
(state[0].s_row): [['0xfc', '0xd9'], ['0x4f', '0x81'], ['0x5e', '0x41'], ['0x9c', '0xfc'], ['0x1f', '0xc3'], ['0xd3', '0x23'], ['0x82', '0x2e'], ['0xbf', '0x40']]
(state[0].m_col): [['0x53', '0xb7'], ['0xe8', '0x57'], ['0x5c', '0x8d'], ['0x8f', '0xd1'], ['0x2', '0x9c'], ['0xc0', '0x8b'], ['0xca', '0x8a'], ['0x94', '0x35']]
(state[0].add_rkey): [['0x53', '0xbf'], ['0xe8', '0x60'], ['0x5e', '0x97'], ['0x92', '0xdc'], ['0x6', '0xa8'], ['0xc5', '0x98'], ['0xd0', '0x98'], ['0x9b', '0x44']]
ROUND 3:
(state[0].s_box): [['0x5a', '0x26'], ['0x4', '0xe7'], ['0xe6', '0x24'], ['0xb6', '0xa5'], ['0x6c', '0xc5'], ['0x84', '0xb'], ['0x6b', '0x28'], ['0x1d', '0xe5']]
(state[0].s_row): [['0x5a', '0x26'], ['0x4', '0xe7'], ['0xe6', '0x24'], ['0xb6', '0xa5'], ['0xc5', '0x6c'], ['0xb', '0x84'], ['0x28', '0x6b'], ['0xe5', '0x1d']]
(state[0].m_col): [['0x86', '0xd0'], ['0x2f', '0x5c'], ['0x1f', '0xb'], ['0x65', '0x2f'], ['0x3b', '0x38'], ['0x77', '0xe2'], ['0x5b', '0xd8'], ['0xa1', '0x7d']]
Kt: [['0x86', '0xd0'], ['0x2f', '0x5c'], ['0x1f', '0xb'], ['0x65', '0x2f'], ['0x3b', '0x38'], ['0x77', '0xe2'], ['0x5b', '0xd8'], ['0xa1', '0x7d']]
Round keys for Kalyna-128/128 (even indexes)
(state[0]): [['0x86', '0xd0'], ['0x2f', '0x5c'], ['0x1f', '0xb'], ['0x65', '0x2f'], ['0x3b', '0x38'], ['0x77', '0xe2'], ['0x5b', '0xd8'], ['0xa1', '0x7d']]
(state[0].Kt): [['0x86', '0xd0'], ['0x2f', '0x5c'], ['0x1f', '0xb'], ['0x65', '0x2f'], ['0x3b', '0x38'], ['0x77', '0xe2'], ['0x5b', '0xd8'], ['0xa1', '0x7d']]
(state[0].id): [['0x0', '0x8'], ['0x1', '0x9'], ['0x2', '0xa'], ['0x3', '0xb'], ['0x4', '0xc'], ['0x5', '0xd'], ['0x6', '0xe'], ['0x7', '0xf']]
*****

```

Рисунок 3.12. Сформований допоміжний ключ в програмі

		Луцевич М. О.			Арк.
		Байлюк Є М.			
Змн.	Арк.	№ докум.	Підпис	Дата	44

```

round[9].m_col:
375DCC2C800C45F61DD9C471E522E0A6

round[9].r_key:
B8AA879A2086A66DD80A9872E25CD2B0

round[9].xor_rkey:
8FF74BB6A08AE39BC5D35C03077E3216

round[10].s_box:
FF661201789C581D0374D34D5983C453

round[10].s_row:
FF6612015983C4530374D34D789C581D

round[10].m_col:
2A996BD4E2BFE707EBBDF763CBFA64A5

round[10].add_rkey:
81BF1C7D779BAC20E1C9EA39B4D2AD06

CIPHERTEXT:
81BF1C7D779BAC20E1C9EA39B4D2AD06

```

Рисунок 3.13. Сформований криптотекст в офіційній документації

```

(round[9].s_box):[['0x97', '0x2c'], ['0x3', '0x7c'], ['0x80', '0xf1'], ['0xc8', '0x48'], ['0xb3', '0x0'], ['0x9e', '0x55'], ['0x90', '0xe5'], ['0x9a', '0xe5']]
(round[9].s_row):[['0x97', '0x2c'], ['0x3', '0x7c'], ['0x80', '0xf1'], ['0xc8', '0x48'], ['0xb3', '0x55'], ['0x9e', '0xe5'], ['0x90', '0xe5'], ['0x9a', '0xe5']]
(round[9].m_col):[['0x37', '0x1d'], ['0x5d', '0xd9'], ['0xcc', '0xc4'], ['0x2c', '0x71'], ['0x80', '0xe5'], ['0xc', '0x22'], ['0x45', '0xe0'], ['0xf6', '0xa6']]
(round[9].rkey):[['0xb8', '0xd8'], ['0xaa', '0xa'], ['0x87', '0x98'], ['0x9a', '0x72'], ['0x20', '0xe2'], ['0x86', '0x5c'], ['0xa6', '0xd2'], ['0x6d', '0xb0']]
(round[9].xor_key):[['0x8f', '0xc5'], ['0xf7', '0xd3'], ['0x4b', '0x5c'], ['0xb6', '0x3'], ['0xa0', '0x7'], ['0x8a', '0x7e'], ['0xe3', '0x32'], ['0x9b', '0x16']]
-----
(round[9].s_box):[['0xff', '0x3'], ['0x66', '0x74'], ['0x12', '0xd3'], ['0x1', '0x4d'], ['0x78', '0x59'], ['0x9c', '0x83'], ['0x58', '0xc4'], ['0x1d', '0x53']]
(round[9].s_row):[['0xff', '0x3'], ['0x66', '0x74'], ['0x12', '0xd3'], ['0x1', '0x4d'], ['0x59', '0x78'], ['0x83', '0x9c'], ['0xc4', '0x58'], ['0x53', '0x1d']]
(round[9].m_col):[['0x2a', '0xeb'], ['0x99', '0xbd'], ['0x6b', '0xf7'], ['0xd4', '0x63'], ['0xe2', '0xcb'], ['0xbf', '0xfa'], ['0xe7', '0x64'], ['0x7', '0xa5']]
(round[9].add_rkey):[['0x81', '0xe1'], ['0xbf', '0xc9'], ['0x1c', '0xea'], ['0x7d', '0x39'], ['0x77', '0xb4'], ['0x9b', '0xd2'], ['0xac', '0xad'], ['0x20', '0x6']]
CIPHERTEXT= [['0x81', '0xe1'], ['0xbf', '0xc9'], ['0x1c', '0xea'], ['0x7d', '0x39'], ['0x77', '0xb4'], ['0x9b', '0xd2'], ['0xac', '0xad'], ['0x20', '0x6']]

```

Рисунок 3.14. Сформований криптотекст в програмі

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

В даному розділі було детально розглянуто основні функції, роботу бібліотеки. Підготовлено мінімально технічну документацію, відповідно основним функціям бібліотеки. Розглянуто найпростіший приклад використання з додатковою конвертацією тексту у 16-кове значення; перевірено роботу алгоритму за допомогою прикладів з офіційної документації. Розглянуто надану кінцевому користувачу бібліотеки можливість покрокової перевірки роботи алгоритму, як для розуміння роботи вбудованої функції, так і для самоперевірки в разі самостійного вивчення алгоритму шифрування «Калина».

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

ВИСНОВКИ

В першому розділі було описано структуру алгоритму шифрування та дешифрування, представлений опис алгоритму для шифрування та розшифрування, проведено аналіз безпеки та стійкості алгоритму, а також порівняння з іншими алгоритмами.

В другому розділі було детально розглянуто основні дії формування ключів, шифрування та дешифрування — формування початкової матриці, SubBytes, ShiftRows, MixColumns, додавання та віднімання по модулю 2^{64} , Inv SubBytes, Inv ShiftRows, Inv MixColumns. Обґрунтовано вибір мови програмування та середовища розробки. Наведено деталі розробки програмного коду.

В третьому розділі було детально розглянуто основні функції, роботу бібліотеки. Підготовлено мінімально технічну документацію, відповідно основним функціям бібліотеки. Розглянути найпростіший приклад використання з додатковою конвертацією тексту у 16-кове значення; перевірено роботу алгоритму за допомогою прикладів з офіційної документації. Розглянуто надану кінцевому користувачу бібліотеки можливість покрокової перевірки роботи алгоритму, як для розуміння роботи вбудованої функції, так і для самоперевірки в разі самотійного вивчення алгоритму шифрування «Калина».

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A New Encryption Standard of Ukraine: The Kalyna Block Cipher [Електронний ресурс] / [R. Oliynykov, I. Gorbenko, O. Kazymyrov та ін.]. – 2014. – Режим доступу до ресурсу: <https://eprint.iacr.org/2015/650.pdf>.
2. ЛАБОРАТОРНА РОБОТА № 5. ДОСЛІДЖЕННЯ ОСНОВНИХ ОПЕРАЦІЙ ШИФРУ «КАЛИНА» У ПРОЦЕСІ ФОРМУВАННЯ ДОПОМІЖНОГО КЛЮЧА [Електронний ресурс] – Режим доступу до ресурсу:
https://learn.ztu.edu.ua/pluginfile.php/244887/mod_resource/content/1/Лаб5.pdf.
3. Щур Н. О. Національний стандарт шифрування ДСТУ 7624:2014 («Калина») [Електронний ресурс] / Наталя Олександрівна Щур. – 2022. – Режим доступу до ресурсу:
https://learn.ztu.edu.ua/pluginfile.php/269838/mod_resource/content/1/Лекція5.pdf.
4. МЕТОДИ СИНТЕЗА І АНАЛІЗУ СИМЕТРИЧНИХ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ [Електронний ресурс] / І. Д.ГОРБЕНКО, О. В. КАЗИМИРОВ, О. О. КУЗНЄЦОВ, О. О. ДИРДА // Харківський національний університет радіоелектроніки. – 2015. – Режим доступу до ресурсу: https://nure.ua/wp-content/uploads/2015/Scientific_editions/181/3.pdf.
5. Rahul J. What is XOR in Python? [Електронний ресурс] / Janghu Rahul – Режим доступу до ресурсу: <https://www.scaler.com/topics/xor-in-python/>.
6. Circular shift of a bit in python (equivalent of Fortran's ISHFTC) [Електронний ресурс] – Режим доступу до ресурсу:
<https://stackoverflow.com/questions/63759207/circular-shift-of-a-bit-in-python-equivalent-of-fortrans-ishftc>.

7. Як реалізувати додавання послідовності бітів по модулю 2^{32} ? [Електронний ресурс] – Режим доступу до ресурсу: <https://qna.habr.com/q/521994>.
8. How do I split a 2 character string in python? [Електронний ресурс] – Режим доступу до ресурсу: <https://qna.habr.com/q/601303>.
9. Lists, tuples, and dictionaries [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/python/tutorial/3.1.php>.
10. Байлюк Є. М. АНАЛІЗ ТА ПОРІВНЯННЯ АЛГОРИТМУ СИМЕТРИЧНОГО БЛОКОВОГО ПЕРЕТВОРЕННЯ «КАЛИНА» (ДСТУ 7624:2014) З МІЖНАРОДНИМ СТАНДАРТОМ ШИФРУВАННЯ ДАНИХ AES [Електронний ресурс] / Є. М. Байлюк, О. А. Покотило. – 2018. – Режим доступу до ресурсу: <https://conf.ztu.edu.ua/wp-content/uploads/2018/05/3-1.pdf>.
11. Single Key Recovery Attacks on 9-round Kalyna-128/256 and Kalyna-256/512 [Електронний ресурс] / K. S.Somitra, C. Donghun, G. Moona, G. Aarushi – Режим доступу до ресурсу: <https://eprint.iacr.org/2015/1227.pdf>.
12. Kazymyrov O. V. ALGEBRAIC PROPERTIES OF SYMMETRIC BLOCK CIPHER “KALYNA” KEY SCHEDULE [Електронний ресурс] / О. V. Kazymyrov, R. V. Oliynykov // Харківський національний університет радіоелектроніки. – 2010. – Режим доступу до ресурсу: <https://okazymyrov.github.io/assets/attachments/articles/2010/9691823db6acee5fb932ecbd053397.pdf>.
13. Li L. Improved Meet-in-the-Middle Attacks on Reduced-Round Kalyna-128/256 and Kalyna-256/512 [Електронний ресурс] / L. Li, W. Wenling // Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China – Режим доступу до ресурсу: <https://eprint.iacr.org/2016/722.pdf>.
14. Python 3.11.4 documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/uk/3/>.

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

15. ГОСТ 28147-89 [Электронный ресурс] – Режим доступа до ресурсу:
https://uk.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2_28147-89.
16. Kalyna (cipher) [Электронный ресурс] – Режим доступа до ресурсу:
[https://en.wikipedia.org/wiki/Kalyna_\(cipher\)](https://en.wikipedia.org/wiki/Kalyna_(cipher)).

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

ДОДАТКИ

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Функція додавання по модулю 2⁶⁴

```

def FullArr(arrA):
    # print(f'arrA[0][0]={arrA[0][0]}')
    # print(f'len(arrA[0])={len(arrA)}')
    # print(f'CHECK={len(arrA[0])}')
    for i in range(len(arrA)): # 4
        # print(f'flag arrA={arrA}')
        # for j in range(len(arrA[0])):#2
        arrA[i] = bin(int(str(arrA[i]), 16))
        # bin(int(str(arrA[i]), 16)) -->
        # arrA[i] = bin(int(str(arrA[i]), 16))
        # print(f'arrA[i] FLAG={arrA[i][j]}')
        if (arrA[i][1] == 'b'):
            arrA[i] = re.findall(r'[0-1]', arrA[i])[1:9]
            # print(f'BarrA[i][j]={arrA[i][j]}')
        else:
            arrA[i] = re.findall(r'[0-1]', arrA[i])
            # print(f'arrA[i][j]={arrA[i][j]}')
        if (len(arrA[i]) <= 7):
            # print(f'arrA[i][j] In last if={arrA[i][j]}')
            for k in range(8 - len(arrA[i])):
                arrA[i].insert(0, '0')
                # print(f'{k}, {len(arrA[i][j])}, {arrA[i][j]}')

            # if (len(arrA[i][j]) <= 7):
            #     for j in range(8 - len(arrA[i][j])):
            #         print(f'type(arrA[i][j])={type(arrA[i][j])}')
    # print(f'arrA={arrA}')
    return arrA

def InvertArr(arr): # take: hex array, get: invert hex array
    InvertArr = [[0x00] * 2 for i in range(8)]
    InvertArr = arr[::-1]
    for i in range(len(arr)): # 2
        InvertArr[i] = InvertArr[i][::-1]
    # print(InvertArr)
    return InvertArr

def FirstFinalAddition(arrA, arrB):
    # print('FirstFinalAddition')
    # print(f'FirstFinalAddition ArrA={arrA}')
    # print(f'FirstFinalAddition ArrB={arrB}')
    result = [9] * (len(arrA + arrB) * 4)
    count = 0
    pointer = (len(arrA + arrB) * 4) - 1
    for i in reversed(range(len(arrA))):
        # print(f'{i}, {arrA[i]}')

```

```

# print(f'{i}, {arrB[i]}')
for j in reversed(range(8)):
    arrA[i][j] = int(arrA[i][j])
    arrB[i][j] = int(arrB[i][j])
    if (arrA[i][j] + arrB[i][j] == 0):
        if count == 1:
            result[pointer] = int(1)
            count = 0
            pointer -= 1
        elif count == 0:
            result[pointer] = int(0)
            count = 0
            pointer -= 1
    elif (arrA[i][j] + arrB[i][j] == 1):
        if count == 1:
            result[pointer] = int(0)
            count = 1
            pointer -= 1
        elif count == 0:
            result[pointer] = int(1)
            count = 0
            pointer -= 1
    elif (arrA[i][j] + arrB[i][j] == 2):
        if count == 1:
            result[pointer] = int(1)
            count = 1
            pointer -= 1
        elif count == 0:
            result[pointer] = int(0)
            count = 1
            pointer -= 1
        # print(f'{arrA[i][j]}+{arrB[i][j]} =
{result[pointer+1]}, count = {count}')
    # print(f'result: {result}')
    resultFinal = [0x00] * int(len(result) / 8)
    # print(f'resultFinal={resultFinal}')
    k = 0
    # print(len(result), len(resultFinal))
    for i in reversed(range(len(result))):
        # print(f'i={i}')
        if i % 8 == 0:
            # print(str(result[i]) + str(result[i + 1]) +
str(result[i + 2]) + str(result[i +
3])+str(result[i+4])+str(result[i+5])+str(result[i+6])+str(result
[i+7]))
            resultFinal[k] = str(result[i]) + str(result[i + 1])
+ str(result[i + 2]) + str(result[i + 3]) + str(
            result[i + 4]) + str(result[i + 5]) +
str(result[i + 6]) + str(result[i + 7])
            # print(resultFinal[k])
            resultFinal[k] = hex(int(resultFinal[k], 2))
            k += 1
    # print(resultFinal)

```

```

return resultFinal[::-1]

def SecondFinalAddition(arrC, arrD):
    # print('FirstFinalAddition')
    # print(f'FirstFinalAddition ArrC={arrC}')
    # print(f'FirstFinalAddition ArrD={arrD}')
    result = [9] * (len(arrC + arrD) * 4)
    count = 0
    pointer = (len(arrC + arrD) * 4) - 1
    for i in reversed(range(len(arrC))):
        # print(f'{i}, {arrC[i]}')
        # print(f'{i}, {arrD[i]}')
        for j in reversed(range(8)):
            arrC[i][j] = int(arrC[i][j])
            arrD[i][j] = int(arrD[i][j])
            if (arrC[i][j] + arrD[i][j] == 0):
                if count == 1:
                    result[pointer] = int(1)
                    count = 0
                    pointer -= 1
                elif count == 0:
                    result[pointer] = int(0)
                    count = 0
                    pointer -= 1
            elif (arrC[i][j] + arrD[i][j] == 1):
                if count == 1:
                    result[pointer] = int(0)
                    count = 1
                    pointer -= 1
                elif count == 0:
                    result[pointer] = int(1)
                    count = 0
                    pointer -= 1
            elif (arrC[i][j] + arrD[i][j] == 2):
                if count == 1:
                    result[pointer] = int(1)
                    count = 1
                    pointer -= 1
                elif count == 0:
                    result[pointer] = int(0)
                    count = 1
                    pointer -= 1
                    # print(f'{arrC[i][j]}+{arrD[i][j]} =
{result[pointer+1]}, count = {count}')
                # print(f'result: {result}')
            resultFinal = [0x00] * int(len(result) / 8)
            # print(f'resultFinal={resultFinal}')
            k = 0
            # print(len(result), len(resultFinal))
        for i in reversed(range(len(result))):
            # print(f'i={i}')
            if i % 8 == 0:

```

```

        # print(str(result[i]) + str(result[i + 1]) +
str(result[i + 2]) + str(result[i +
3]))+str(result[i+4])+str(result[i+5])+str(result[i+6])+str(result
[i+7]))
        resultFinal[k] = str(result[i]) + str(result[i + 1])
+ str(result[i + 2]) + str(result[i + 3]) + str(
        result[i + 4]) + str(result[i + 5]) +
str(result[i + 6]) + str(result[i + 7])
        # print(resultFinal[k])
        resultFinal[k] = hex(int(resultFinal[k], 2))
        k += 1
# print(resultFinal)
return resultFinal[::-1]

def Add2in64(a, b):
    # print(f'a={a}')
    # print(f'b={b}')

    arrAinvert = [0x00] * (int(len(a) / 8) * 8)
    arrBinvert = [0x00] * (int(len(b) / 8) * 8)
    # print('-----')
    arrAinvert = InvertArr(a) # invert array a #
    # print(f'arrAinvert={arrAinvert}')
    arrBinvert = InvertArr(b) # invert array b #
    # print(f'arrBinvert={arrBinvert}')
    # print('-----')

    # print(f'Start invert array A in hex: {arrAinvert}')
    # print(f'Start invert array B in hex: {arrBinvert}')

    arrA = [[0x00] * int(len(arrAinvert[0]) / 2) for i in
range(8)]
    # print(f'arrA={arrA}')
    arrB = [[0x00] * int(len(arrBinvert[0]) / 2) for i in
range(8)]
    # print(f'arrB={arrB}')
    arrC = [[0x00] * int(len(arrAinvert[0]) / 2) for i in
range(8)]
    # print(f'arrC={arrC}')
    arrD = [[0x00] * int(len(arrBinvert[0]) / 2) for i in
range(8)]
    # print(f'arrD={arrD}')

    # cut a and b into 4 arrays
    # k = 0
    # print(f'arrAinvert={arrAinvert}')
    for j in range(len(arrAinvert[0])): # 2
        for i in range(len(arrAinvert)): # 8

            if j == int(len(arrAinvert[0]) / 2):
                arrC[i] = arrAinvert[i][j]
            else:

```

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```

        arrA[i] = arrAInvert[i][j]
#         print(f'arrA[i]={arrA[i]}')

# print(f'arrBInvert={arrBInvert}')
for j in range(len(arrBInvert[0])): # 2
    for i in range(len(arrBInvert)): # 8
        if j == int(len(arrBInvert[0]) / 2):
            arrD[i] = arrBInvert[i][j]
        else:
            arrB[i] = arrBInvert[i][j]
#         print(f'arrB[i]={arrB[i]}')

# print(f'arrA={arrA}')
# print(f'arrB={arrB}')
# print(f'arrC={arrC}')
# print(f'arrD={arrD}')

# print(f'arrA===== {arrA}')
arrA = FullArr(arrA)
# print(f'arrB===== {arrB}')
arrB = FullArr(arrB)
# print(f'arrC===== {arrC}')
arrC = FullArr(arrC)
# print(f'arrD===== {arrD}')
arrD = FullArr(arrD)

result1 = FirstFinalAddition(arrA, arrB)
# print(f'FirstFinalAddition(arrA, arrB)={result1}')
result2 = SecondFinalAddition(arrC, arrD)
# print(f'SecondFinalAddition(arrC, arrD)={result2}')

tmp = result1 + result2
tmp = tmp[::-1]
resultFinal = [[0x00] * len(a[0]) for i in range(len(a))]
k = 0
for j in range(len(a[0])):
    for i in range(len(a)):
        resultFinal[i][j] = tmp[k]
        k += 1
return resultFinal

```

Функція Перемішування стовпців

```

# ПЕРЕМІШУВАННЯ СТОВПЦІВ
def multiplyArr(arr1, arr2):
    arr3 = [[0x00] * len(arr2[0]) for i in range(8)]
    maxSizeArrayJ = len(arr2[0])
    result = [0x00] * 8
    FixArrN = 0
    NResult = 0
    iSave = 0

    while (FixArrN < maxSizeArrayJ):
        for i in range(len(arr1)):
            for j in range(len(arr1[0])):
                if (FixArrN < maxSizeArrayJ):

                    result[j] = multiply(arr1[i][j],
arr2[j][FixArrN])

                    if (
                        NResult == 7 or NResult == 15 or
NResult == 23 or NResult == 31 or NResult == 39 or NResult == 47
or NResult == 55 or NResult == 63) and (
                        j == 7 and i == 7):
                            FixArrN += 1
                            NResult += 1
                            iSave = i
                            i = 0
                            j = 0
                    if j == 7:
                        NResult += 1
                    elif FixArrN >= (len(arr2[0])):
                        break
                    if (FixArrN <= maxSizeArrayJ):
                        if NResult == 8 or NResult == 16 or NResult == 24
or NResult == 32 or NResult == 40 or NResult == 48 or NResult ==
56 or NResult == 64:
                            iSave = 7

                            arr3[iSave][FixArrN - 1] =
xorElemMatrix(result)
                        else:
                            arr3[i][FixArrN] = xorElemMatrix(result)

    return arr3

def multiply(a, b): # приймає десяткові значення, видає в
десятковій
    b = int(str(b), 16)
    # print(b)
    if (a * b) < 255:

```

```

result = a * b
tmpArr1 = ['0'] * 12
tmpArr2 = ['0'] * 12
tmpArr3 = ['0'] * 12
stra = str(bin(a))
strb = str(bin(b))
g = stra[::-1]
h = strb[::-1]
for i in range(len(g) - 2):
    tmpArr1[i] = g[i]
for i in range(len(h) - 2):
    tmpArr2[i] = h[i]

ArrToStay = [999] * 12
t = 0
ArrToDelete = [999] * 12
k = 0

for i in range(len(tmpArr1)):
    if tmpArr1[i] == '1':
        for I in range(len(tmpArr2)):
            if tmpArr2[I] == '1':
                if (i != 0 or I != 0):
                    tmpArr3[i + I] = '1'
                    if ((i + I) in ArrToStay):
                        if ((i + I) in ArrToDelete):

                            for II in
range(len(ArrToDelete)):
                                if ArrToDelete[II] == i +
I:

                                    ArrToDelete[II] = 999
                                elif ((i + I) not in
ArrToDelete):
                                    ArrToDelete[k] = i + I

                                    k += 1
                                elif ((i + I) not in ArrToStay):
                                    ArrToStay[t] = i + I

                                    t += 1
                            else:
                                tmpArr3[0] = '1'
                                ArrToStay[t] = i + I

                                t += 1

for i in range(len(ArrToDelete)):
    if ArrToDelete[i] != 999:
        tmpArr3[ArrToDelete[i]] = '0'
result = (''.join(tmpArr3)[:8])[:-1]

```

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

```

else:
    tmpArr1 = ['0'] * 24
    tmpArr2 = ['0'] * 24
    tmpArr3 = ['0'] * 24
    stra = str(bin(a))
    strb = str(bin(b))

    result = 0
    g = stra[::-1]
    h = strb[::-1]
    for i in range(len(g) - 2):
        tmpArr1[i] = g[i]
    for i in range(len(h) - 2):
        tmpArr2[i] = h[i]

    ArrToStay = [999] * 40
    t = 0
    ArrToDelete = [999] * 40
    k = 0

    for i in range(len(tmpArr1)):
        if tmpArr1[i] == '1':
            for I in range(len(tmpArr2)):
                if tmpArr2[I] == '1':
                    if (i != 0 or I != 0):
                        tmpArr3[i + I] = '1'
                        if ((i + I) in ArrToStay):

                            if ((i + I) in ArrToDelete):

                                for II in
range(len(ArrToDelete)):
                                    if ArrToDelete[II] == i +
I:
                                        ArrToDelete[II] = 999
                                elif ((i + I) not in
ArrToDelete):
                                    #print(f'k===== {k}')
                                    ArrToDelete[k] = i + I
                                    # print(f'ArrToDelete[{k}]=
{ArrToDelete[k]}')

                                    k += 1
                                elif ((i + I) not in ArrToStay):
                                    ArrToStay[t] = i + I
                                    # print(f'ArrToStay[{t}]=
{ArrToStay[t]}')

                                    t += 1
                                else:
                                    tmpArr3[0] = '1'
                                    ArrToStay[t] = i + I
                                    # print(f'====ArrToStay[{t}]=
{ArrToStay[t]}')

```

		Луцевич М. О.			ККІК.КР.Б – 125 – 9 – ПЗ	Арк.
		Байлюк Є М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

        t += 1

    for i in range(len(ArrToDelete)):
        if ArrToDelete[i] != 999:
            tmpArr3[ArrToDelete[i]] = '0'

    fixedArr = ['1', '0', '1', '1', '1', '0', '0', '0', '1',
'0', '0', '0']
    tmpDodArr = ['0'] * 24
    lastIndex = 0
    neededIndex = 0

    while (tmpArr3[8] == '1') or (tmpArr3[9] == '1') or
(tmpArr3[10] == '1') or (tmpArr3[11] == '1') or (
        tmpArr3[12] == '1') or (tmpArr3[13] == '1') or
(tmpArr3[14] == '1') or (tmpArr3[15] == '1') or (tmpArr3[16] ==
'1') or (tmpArr3[17] == '1') or (tmpArr3[18] == '1') or
(tmpArr3[19] == '1') or (tmpArr3[20] == '1') or (tmpArr3[21] ==
'1') or (tmpArr3[22] == '1') or (tmpArr3[23] == '1') :
        for i in range(len(tmpArr3)):
            if tmpArr3[i] == '1':
                lastIndex = i
        if lastIndex >= 8:
            neededIndex = lastIndex - 8
        for i in range(len(fixedArr)):
            if fixedArr[i] == '1':
                tmpDodArr[i + neededIndex] = '1'
        for i in range(len(tmpArr3)):
            if (tmpArr3[i] == '1') and (tmpDodArr[i] == '1'):
                tmpArr3[i] = '0'
                tmpDodArr[i] = '0'

        for i in range(len(tmpArr3)):
            if tmpDodArr[i] == '1':
                tmpArr3[i] = '1'
        for i in range(len(tmpArr3)):
            tmpDodArr[i] = '0'
    result = (''.join(tmpArr3)[:8])[:-1]

    return result

def xorElemMatrix(arr): # приймає в двійковій, повертає в 16ові
    xorRes = 0
    # print(f'arr:{arr}')
    for i in range(len(arr)):
        if i == 0:
            # print(type(arr[0]), arr[0])
            xorRes = int(arr[0], 2)
        else:
            xorRes = xorRes ^ int(arr[i], 2)
    return hex(xorRes)

```

