Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування факультету)

Кафедра інформаційної безпеки та наноелектроніки
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему Дослідження застосування режимів шифрування блокових шифрів
(назва теми)

Виконав(ла): студент(ка) 6 курсу, групи БК-812м
Спеціальності 125 Кібербезпека
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Безпека інформаційних і комунікаційних систем
ДЕЙНЕГА В.Р.
(ПРІЗВИЩЕ та ініціали)

Керівник КОЗІНА Г.Л.
(ПРІЗВИЩЕ та ініціали)

Рецензент САМОЙЛИК С.С.
(ПРІЗВИЩЕ та ініціали)

2023

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
Кафедра інформаційної безпеки та наноелектроніки
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека
(код і найменування)
Освітня програма (спеціалізація) Безпека інформаційних і комунікаційних мереж
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІБтаН
Андрій КОРОТУН
«____» _____2023року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДЕЙНЕГИ Вадима Романовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження застосування режимів шифрування блокових шифрів
Study of using block coding modes

керівник проєкту (роботи) к.ф.-м.н., доцент КОЗІНА Галина Леонідівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від «_28__» листопада 2023 року №475
2. Строк подання студентом проєкту (роботи) 11.12.2023
3. Вихідні дані до проєкту (роботи) Стандарти симетричного шифрування, алгоритми симетричних шифрів, режими шифрування симетричних шифрів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз стандартів симетричного шифрування, дослідження методів блокових симетричних шифрів, аналіз та порівняння режимів симетричних блокових шифрів, програмна реалізація алгоритмів режимів шифрування

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Презентація доповіді (в MS PowerPoint), 18 слайдів.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 – 4	КОЗІНА Г.Л., доцент кафедри ІБтаН	04.09.2023	04.12.2023
Нормоконтроль	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН		06.12.2023

7. Дата видачі завдання «04» вересня 2023року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Аналіз літературних джерел за тематикою дослідження	04.09.23 – 18.09.23	
2.	Огляд стандартів симетричного шифрування	18.09.23 – 25.09.23	
3.	Вибір та аналіз режимів шифрування	25.09.23 – 02.10.23	
4.	Дослідження обраних режимів	02.10.23 – 16.10.23	
5.	Програмна реалізація обраних режимів	16.10.23 – 06.11.23	
6.	Тестування створених програм	06.11.23 – 13.11.23	
7.	Формування підсумків	13.11.23 – 20.11.23	
8.	Оформлення матеріалів магістерської роботи	20.11.23 – 11.12.23	

Студент(ка) _____ Вадим ДЕЙНЕГА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Галина КОЗІНА
(підпис) (Ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

Пояснювальна записка до магістерської роботи: 109 с., 2 табл., 23 рис., 10 дод., 14 джерел.

АЛГОРИТМ ШИФРУВАННЯ, БАЗОВЕ ПЕРЕТВОРЕННЯ, ДЕШИФРУВАННЯ, ДСТУ 7624:2014, ЗАШИФРУВАННЯ, МЕРЕЖА ФЕЙСТЕЛЯ, РЕЖИМ ШИФРУВАННЯ, СИМЕТРИЧНИЙ ШИФР, AES, DES, SP-МЕРЕЖА

Актуальність теми магістерської роботи зумовлена важливістю дослідження сучасних алгоритмів шифрування та їх застосування.

Об’єкт дослідження – режими шифрування блокових шифрів.

Предмет дослідження – застосування режимів шифрування.

Мета роботи – аналіз режимів шифрування та створення програм для їх реалізації.

Задачі дослідження: проведення аналізу режимів шифрування та їх застосування.

Наукова новизна одержаних результатів полягає у створенні нової програмної реалізації для режимів шифрування.

Практичне значення одержаних результатів полягає у реалізації алгоритмів режимів шифрування у вигляді програм для зашифрування і дешифрування.

Апробація результатів: створені програми для зашифрування і дешифрування були успішно протестовані та повністю відповідають заданим умовам.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 109 pp., 2 tables, 23 figures, 10 appendixes, 14 sources.

AES, BASE TRANSFORMATION, DECRYPTION, DES, DSTU 7624:2014, ENCRYPTION, ENCRYPTION ALGORITHM, ENCRYPTION MODE, FEISTEL NETWORK, SP-NETWORK, SYMMETRICAL CIPHER

The relevance of the topic of the diploma qualifying work of the master is determined by the importance of researching modern encryption algorithms and their application.

The object of research is encryption modes of block ciphers.

The subject of research is the use of encryption modes.

The purpose of the work is to analyze encryption modes and create programs for their implementation.

Research objectives: analysis of encryption modes and their application.

The scientific novelty of the obtained results lies in the creation of a new software implementation for encryption modes.

The practical significance of the obtained results lies in the implementation of algorithms of encryption modes in the form of programs for encryption and decryption.

Approbation of the results: the created programs for encryption and decryption were successfully tested and fully meet the specified conditions.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДСТУ – Державний стандарт України;

AES – Advanced Encryption Standard;

CFB – Cipher Feedback;

DES – Data Encryption Standard;

ECB – Electronic Coding Book;

KW – Key Wrap;

NIST – National Institute of Standards & Technology;

SP – Substitution-permutation (заміна-перестановка).

ВСТУП

У сучасному світі цифрова інформація має визначальну роль у будь-якій сфері діяльності. Сьогодні неможливо уявити будь-який дистанційний обмін відомостями та подальше їх зберігання без застосування електронних засобів. Отже, гостро стоїть питання захисту важливої інформації.

Одним з найпоширеніших засобів забезпечення безпеки інформації є її шифрування. З розвитком криптології було винайдено багато різноманітних типів шифрування для вирішення різних задач захисту інформації.

У дипломній роботі розглядаються стандарти симетричного шифрування та їх окремі режими, що мають різне застосування у шифруванні інформації. Заради їх дослідження проводиться аналіз та порівняння характеристик цих режимів шифрування.

Задля повного дослідження обраних режимів шифрування потрібно також відтворити їх алгоритми за допомогою програмної реалізації процедур зашифрування та дешифрування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Сучасні інформаційно-комунікаційні технології активно впроваджуються в усі сфери людського життя. Інформаційні ресурси стають головною цінністю наукового, економічного та технічного розвитку будь-якої галузі в усьому світі.

При цьому великого значення набуває проблема захисту даних, що полягає у забезпеченні їх конфіденційності, цілісності та доступності при зберіганні, обробці та передачі.

Забезпечення конфіденційності полягає у вирішення проблеми захисту інформаційних ресурсів від несанкціонованого ознайомлення з інформацією з обмеженим доступом, яку вони зберігають.

Цілісність інформаційних ресурсів полягає у гарантуванні неможливості їх несанкціонованої підміни або модифікації.

Доступність ресурсу полягає у можливості його використання за вимогою користувача, який має відповідні повноваження.

1.1 Значимість криптології для захисту інформації

До наших часів збереглися стародавні прадавні записи, які були зроблені з використанням рунічних алфавітів. Символізм, який був закладений у кожну літеру, виконував дві функції: по-перше, він приховував таємниці від непосвячених, а по-друге, навпаки, відкривав їх тим, хто був цього гідний, хто розумів прихований зміст цих символів [1].

Це свідчить про те, що вже з тих часів люди зрозуміли, що інформація становить визначену цінність та потребує захисту. В результаті її вирішили

приховувати та почали робити це за допомогою доступних для свого часу способів. З часом з’явилась ціла сучасна наука, що вивчає це питання.

Криптологія – наука, яка вивчає методи побудови та аналізу систем захисту інформаційних ресурсів, основаних на математичних перетвореннях даних з використанням секретних параметрів [2].

Фундамент криптології як науки заклала робота американського вченого Клода Шеннона «Теорія зв’язку в секретних системах» 1949 року, у якій фактично вперше було представлено математичну модель шифрів.

1.2 Криптографія та криптоаналіз

Криптологія поєднує у собі два взаємозалежні напрями: криптографію та криптоаналіз, що показано на рисунку 1.1.

Рисунок 1.1 – Криптографія та криптоаналіз

Криптографія – наука про принципи, засоби та методи перетворення даних з метою приховування їх змісту та запобігання несанкціонованого використання або

підробки даних, що підлягають захисту. Цей напрям криптології вивчає основні закономірності, протиріччя, методи, системи та засоби забезпечення конфіденційності, цілісності та доступності інформації, ґрунтуючись на криптографічних перетвореннях.

Криптоаналіз – наука про методи та способи розкриття перетворених повідомлень, а також про тактику та стратегію їх застосування. Цей напрям криптології вивчає основні закономірності, протиріччя, методи та засоби аналізу криптографічних систем, ґрунтуючись на їх вхідних та вихідних даних, у тому числі на частині ключових даних, що здійснюється з метою визначення спеціальних даних і значущої інформації, які можуть бути використані для порушення конфіденційності, цілісності та доступності інформації, яка підлягає захисту.

1.3 Сутність шифрування

Кожне криптографічне перетворення однозначно визначається ключем (секретним параметром) та описується криптографічним алгоритмом. Криптографічний алгоритм – це набір математичних правил та процедур, який описує такі види перетворень, як шифрування, формування та перевірка цифрового підпису, обчислення хеш-значень, спеціальних криптографічних контрольних сум тощо. Сукупність криптографічних алгоритмів, що використовуються для шифрування, називають шифром.

Шифрування даних – це процес, що складається із зашифрування та дешифрування (розшифрування).

Зашифрування – процес перетворення відкритого тексту до виду, незрозумілого для несанкціонованого користувача. Відкритий текст представляє собою вихідне повідомлення, яке підлягає зашифруванню. Результатом

зашифрування відкритого тексту є шифротекст, який також називають криптотекстом або криптограмою.

Дешифрування – процес, обернений до зашифрування, це перетворення шифрованого повідомлення до виду початкової інформації (відкритого тексту) за допомогою певних правил самого шифру та відомого ключа.

Криптосистема – це система криптографічного перетворення даних, що містить у собі п’ять компонентів: множину відкритих текстів, множину шифротекстів, множину ключів, сімейства зашифровуючих та дешифровуючих перетворень. Фахівець, який займається розробкою криптосистем, називається криптографом.

Криптостійкість – це властивість криптосистеми протидіяти атакам зловмисника, спрямованим на отримання секретного ключа або відкритого повідомлення.

Стійкість криптосистеми визначається її здатністю протидіяти усім можливим атакам. Під атакою на криптосистему розуміється спроба порушення безпеки конкретної реалізації криптосистеми. Вдалу криптоатаку називають зламом.

В криптографії існує загальноприйняте правило, яке сформулював голландський вчений Огюст Керкхоф: стійкість зашифрованого повідомлення забезпечується в першу чергу ключем. Тобто передбачається ймовірність того, що сам алгоритм шифрування, шифротекст або якась його частина є відомими зловмиснику та доступні для вивчення.

Криптостійкість системи зазвичай вимірюється кількістю операцій, необхідних для перебору всіх можливих ключів, або інтервалом часу, який необхідно витратити для зламу. Вона оцінюється у процесі проведення криптографічного аналізу. Фахівець який займається розробкою методів криптоаналізу називається криптоаналітиком [2].

Відношення між описаними вище термінами можна представити у вигляді схеми обміну секретними повідомленнями, що зображена на рисунку 1.2.

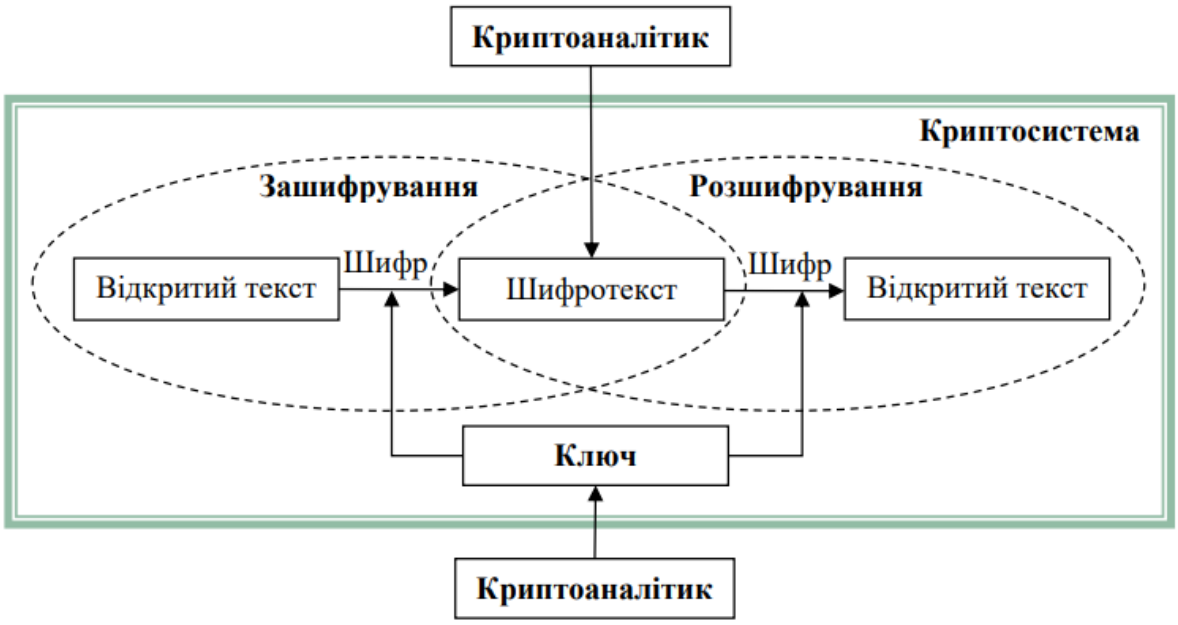

Рисунок 1.2 – Схема обміну секретними повідомленнями

З часом задачі криптології значно розширилися та вийшли за межі виключно шифрування повідомлень. На сьогоднішній день вони також включають у себе розробку систем цифрового підпису, протоколів автентифікації та ідентифікації користувачів та інші.

1.4 Класифікація шифрів

Важливо зазначити, що не існує єдиного шифру, що підходить для всіх випадків. Вибір шифру залежить від:

- особливостей інформації (вона може мати різний характер);
- цінності інформації;
- обсягів інформації;
- потрібної швидкості її передачі;
- тривалості захисту;
- можливостей потенційного зловмисника;

– можливостей власників із захисту своєї інформації.

Криптографічні системи та шифри класифікуються за різними ознаками. На рисунку 1.3 представлено схему основної класифікації шифрів.

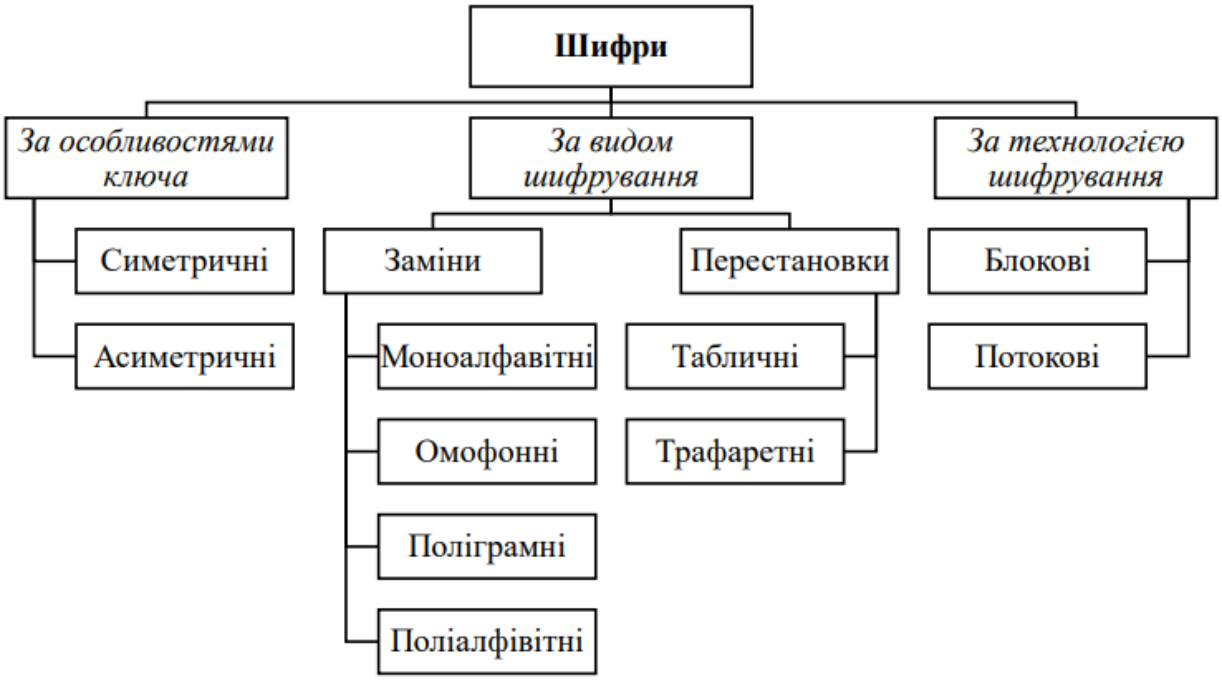

Рисунок 1.3 – Класифікація шифрів

За особливостями ключа розрізняють симетричні та асиметричні криптосистеми.

Симетрична криптосистема (із закритим ключем) – криптосистема, у якій один і той самий алгоритм, а також один і той самий ключ використовується як для зашифрування, так і для дешифрування повідомлень. На рисунку 1.4 представлено схему симетричного шифрування.

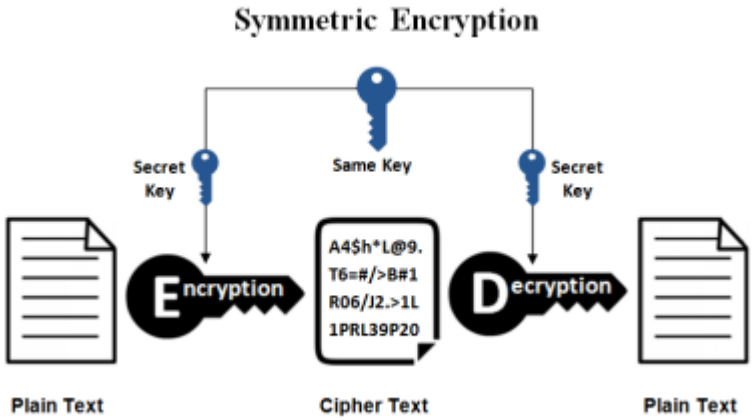

Рисунок 1.4 – Схема симетричного шифрування

Асиметрична криптосистема (із відкритим ключем, двоключова) – криптосистема, у якій використовуються два ключі: відкритий (публічний) і закритий (секретний), які математично пов'язані один з одним. Повідомлення зашифровується за допомогою відкритого ключа, який є доступним усім бажаючим, а дешифровується за допомогою закритого ключа, який відомий тільки одержувачу. На рисунку 1.5 представлено схему асиметричного шифрування.

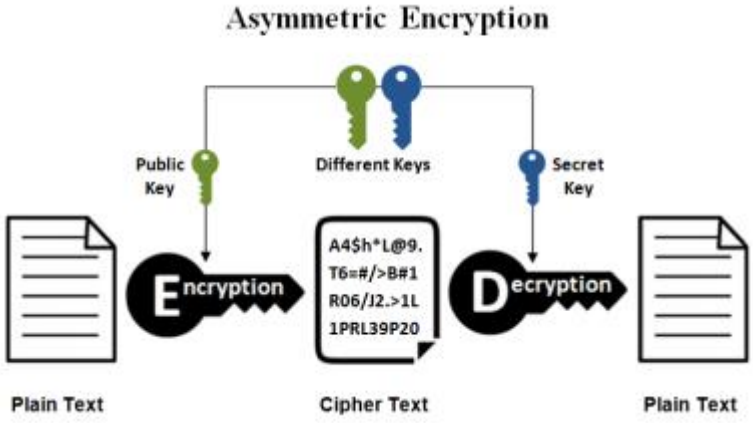

Рисунок 1.5 – Схема асиметричного шифрування

Залежно від виду криптографічного перетворення криптосистеми можуть містити шифр підстановки або шифр перестановки.

Шифр підстановки (заміни) – шифр, у якому кожен символ відкритого тексту у шифротексті замінюється іншим символом.

Найчастіше виділяють такі типи шифрів підстановки: проста підстановка (моноалфавітна заміна), однозвучна підстановка (омофонна заміна), поліграмна заміна та поліалфавітна заміна.

У простій підстановці символ відкритого тексту замінюється відповідним символом шифротексту, конкретній літері відкритого повідомлення відповідає єдина, завжди одна і та сама, літера шифротексту.

Однозвучний шифр підстановки схожий на простий шифр підстановки за винятком того, що один символ відкритого тексту символ відкритого тексту замінюється на один з декількох можливих символів шифротексту.

Поліграмний шифр підстановки – це шифр, який шифрує блоки символів по групах. Наприклад, біграма – це група з двох символів, триграма – з трьох символів і так далі.

Поліалфавітна підстановка складається з декількох простих шифрів підстановки, тобто одна і та сама літера відкритого тексту може бути замінена кожен раз по-різному (відбувається циклічне застосування декількох моноалфавітних шифрів).

Шифр перестановки – це шифр, у якому символи повідомлення переставляються місцями безпосередньо у відкритому тексті за певним правилом, що залежить від ключа. Найчастіше перестановка виконується за допомогою таблиці, комірки якої спочатку заповнюють відкритим текстом в деякому порядку, а потім шифротекст зчитують відповідно до заздалегідь визначеного алгоритму.

За технологією шифрування розрізняють блокові та потокові шифри.

Блокові шифри здійснюють шифрування блоків фіксованої довжини, що складаються з послідовності символів відкритого тексту.

Потокові шифри здійснюють шифрування окремих символів відкритого тексту.

2 СТАНДАРТИ СИМЕТРИЧНОГО ШИФРУВАННЯ

Схеми симетричного шифрування засновані на одному ключі, який використовується двома чи більше користувачами. Однаковий ключ використовується для шифрування і розшифрування так званого відкритого тексту (який представляє собою повідомлення або фрагмент даних, що зашифровується).

Процес шифрування складається з пропущення відкритого тексту через алгоритм шифрування, який називається шифром, що в свою чергу, генерує зашифрований текст.

Якщо схема шифрування досить надійна, єдиний спосіб прочитати або отримати доступ до інформації, що міститься в зашифрованому тексті – це використовувати відповідний ключ для її розшифрування [3].

Симетричне шифрування – один із двох основних методів шифрування даних у сучасних комп'ютерних системах. Іншим методом є асиметричне шифрування, яке використовує шифри з відкритим (публічним) ключем.

2.1 Розвиток стандартів симетричного шифрування

Історія симетричного шифрування має глибокі корені. Всі шифри до сімдесятих років XX століття були симетричними, а криптосистеми з відкритим ключем (асиметричні криптосистеми) були розроблені лише в другій половині сімдесятих років [4].

2.1.1 Шифр Вернама

Варто згадати так званий шифр одноразового блокноту. Цей шифр було запропоновано у 1917 році співробітниками телеграфної компанії AT&T Мейджором Джозефом Моборном та Гільбертом Вернамом.

У класичному розумінні одноразовий блокнот є унікальною послідовністю символів ключа, що згенерована випадковим чином. Заздалегідь готувалася «гама», що представляла собою перфострічку з випадковими знаками. Потім електромеханічно складалися її імпульси з імпульсами знаків відкритого тексту. Отримана сума представляла собою шифротекст. На приймальному кінці імпульси, отримані по каналу зв’язку, складалися з імпульсами тієї ж самої гами, в результаті чого відновлювалися вихідні імпульси повідомлення.

Ключ має володіти трьома важливими властивостями: бути випадковим, збігатись за розміром з відкритим текстом і застосовуватись лише один раз. Таким чином, якщо криптоаналітик спробує використовувати всі можливі набори ключів, заданих для зашифрованих даних, і відновити варіанти вхідних даних, то вони виявляться рівноймовірними і неможливо буде встановити надіслані дані.

У 1949 році Клод Шеннон опублікував роботу, в якій довів абсолютну стійкість шифру Вернама. Проте, умови яким повинен задовольняти ключ, настільки сильні, що практичне використання шифру Вернама є важко здійсненним [2].

2.1.2 Алгоритм DES

DES (Data Encryption Standard) – американський стандарт шифрування, що був прийнятий у 1977 році. Є одним з найвідоміших представників блокових шифрів.

Ключ шифрування складається з 56 випадкових бітів, до яких додається ще 8 біт в позиціях 8, 16, 24, 32, 40, 48, 56 і 64, таким чином, щоб кожен байт містив непарну кількість одиниць, що використовується при знаходженні помилок при обміні та зберіганні ключів.

Процес шифрування в алгоритмі DES полягає в початковій перестановці 64 бітів вхідного блоку, 16 циклах шифрування та кінцевій перестановці бітів. На рисунку 2.1 представлено загальну схему алгоритму.

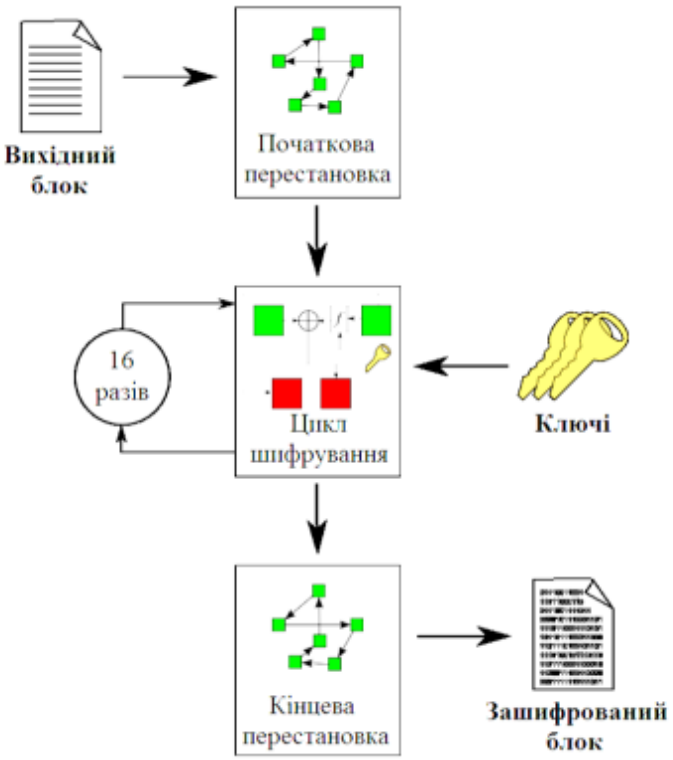

Рисунок 2.1 – Схема алгоритму DES

При дешифруванні даних всі дії у алгоритмі відбуваються в зворотному порядку.

Загалом, алгоритм DES не вважається стійким, в основному через недостатню довжину ключа. З самого початку його застосування він був визнаний вразливим через можливість здійснення атаки brute-force (атаки грубої сили) [5].

2.1.3 Алгоритм AES

У 1997 році американський інститут стандартизації NIST (National Institute of Standards & Technology) оголосив конкурс на новий стандарт алгоритму симетричного шифрування. У 2000 році переможцем було визнано криптоалгоритм Rijndael, розробниками якого були бельгійські криптографи Реймен і Даймон. Згодом цей алгоритм було затверджено як стандарт Advanced Encryption Standard (AES).

AES є симетричним ітеративним блоковим алгоритмом шифрування з довжиною блока 128 біт та змінною довжиною ключа, яка може дорівнювати 128, 192 або 256 біт. На відміну від DES, алгоритм AES не використовує збалансовану мережу Фейстеля.

AES базується на архітектурі Square, для якої властиві такі елементи:

– представлення блоку у вигляді масиву байтів;

– шифрування за один раунд всього блоку даних;

– виконання криптографічних перетворень як над окремими байтами масиву, так і над його рядками і стовпцями.

У наступному розділі цієї роботи було докладно розглянуто перетворення, що відбуваються у алгоритмі.

Згідно з дослідженнями, усі можливі атаки на алгоритм AES станом на зараз є нездійсненними з сучасними обчислювальними можливостями [6].

Цей алгоритм став одним з найпоширеніших криптоалгоритмів у світі. На його основі також було створено інші алгоритми симетричного шифрування, які прийняті як стандарти у різних країнах світу.

2.2 Методи побудови блокових симетричних шифрів

Характерною особливістю блокових шифрів є те, що дані, представлені в комп'ютерній пам'яті, розбиваються на блоки фіксованої довжини. Якщо останній фрагмент коротше довжини блоку – його доповнюють певним способом (наприклад, додають нулі).

Алгоритми зашифрування та дешифрування блокових шифрів використовують один і той самий секретний ключ. Ключем є послідовність бітів фіксованої довжини, з якої генеруються раундові ключі за яким-небудь математичним правилом.

Існують різні методи побудови блокових симетричних шифрів. Розглянемо два з них, а саме мережу Фейстеля та SP-мережу.

2.2.1 Мережа Фейстеля

Мережа Фейстеля [7] названа на честь дослідника Горста Фейстеля, який працював свого часу в корпорації IBM і був одним з авторів стандарту DES. Мережею Фейстеля є загальний метод перетворення за допомогою довільної функції (яку називають також f-функцією) у перестановку на множині блоків. Цю конструкцію було використано в багатьох шифрах, найвідомішими з яких є DES, Camellia, Tworish.

У мережі Фейстеля кожен блок даних розбивається на дві рівні частини, ліву (L) та праву (R). Права частина видозмінюється деякою функцією f(R,K) залежно від раундового ключа K. Здійснюється додавання за модулем 2 лівої частини L та f(R,K). Результат додавання присвоюється новому правому підблоку, а минулий правий підблок присвоюється без змін новому лівому підблоку. Ці операції повторюються N раундів. При переході від одного раунду до іншого заміняються раундові ключі.

На рисунку 2.2 зображено схему мережі Фейстеля.

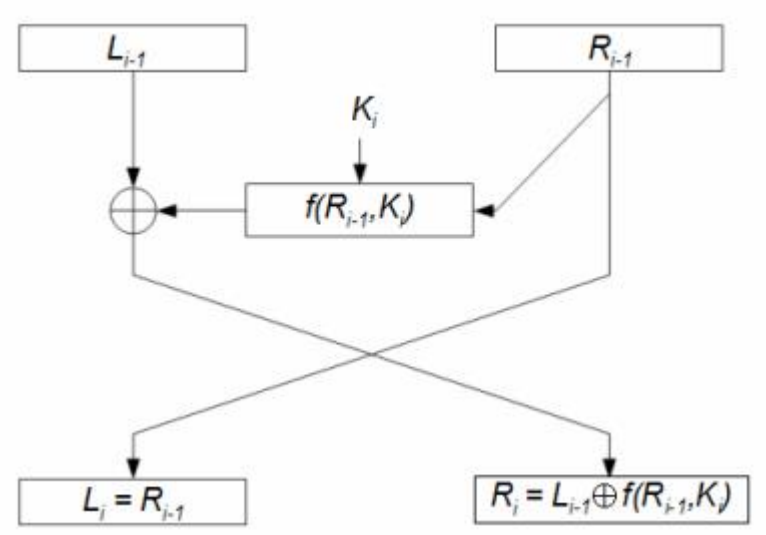

Рисунок 2.2 – Схема мережі Фейстеля

Дешифрування у цій схемі відбувається так само, як і зашифрування, але раундові ключі йдуть у зворотному порядку.

2.2.2 SP-мережа

Підхід до побудови блокового шифру, що містить послідовні перестановки й підстановки, називають мережею замін-перестановок (substitution-permutation) або

SP-мережею. Прикладом здійснення такого підходу до побудови шифру є стандарт блокового симетричного шифрування AES.

В основі роботи SP-мережі лежать два методи маскування надлишковості відкритого тексту, описані Клодом Шенноном [8], якими є плутанина (англ. confusion) та поширення (англ. diffusion).

Плутанина маскує зв’язок між відкритим текстом і шифротекстом. Вона ускладнює спроби знайти у шифротексті надлишковість і статистичні закономірності. Простим способом застосування цього метода є підстановка.

Поширення розсіює надлишковість відкритого тексту, поширюючи її по всьому шифротексту. Простим способом здійснити плширення є перестановка (транспозиція). Сучасні шифри використовують форми поширення, які дають можливість розкидати частини повідомлення по всьому повідомленню, що сприяє виникненню так званого лавинного ефекту. Це означає, що навіть невеликі зміни в початкових даних (чи в ключі) можуть викликати значні зміни в зашифрованих даних.

У SP-мережі для кожного блоку відкритого тексту застосовуються декілька раундів S-скринь (англ. S-box) і P-скринь (англ. P-box), що чергуються. В результаті перетворень отримують блок шифротексту. На рисунку 2.3 зображено приклад схеми SP-мережі.

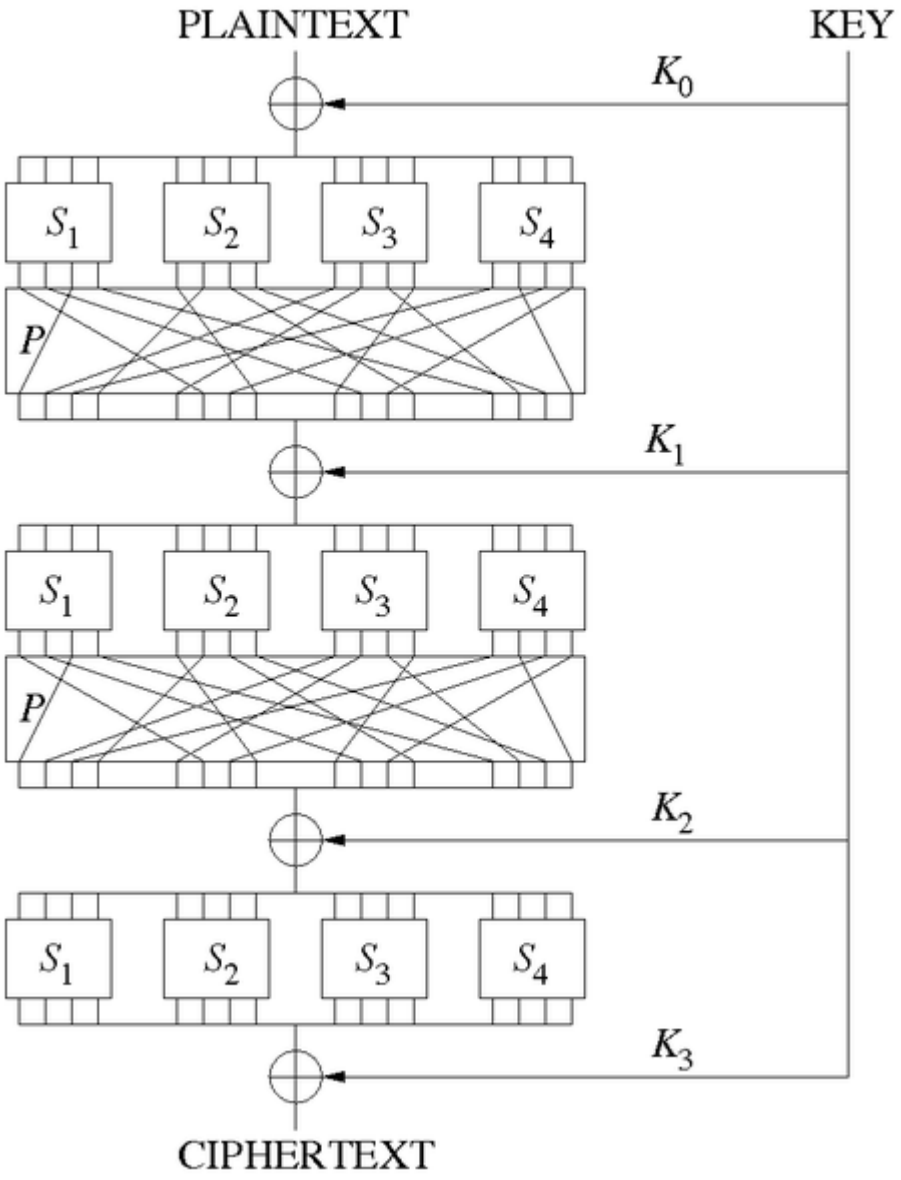

Рисунок 2.3 – Приклад схеми SP-мережі

Розглянемо схему шифрування у алгоритмі AES. У ньому для кожного блоку даних відбуваються наступні перетворення:

а) додавання раундового ключа до початкового блоку за модулем 2 (AddRoundKey);

б) визначена кількість раундів, крім останнього, з наступними перетвореннями:

1) підстановка байтів (SubBytes);

2) зсув рядків (ShiftRows);

3) перемішування стовпців (MixColumns);

4) додавання раундового ключа за модулем 2 (AddRoundKey).

в) останнього раунду, з наступними перетвореннями:

1) підстановка байтів (SubBytes);

2) зсув рядків (ShiftRows);

3) додавання раундового ключа за модулем 2 (AddRoundKey).

Для дешифрування у алгоритмі AES застосовуються наступні перетворення:

а) додавання раундового ключа до початкового блоку за модулем 2 (AddRoundKey);

б) визначена кількість раундів, крім останнього, з наступними перетвореннями:

1) обернений зсув рядків (InvShiftRows);

2) обернена підстановка байтів (InvSubBytes);

3) обернене перемішування стовпців (InvMixColumns);

4) додавання раундового ключа за модулем 2 (AddRoundKey).

в) останнього раунду, з наступними перетвореннями:

1) обернений зсув рядків (InvShiftRows);

2) обернена підстановка байтів (InvSubBytes);

3) додавання раундового ключа за модулем 2 (AddRoundKey).

У національному стандарті ДСТУ 7624:2014 (Калина) описаний алгоритм шифрування, що створений на базі алгоритму AES та містить дуже подібні криптоперетворення [9]. Його значною перевагою є можливість застосування максимальної довжини блоків та ключів, що дорівнює 512 біт, в той час як AES та більшість інших відомих алгоритмів обмежені максимальною довжиною, що дорівнює 256 біт [10]. На основі цього стандарту розглянемо режими шифрування у алгоритмах симетричного шифрування ті їх можливе використання.

3 АНАЛІЗ ТА ПОРІВНЯННЯ РЕЖИМІВ ШИФРУВАННЯ

Режим шифрування – метод застосування блокового шифру, що дозволяє перетворити послідовність блоків відкритих даних у послідовність блоків зашифрованих даних. При цьому для шифрування одного блоку можуть використовуватися дані іншого блоку.

Зазвичай режими шифрування використовуються для модифікації процесу шифрування так, щоб результат шифрування кожного блоку був унікальним незалежно від даних, що шифруються і не дозволяв зробити будь-які висновки про їх структуру [2].

Різні режими можуть мати різне практичне застосування та вирішувати різноманітні задачі захисту інформації в інформаційних системах. В залежності від особливостей алгоритмів режимів шифрування вони можуть мати як певні переваги, так і певні недоліки у порівнянні з іншими режимами.

Розглянемо режими шифрування симетричних алгоритмів на основі режимів ДСТУ 7624:2014 (Калина). Цей стандарт описує використання 10 режимів шифрування, які можуть використовуватися для забезпечення конфіденційності та цілісності інформації, яка підлягає захисту.

Кожен режим шифрування, який визначений у цьому стандарті, записується наступним чином: «Калина-l/k-позначення режиму-параметри режиму» (для деяких режимів параметри відсутні), де l – розмір блоку базового перетворення, k – довжина ключа. Позначення режимів відповідають загальновідомим позначенням для відповідних режимів шифрування.

Режими криптографічного алгоритму, визначеного в цьому стандарті, їх позначення та послуги безпеки, які забезпечує відповідний режим, визначені у таблиці 3.1 [11].

Таблиця 3.1 – Режими ДСТУ 7624:2014

№ режиму	Назва режиму	Позначення	Послуга безпеки
1	Проста заміна (базове перетворення)	ECB	Конфіденційність
2	Гамування	CTR	Конфіденційність
3	Гамування зі зворотнім зв’язком за шифртекстом	CFB	Конфіденційність
4	Вироблення імітовставки	CMAC	Цілісність
5	Зчеплення шифроблоків	CBC	Конфіденційність
6	Гамування зі зворотнім зв’язком за шифргамою	OFB	Конфіденційність
7	Вибіркове гамування із прискореним виробленням імітовставки	GCM, GMAC	конфіденційність і цілісність (GCM), тільки цілісність (GMAC)
8	Вироблення імітовставки і гамування	CCM	цілісність і конфіденційність
9	Індексованої заміни	XTS	конфіденційність
10	Захисту ключових даних	KW	конфіденційність і цілісність

Розглянемо докладно три режими шифрування з тих, що зазначені вище у таблиці, а саме:

а) режим простої заміни.

б) режим гамування зі зворотнім зв’язком за шифртекстом.

в) режим захисту ключових даних.

3.1 Режим простої заміни

Режим простої заміни є компонентом усіх інших режимів роботи криптографічного алгоритму симетричного блокового перетворення. Він використовується в якості базового перетворення.

Базове перетворення реалізує пряме перетворення (зашифрування) та обернене перетворення (дешифрування).

Позначається цей режим як ECB (Electronic Coding Book).

На рисунку 3.1 представлено схему режиму простої заміни.

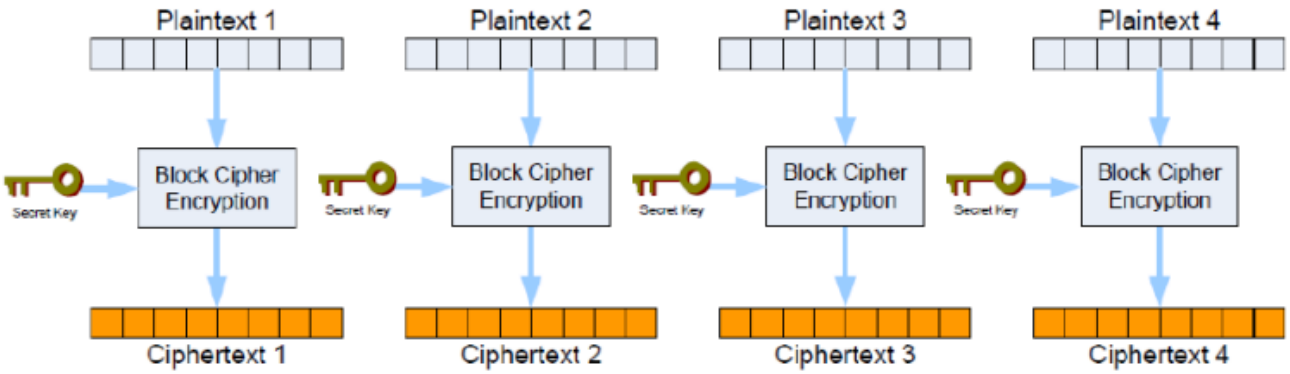

Рисунок 3.1 – Режим простої заміни

3.1.1 Опис режиму ECB

Базове перетворення зашифрування $T_{l,k}^{(K)}$ є параметризованим ключем шифрування K відображенням $T_{l,k}^{(K)}: V_l \to V_l,\ K \in V_k,\ l,k \in \{128, 256, 512\}$ при цьому $k = l$ або $k = 2 \cdot l$, що реалізоване у вигляді ітеративного застосування низки функцій, які обробляють вхідний аргумент $x \in V_l$ як матрицю внутрішнього стану розміром $8 \times c$ байтів, що містить елементи поля GF(2^8).

Базове перетворення розшифрування $U_{l,k}^{(K)}$ є параметризованим ключем шифрування K відображенням, оберненим до $T_{l,k}^{(K)}$, також реалізованим у вигляді ітеративного перетворення.

Залежність кількості ітерацій (t) при реалізації перетворень $T_{l,k}^{(K)}$ та $U_{l,k}^{(K)}$, кількості стовпців матриці внутрішнього стану (c) від розміру блоку (l) і довжини ключа шифрування наведено у таблиці 3.2.

Таблиця 3.2 – Параметри базового перетворення

№ з/п	Розмір блоку (l)	Довжина ключа (k)	Кількість ітерацій перетворення (t)	Кількість стовпців в матриці (c)
1	128	128	10	2
2		256	14	
3	256	256	14	4
4		512	18	
5	512	512	18	8

Базове перетворення виконує обробку вхідного блоку даних довжиною l бітів (відкритий текст при зашифруванні або шифртекст при розшифруванні).

Матриця внутрішнього стану позначається як $G = (g_{i,j}), g_{i,j} \in GF(2^8)$, де $i = 0,7$, $j = 0, c - 1$.. Запис байтів $B_1, B_2, \dots, B_{l/8}$ для перетворень $T_{l,k}^{(K)}$ та $U_{l,k}^{(K)}$ до матриці і зчитування з неї здійснюється по стовпцях.

Приклад запису байтів до внутрішнього стану для $l = 512, k = 512, c = 8$ зображено на рисунку 3.2.

Вхідна послідовність							
B_1	B_9	B_{17}	B_{25}	B_{33}	B_{41}	B_{49}	B_{57}
B_2	B_{10}	B_{18}	B_{26}	B_{34}	B_{42}	B_{50}	B_{58}
B_3	B_{11}	B_{19}	B_{27}	B_{35}	B_{43}	B_{51}	B_{59}
B_4	B_{12}	B_{20}	B_{28}	B_{36}	B_{44}	B_{52}	B_{60}
B_5	B_{13}	B_{21}	B_{29}	B_{37}	B_{45}	B_{53}	B_{61}
B_6	B_{14}	B_{22}	B_{30}	B_{38}	B_{46}	B_{54}	B_{62}
B_7	B_{15}	B_{23}	B_{31}	B_{39}	B_{47}	B_{55}	B_{63}
B_8	B_{16}	B_{24}	B_{32}	B_{40}	B_{48}	B_{56}	B_{64}

↓

Внутрішній стан базового перетворення							
$g_{0,0}$	$g_{0,1}$	$g_{0,2}$	$g_{0,3}$	$g_{0,4}$	$g_{0,5}$	$g_{0,6}$	$g_{0,7}$
$g_{1,0}$	$g_{1,1}$	$g_{1,2}$	$g_{1,3}$	$g_{1,4}$	$g_{1,5}$	$g_{1,6}$	$g_{1,7}$
$g_{2,0}$	$g_{2,1}$	$g_{2,2}$	$g_{2,3}$	$g_{2,4}$	$g_{2,5}$	$g_{2,6}$	$g_{2,7}$
$g_{3,0}$	$g_{3,1}$	$g_{3,2}$	$g_{3,3}$	$g_{3,4}$	$g_{3,5}$	$g_{3,6}$	$g_{3,7}$
$g_{4,0}$	$g_{4,1}$	$g_{4,2}$	$g_{4,3}$	$g_{4,4}$	$g_{4,5}$	$g_{4,6}$	$g_{4,7}$
$g_{5,0}$	$g_{5,1}$	$g_{5,2}$	$g_{5,3}$	$g_{5,4}$	$g_{5,5}$	$g_{5,6}$	$g_{5,7}$
$g_{6,0}$	$g_{6,1}$	$g_{6,2}$	$g_{6,3}$	$g_{6,4}$	$g_{6,5}$	$g_{6,6}$	$g_{6,7}$
$g_{7,0}$	$g_{7,1}$	$g_{7,2}$	$g_{7,3}$	$g_{7,4}$	$g_{7,5}$	$g_{7,6}$	$g_{7,7}$

Рисунок 3.2 – Заповнення внутрішнього стану

3.1.2 Алгоритм зашифрування у режимі ECB

Базове перетворення зашифрування $T_{l,k}^{(K)}$ визначено формулою (3.1):

$$T_{l,k}^{(K)} = \eta_l^{(K_l)} \circ \psi_l \circ \tau_l \circ \pi_l' \circ \left(\prod_{v=l}^{t-1} (\kappa_l^{(K_v)} \circ \psi_l \circ \tau_l \circ \pi_l')\right) \circ \eta_l^{(K_0)}, \quad (3.1)$$

де l – розмір внутрішнього стану блокового шифру (у бітах);

K – ключ шифрування;

k – довжина ключа шифрування (у бітах);

$\eta_l^{(K_v)}$ – функція додавання циклового ключа $K_v (v \in \{0, t\}$ за модулем 2^{64};

π_l' – шар нелінійного бієктивного відображення, який виконує обробку векторів, заданих над V_8 (байтова підстановка);

τ_l – перестановка елементів $g_{i,j} \in GF(2^8)$ внутрішнього стану (циклічний зсув рядків вправо при матричному поданні);

ψ_l – лінійне перетворення(множення матриці лінійного перетворення на матрицю внутрішнього стану над скінченним полем);

$\kappa_l^{(K_v)}$ – функція додавання циклового ключа K_v ($v \in \{1, 2, \dots, t-1\}$) за модулем 2 (інволютивне перетворення).

В функціях π_l' , τ_l і ψ_l вхідний аргумент $x \in V_l$ та вихідне значення $\chi(x) \in V_l$, $\chi \in \{\pi_l'$, τ_l , $\psi_l\}$ розглядаються як матриці розміром $8 \times c$ байтів (табл. 1.2)

Функції $\eta_l^{(K_v)}$ і $\kappa_l^{(K_v)}$ залежать від параметра $K_v \in V_l$ (циклового ключа ν -ї ітерації), мають вхідний аргумент $x \in V_l$ (внутрішній стан шифру), та вихідне значення $\chi \in \{\eta_l^{(K_v)}$, $\kappa_l^{(K_v)}\}$, при цьому вхідні аргументи та вихідне значення розглядаються як матриці розміром $8 \times c$ байтів.

3.1.3 Алгоритм дешифрування у режимі ECB

Базове перетворення розшифрування $U_{l,k}^{(K)}$ визначено формулою (3.2):

$$U_{l,k}^{(K)} = {}_{-1}\eta^{(K_0)} \circ \left(\prod_{v=t-1}^{1} \left({}_{-1}\pi_l'^{\circ}{}_{-1}\tau_l^{\circ}{}_{-1}\psi_l^{\circ}\kappa_l^{(K_v)}\right)\right)^{\circ}{}_{-1}\pi_l'^{\circ}{}_{-1}\tau_l^{\circ}{}_{-1}\psi_l^{\circ}{}_{-1}\eta^{(K_t)}, \quad (3.2)$$

де *l* – розмір внутрішнього стану блокового шифру (у бітах),

K – ключ шифрування;

k – довжина ключа шифрування (у бітах);

${}_{-1}\eta^{(K_v)}$ – функція віднімання циклового ключа K_v $(v \in \{1,2,\dots,t-1\})$ за модулем 2^{64} (обернена до $\eta_l^{(K_v)}$);

${}_{-1}\psi_l$ – обернене лінійне перетворення (множення матриці оберненого лінійного перетворення на матрицю внутрішнього стану над скінченним полем);

${}_{-1}\tau_l$ – обернена перестановка елементів $g_{i,j} \in GF(2^8)$ внутрішнього стану (циклічний зсув рядків вліво при матричному поданні);

${}_{-1}\pi_l'$ – шар оберненого нелінійного бієктивного відображення, який виконує обробку векторів, заданих над V_8 (обернена байтова підстановка);

$\boldsymbol{\kappa}_l^{(K_v)}$ – інволютивна функція додавання циклового ключа K_v $(v \in \{1,2,\dots,t-1\})$ за модулем 2 (однакова для зашифрування і розшифрування).

Як і при зашифруванні, в функціях ${}_{-1}\pi_l'$, ${}_{-1}\tau_l$, ${}_{-1}\psi_l$ вхідний аргумент $x \in V_l$ та вихідне значення $\chi(x) \in V_l$, $\chi \in \{{}_{-1}\pi_l', {}_{-1}\tau_l, {}_{-1}\psi_l\}$ розглядаються як матриці розміром 8×c байтів.

Функція ${}_{-1}\eta^{(K_v)}$ має два вхідних аргументи $x \in V_l$ (внутрішній стан шифру) і $K_v \in V_l$ (цикловий ключ ν -ї ітерації) та вихідне значення ${}_{-1}\eta^{(K_v)}(x, K_v) \in V_l$, при цьому вхідні аргументи та вихідне значення розглядаються як матриці розміром 8×c байтів.

3.1.4 Аналіз режиму ECB

Режим простої заміни – це найпростіший режим роботи блокових шифрів.

Усі блоки відкритого тексту шифруються окремо та незалежно один від одного із застосуванням того самого ключа шифрування.

Як було зазначено раніше, режим простої заміни використовується в якості базового перетворення. Без додаткових перетворень, визначених іншими режимами, використання простої заміни для захисту даних не рекомендується [12].

3.2 Режим гамування зі зворотнім зв’язком за шифртекстом

Режим гамування зі зворотнім зв’язком за шифртекстом відноситься до режимів, що створені відповідно до стандарта ISO/IEC 10116:2006 та є відносно простим режимом шифрування.

Режим забезпечує конфіденційність повідомлення шляхом шифрування. Шифрування виконує пряме відображення повідомлення у шифртекст та обернене відображення шифртексту в повідомлення.

Режим гамування зі зворотнім зв’язком за шифртекстом позначається як CFB (Cipher Feedback).

3.2.1 Опис режиму CFB

Параметрами режиму гамування зі зворотнім зв’язком за шифртекстом є ключ шифрування $|K| = k$, синхропосилка $|S| = l$ та додаткове значення q, яке визначає кількість бітів повідомлення, що обробляються за допомогою одного застосування базового перетворення. Рекомендованим значенням параметра є $q = l$.

Додатковою вимогою до синхропосилки в цьому режимі є випадковість, в тому числі непередбачуваність значення, яке буде застосовано для будь-якого повідомлення, до його формування.

Вимоги на кратність довжини повідомлення розміру блоку базового перетворення не накладаються.

На рисунку 3.2 представлено схему роботи цього режиму.

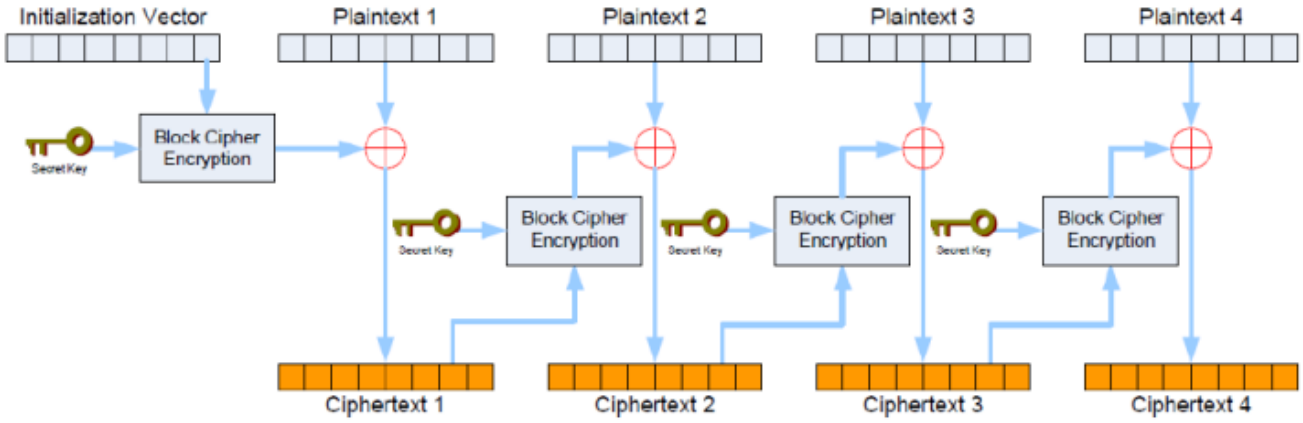

Рисунок 3.2 – Режим гамування зі зворотнім зв’язком за шифртекстом

3.2.2 Зашифрування у режимі CFB

Повідомлення M ($|M| \geq 1$) подається у вигляді послідовності блоків за формулою (3.3):

$$M = m_1||m_2|| \dots ||m_n\ , |m_i| = q \text{ для } i = 1,2, \dots, n-1, 1 \leq |m_n| \leq q \qquad (3.3)$$

Встановлюється значення $c_0^{\#} = T_{l,k}^{(K)}(S), \left|c_0^{\#}\right| = l$.

Кожен з блоків шифртекста c_i ($|c_i| = q$) обчислюється відповідно до співвідношення (3.4):

$$c_i = m_i \oplus R_{l,|m_i|}\left(c_{i-1}^{\#}\right) \text{ для } i = 1,2, \dots, n-1, \left|c_i^{\#}\right| = l \qquad (3.4)$$

Результатом зашифрування повідомлення є шифртекст $C = c_1||c_2|| \dots ||c_n$.

3.2.3 Алгоритм розшифрування у режимі CFB

Шифртекст C ($|C| \geq 1$) подається у вигляді послідовності блоків за формулою (3.5):

$$C = c_1||c_2|| \dots ||c_n\ , |c_i| = q \text{ для } i = 1,2, \dots, n-1, 1 \leq |c_n| \leq q \quad (3.5)$$

Встановлюється значення $c_0^{\#} = T_{l,k}^{(K)}(S)$.

Кожен з блоків повідомлення обчислюється відповідно до співвідношення (3.6):

$$m_i = c_i \oplus R_{l,|c_i|}(c_{i-1}^{\#}) \text{ для i=1,2,…,n,}$$
$$c_i^{\#} = T_{l,k}^{(K)}(L_{l,l-q}(c_{i-1}^{\#})||c_i) \text{ для } i = 1,2, \dots, n-1 \quad (3.6)$$

Результатом розшифрування є повідомлення $M = m_1||m_2|| \dots ||m_n$.

3.2.4 Аналіз режиму CFB

Режим гамування зі зворотнім зв’язком за шифртекстом належить до більш простих режимів шифрування і тому має як переваги, так і недоліки.

Головною перевагою цього режиму можна назвати швидкість його роботи та велику кількість блоків, що можуть захищатися на одному ключі.

Головним недоліком цього режиму є вразливість до модифікацій (модифікація шифртексту призводить до модифікації відкритого тексту), тому при застосуванні цього режиму шифрування потрібний додатковий контроль за цілісністю повідомлень.

3.3 Режим захисту ключових даних

Режим захисту ключових даних відноситься до додаткових режимів, які є більш складними та використовуються у сучасних засобах криптографічного захисту.

Режим забезпечує конфіденційність та цілісність повідомлення.

Шифрування виконує пряме відображення повідомлення у шифртекст та обернене відображення шифртексту в повідомлення.

Режим захисту ключових даних позначається як KW (без доповнення) або KW-p (із доповненням).

3.3.1 Опис режиму KW/ KW-p

Параметром режиму є ключ шифрування K, $|K| = k$

У разі, коли розмір повідомлення є кратним розміру блоку базового перетворення, виконується шифрування без доповнення. У іншому випадку застосовується алгоритм із доповненням.

Вимоги на кратність довжини повідомлення розміру блоку базового перетворення не накладаються.

3.3.2 Алгоритм зашифрування без доповнення у режимі KW

До повідомлення M ($|M| = l \cdot r$, де r – додатне ціле) додається 0^l для отримання M^*: $M^* = M||0^l$.

M^* подається у вигляді послідовності напівблоків за формулою (3.7):

$$M^* = m_1^* || m_2^* || .. || m_n^* , |m_i^*| = l/2 \text{ для } i = 1,2,..,n;$$
$$n = 2 \cdot (r + 1) \quad (3.7)$$

Встановлюються значення за формулами (3.8) та (3.9):

$$V = (n - 1) \cdot 6, \quad (3.8)$$
$$B^0 = m_l^*, \quad (3.9)$$

де $\left|B^j\right| = l/2$ для $j = 0,1,..,V$.

Задається (3.10):

$$b_i^0 = m_i^* \text{ для } i = 2,..,n , \quad (3.10)$$

де $\left|b_i^j\right| = l/2$ для $j = 0,1,..,V$.

Для $j = 0,1,..,V$ обчислюється (3.11):

$$B^j = R_{l,l/2}(T_{l,k}^{(K)}(B^{j-1}||b_2^{j-1}) \oplus \mu_{l/2}^{(j)} ,$$
$$b_n^j = L_{l,l/2}(T_{l,k}^{(K)}(B^{j-1}||b_2^{j-1}) \oplus \mu_{l/2}^{(j)},$$
$$b_i^j = b_{i+1}^{j-1} \text{ для } i = 2,..,n - 1 \quad (3.11)$$

Задається (3.12):

$$c_1 = B^V,$$
$$c_i = b_i^V \text{ для } i = 2,..,n \quad (3.12)$$

Результатом є шифртекст $C = c_1 || c_2 || ... || c_n$.

3.3.3 Алгоритм розшифрування без доповнення у режимі KW

Шифртекст C подається у вигляді послідовності напівблоків (3.13):

$$C = c_1 \,||\, c_2 \,||\, \ldots ||c_n \,, |c_i| = \frac{l}{2}\; i = 1,2,..,n,$$
$$n = 2 \cdot r \tag{3.13}$$

Встановлюється значення за формулою (3.14):

$$V = (n-1) \cdot 6 \text{ і } B^V = c_1, \tag{3.14}$$

де $\left|B^j\right| = l/2$ для $j = 0,1,..,V$.

Для $j = V, V-1,..,1$ обчислюється (3.15):

$$B^{j-1} = L_{l,\frac{l}{2}}(\, U_{l,k}^{(K)}(b_n^j || \left(B^j \oplus \mu_{\frac{l}{2}}^{(j)}\right))\,,$$
$$b_2^{j-1} = R_{l,\frac{l}{2}}(\, U_{l,k}^{(K)}(b_n^j || \left(B^j \oplus \mu_{\frac{l}{2}}^{(j)}\right))),$$
$$b_{i+1}^{j-1} = b_i^j \text{ для } i = 2,..,n \tag{3.15}$$

Формується $M^* = m_1^* || \, m_2^* \,||\, .. ||m_n^*$

У разі, коли $R_{n\cdot\frac{l}{2},l}(M^*)$ не дорівнює 0^l, повертається повідомлення про порушення цілісності. У іншому випадку повертається розшифроване повідомлення $M = L_{n\cdot\frac{l}{2},n\cdot\frac{l}{2}-l}(M^*)$.

3.3.4 Алгоритм зашифрування із доповненням у режимі KW-p

До повідомлення M ($|M| > l$) додається бітове подання довжини $\mu_{l/2}^{(|M|)}$, після чого до результату ($M \,||\, \mu_{l/2}^{(|M|)}$) застосовується алгоритм доповнення для отримання доповненого повідомлення $M'' = m_1''||\ m_2''\ ||\ m_{n_m}''$, $|m_i''| = l$ для $i = 1,2,..,n_m$.

Доповнене повідомлення M'' обробляється відповідно до алгоритма зашифрування без доповнення для отримання шифртексту $C = c_1\ ||\ c_2\ ||\ ...||c_{n_m}$, який є результатом роботи режиму.

3.3.5 Алгоритм розшифрування із доповненням у режимі KW -p

Шифртекст C ($|C| = l \cdot r$, де r – додатне ціле) обробляються відповідно до алгоритма розшифрування без доповнення для отримання доповненого повідомлення M''.

Якщо результатом роботи є повідомлення про порушення цілісності, подальша обробка припиняється із повертанням повідомлення про порушення цілісності.

У іншому випадку до доповненого повідомлення M″ застосовується алгоритм зняття доповнення повідомлення. Якщо результатом є помилка оберненого перетворення, подальша обробка припиняється із повідомленням про порушення цілісності.

У разі, коли $\mu_{l/2}^{(|M''|-l/2)} \neq R_{|M''|,\frac{l}{2}}(M'')$, повертається повідомлення про порушення цілісності

При $\mu_{l/2}^{(|M''|-l/2)} = R_{|M''|,\frac{l}{2}}(M'')$, результатом роботи режиму є відкритий текст $M = L_{|M''|,|M''|-\frac{l}{2}}(M'')$.

3.3.6 Аналіз режиму KW/ KW-p

Режим захисту ключових даних забезпечує надійний захист інформації, шо потребує додатковий рівень захисту та дозволяє захищати блоки даних без використання синхропосилки.

Він забезпечує як конфіденційність даних, так і їх цілісність, що є перевагою над більшістю інших режимів шифрування.

Істотним недоліком режиму захисту ключових даних є швидкість його роботи, яка є нижчою, ніж у інших режимах. Ця особливість виникає через необхідність забезпечення залежності всіх блоків шифртексту від кожного блоку повідомлення (і навпаки).

3.4 Порівняння досліджених режимів

Режим простої заміни не використовують окремо, він застосовується в якості базового перетворення в алгоритмах режимів, отже його порівняння з іншими режимами не має сенсу.

Режим гамування зі зворотнім зв’язком за шифртекстом та режим захисту ключових даних істотно відрізняються один від одного.

Режим гамування зі зворотнім зв’язком за шифртекстом є простішим, але більш швидким у роботі. Він підходить для вирішення задачі забезпечення

конфіденційності повідомлення, але для забезпечення його цілісності потребується додатковий захист.

Режим захисту ключових даних є більш надійним, він потрібний для більш складного захисту даних. Цей режим забезпечує одночасно конфіденційність та цілісність інформації, але при його використанні прийдеться жертвувати швидкістю роботи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ РЕЖИМІВ ШИФРУВАННЯ

Для розглянутих у минулому розділі режимів шифрування було створено програмні реалізації алгоритмів зашифрування і розшифрування. Для цього було написано окремі програми, що відтворюють послідовність дій для зашифрування і розшифрування, що вказані у стандарті.

4.1 Вибір мови програмування

Для вирішення задачі можливе використання різноманітних мов програмування та середовищ розробки для них.

При виборі мови програмування для роботи були розглянуті наступні критерії:

– актуальність на сьогоднішній день;
– зручність для написання коду;
– загальна розбірливість та наочність.

Цим критеріям повністю відповідає сучасна мова програмування Python. Вона є широко поширеною завдяки її ефективності та легкості для розуміння [13].

В якості середовища розробки було використано стандартне середовище IDLE, яке поширюється офіційно з самою мовою програмування як інтегроване середовище розробки.

4.2 Реалізація режиму простої заміни

Для програмної реалізації режиму простої заміни (базового перетворення) шифру Калина було вже створено чимало зразків програм.

Оскільки у роботі було поставлено задачу дослідження різних режимів шифрування, написання програмного коду для алгоритму простої заміни не має практичної користі.

Для подальшого використання цього режиму у якості базового перетворення для інших режимів застосовується відкритий програмний код для його реалізації мовою Python [14]. Цей код відтворює криптоперетворення у цьому режимі з довжиною блоків та ключа, що дорівнюють 128 біт.

4.3 Реалізація режиму гамування зі зворотнім зв’язком за шифртекстом

Було створено дві програми, що відтворюють криптоперетворення режиму гамування зі зворотнім зв’язком за шифртекстом для зашифрування та дешифрування відповідно.

Для відтворення базового перетворення застосовується програмний код, зазначений у підрозділі 4.2, отже для цих перетворень довжина блоків та ключа дорівнює 128 біт. Додаткове значення, яке визначає кількість бітів повідомлення, що обробляються за допомогою одного застосування базового перетворення, дорівнює довжині синхропосилки, отже також дорівнює 128 біт.

4.3.1 Програма для зашифрування у режимі CFB

Програма для зашифрування використовує в якості вхідних значень ключ шифрування, значення синхропосилки та повідомлення (відкритий текст, який потрібно зашифрувати). Значення задаються у шістнадцятковій системі числення.

Результатом виконання програми є зашифроване повідомлення, відображене у шістнадцятковій системі числення.

Повний текст програми для зашифрування подано у додатку А.

На рисунках 4.1-4.2 представлено знімок екрану виконання програми на етапі введення вхідних даних та на етапі завершення роботи з отриманням результату.

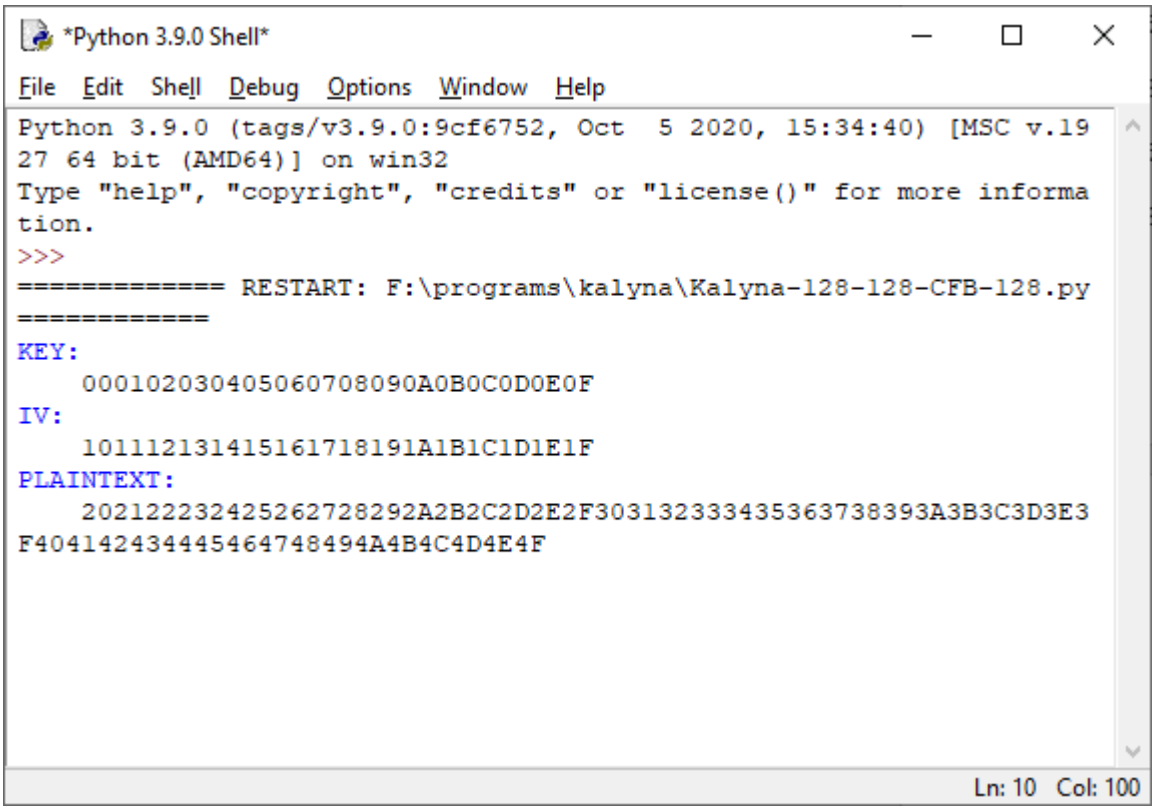

Рисунок 4.1 – Введення вхідних даних для зашифрування у режимі CFB

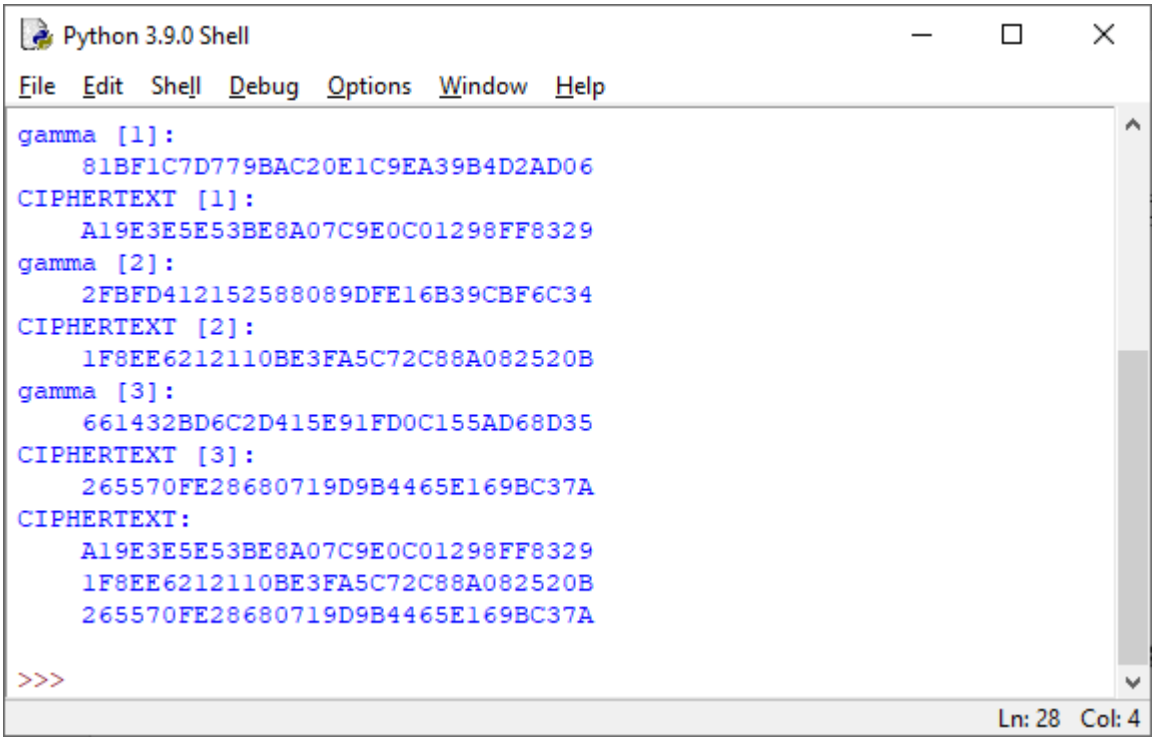

Рисунок 4.2 – Отримання результату для зашифрування у режимі CFB

Повний приклад результату роботи цієї програми подано у додатку Б.

4.3.2 Програма для дешифрування у режимі CFB

Програма для дешифрування використовує в якості вхідних значень ключ шифрування, значення синхропосилки та зашифроване повідомлення (зашифрований текст, який потрібно розшифрувати). Значення задаються у шістнадцятковій системі числення.

Результатом виконання програми є початкове повідомлення, відображене у шістнадцятковій системі числення.

Повний текст програми для зашифрування подано у додатку В.

На рисунках 4.3-4.4 представлено знімок екрану виконання програми на етапі введення вхідних даних та на етапі завершення роботи з отриманням результату.

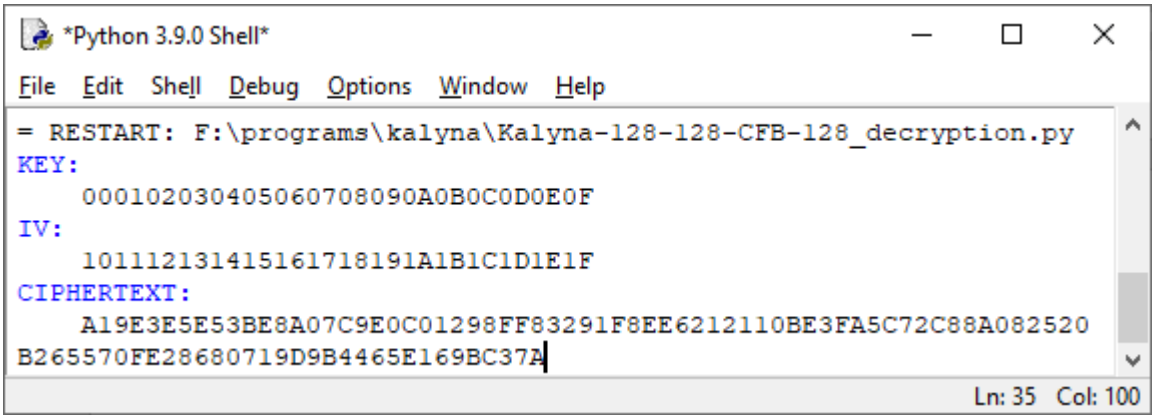

Рисунок 4.3 – Введення вхідних даних для дешифрування у режимі CFB

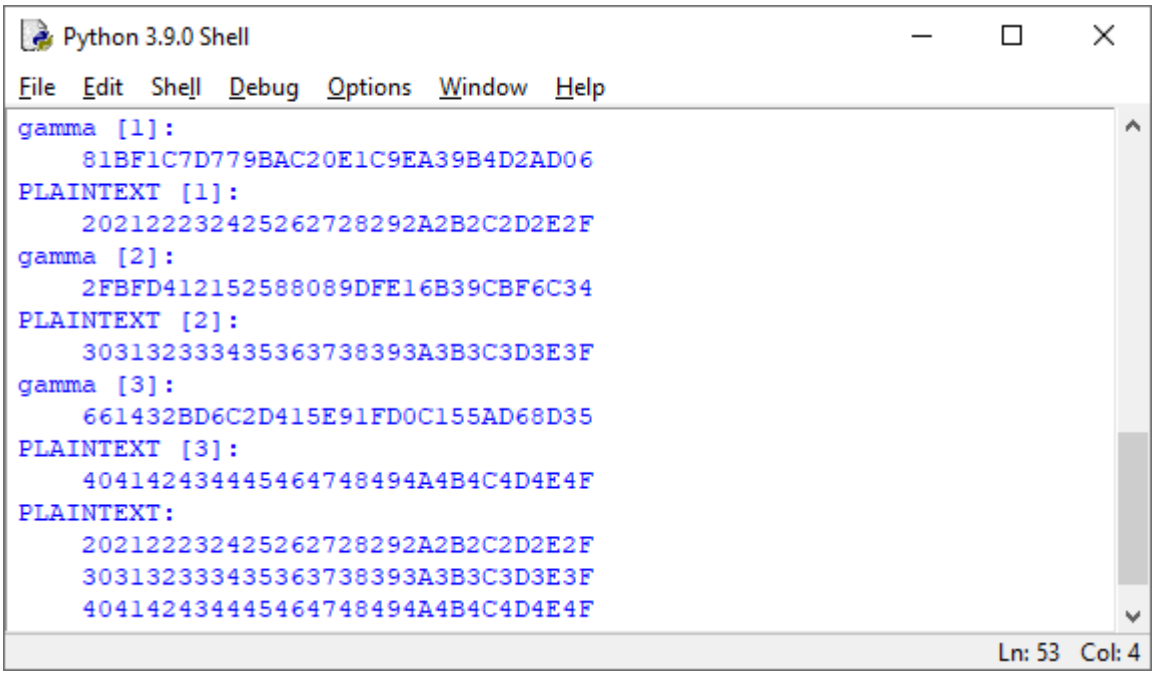

Рисунок 4.4 - Отримання результату для дешифрування у режимі CFB

Повний приклад результату роботи програми для дешифрування у режимі CFB подано у додатку Г.

4.4 Реалізація режиму захисту ключових даних

Було створено дві програми, що відтворюють криптоперетворення режиму захисту ключових даних для зашифрування та дешифрування відповідно.

Для відтворення базового перетворення застосовується програмний код, зазначений у підрозділі 4.2, отже для цих перетворень довжина блоків та ключа дорівнює 128 біт.

4.4.1 Програма для зашифрування у режимі KW/KW-p

Програма для зашифрування використовує в якості вхідних значень ключ шифрування та повідомлення (відкритий текст, який потрібно зашифрувати). Значення задаються у шістнадцятковій системі числення.

У програмі за необхідності застосовується алгоритм доповнення повідомлення, після цього відбувається зашифрування.

Результатом виконання програми є зашифроване повідомлення, відображене у шістнадцятковій системі числення.

Повний текст програми для зашифрування подано у додатку Д.

На рисунках 4.5-4.8 представлено знімки екрану виконання програми на етапі введення вхідних даних та на етапі завершення роботи з отриманням результату для варіантів без доповнення та з доповненням.

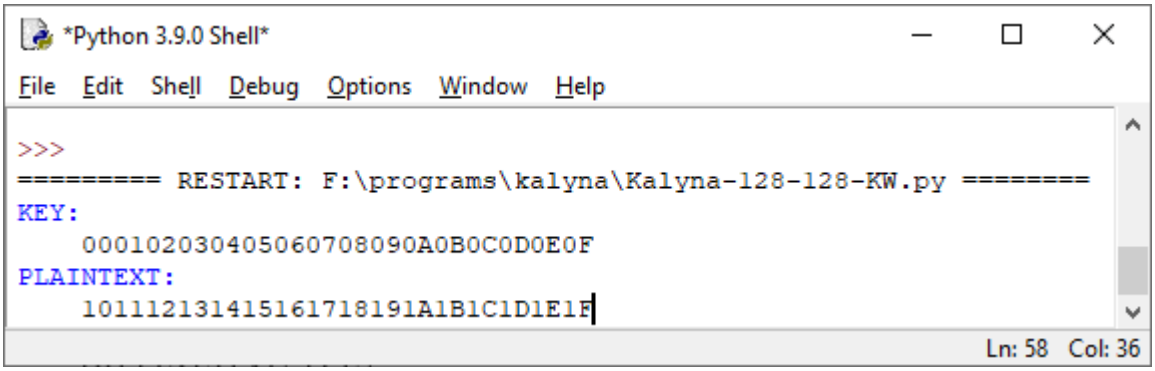

Рисунок 4.5 – Введення вхідних даних для зашифрування у режимі KW

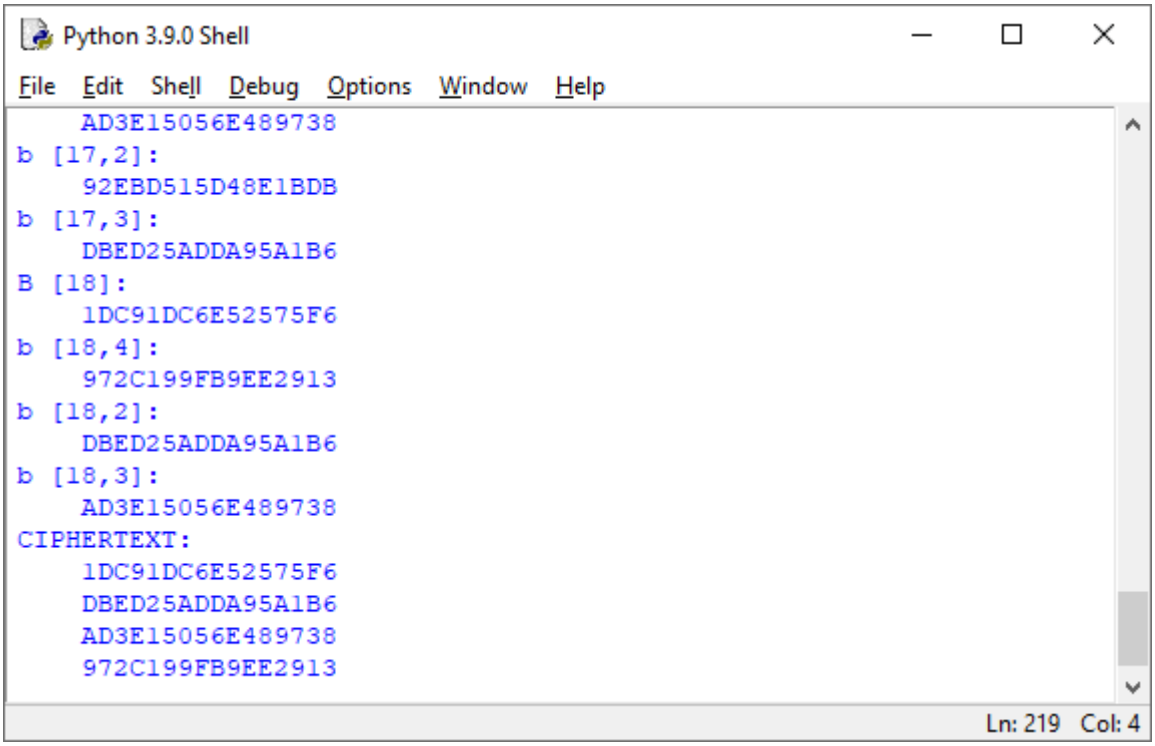

Рисунок 4.6 – Отримання результату для зашифрування у режимі KW

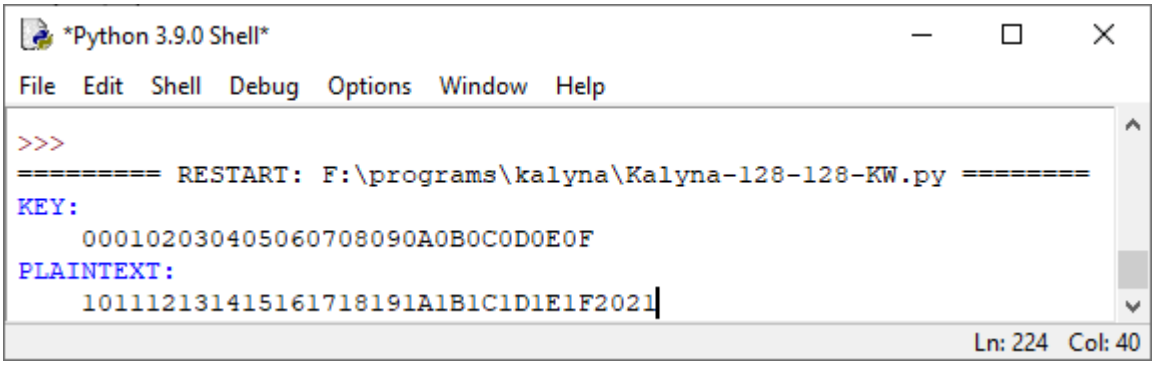

Рисунок 4.7 – Введення вхідних даних для зашифрування у режимі KW-p

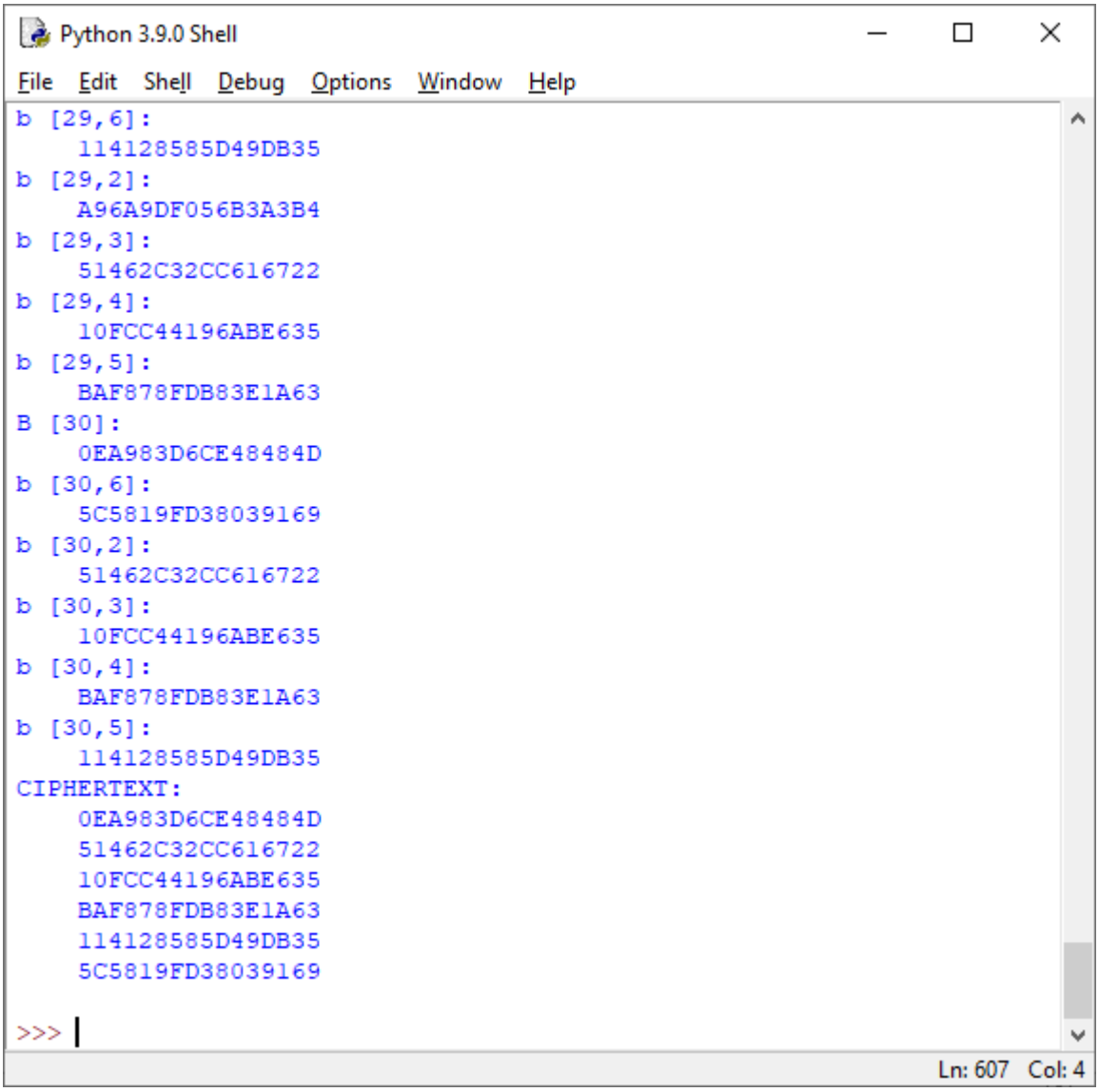

Рисунок 4.8 – Отримання результату для зашифрування у режимі KW-p

Повні приклади результатів роботи програм для зашифрування у режимі KW/KW-p подано у додатках Е-Ж.

4.4.2 Програма для дешифрування у режимі KW/KW-p

Програма для дешифрування використовує в якості вхідних значень ключ шифрування та зашифроване повідомлення (зашифрований текст, який потрібно розшифрувати). Значення задаються у шістнадцятковій системі числення.

Програма здійснює розшифрування заданого шифртексту, після чого за необхідності здійснює процедуру зняття доповнення.

Якщо результатом є помилка оберненого перетворення, подальша обробка припиняється із повідомленням про порушення цілісності.

Якщо перетворення було завершено успішно, результатом виконання програми є початкове повідомлення, відображене у шістнадцятковій системі числення.

Повний текст програми для дешифрування подано у додатку И.

На рисунках 4.9-4.12 представлено знімки екрану виконання програми на етапі введення вхідних даних та на етапі успішного завершення роботи з отриманням результату для варіантів без доповнення та з доповненням.

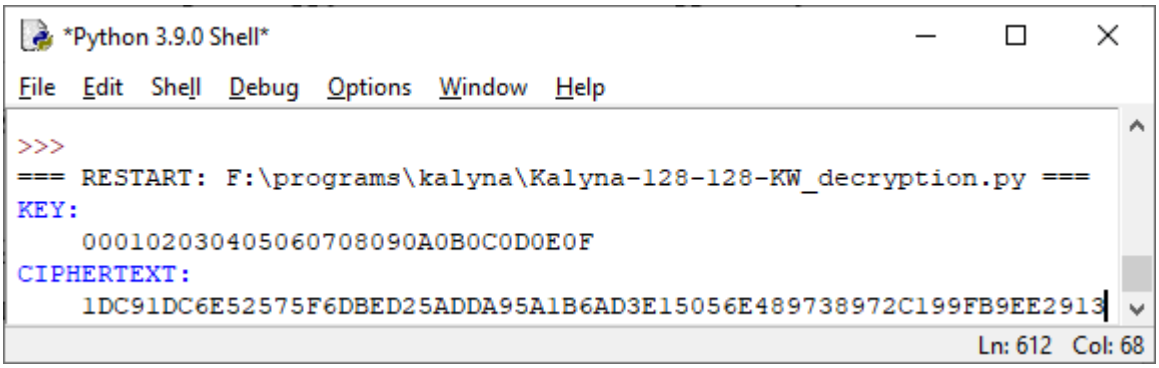

Рисунок 4.9 – Введення вхідних даних для дешифрування у режимі KW

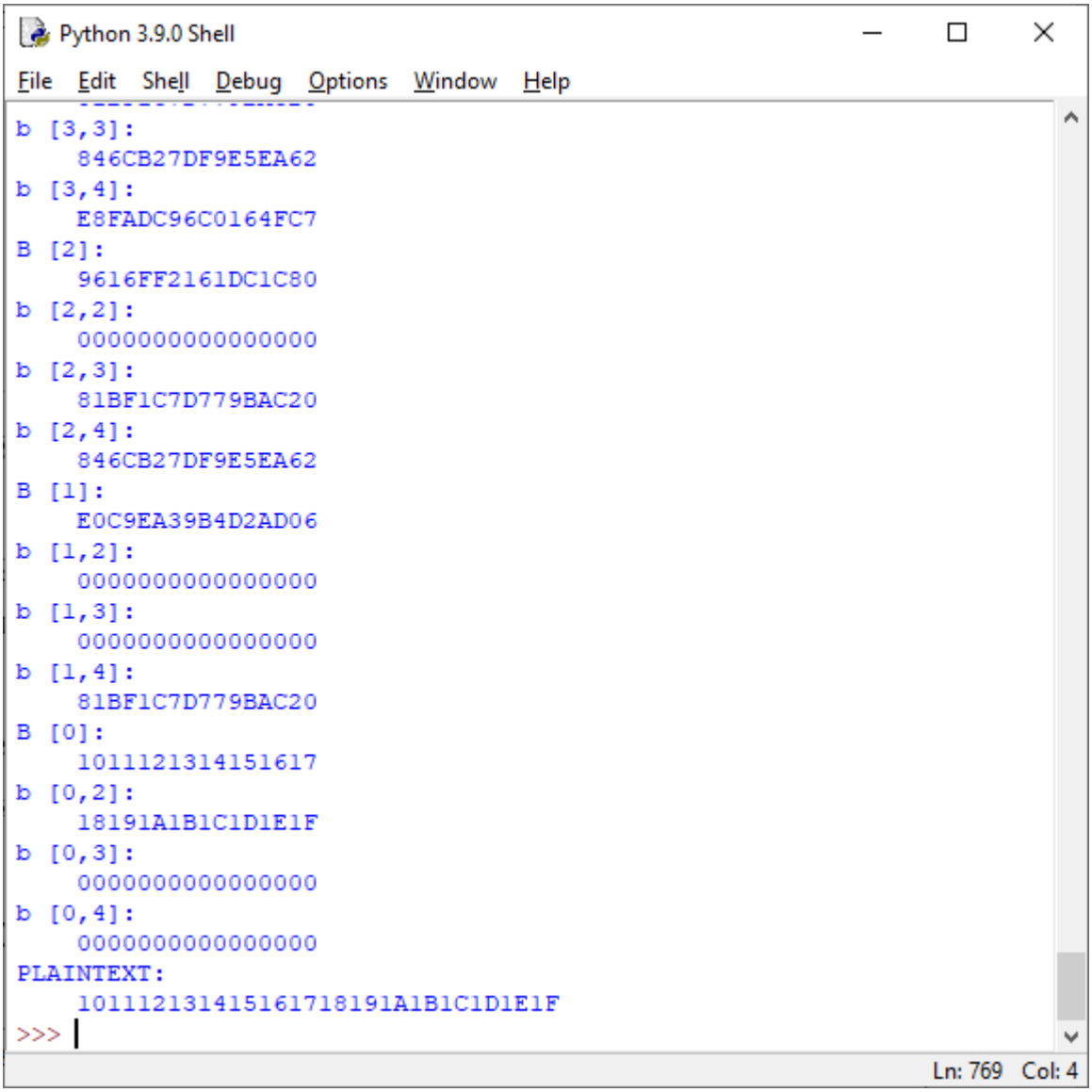

Рисунок 4.10 – Отримання результату для дешифрування у режимі KW

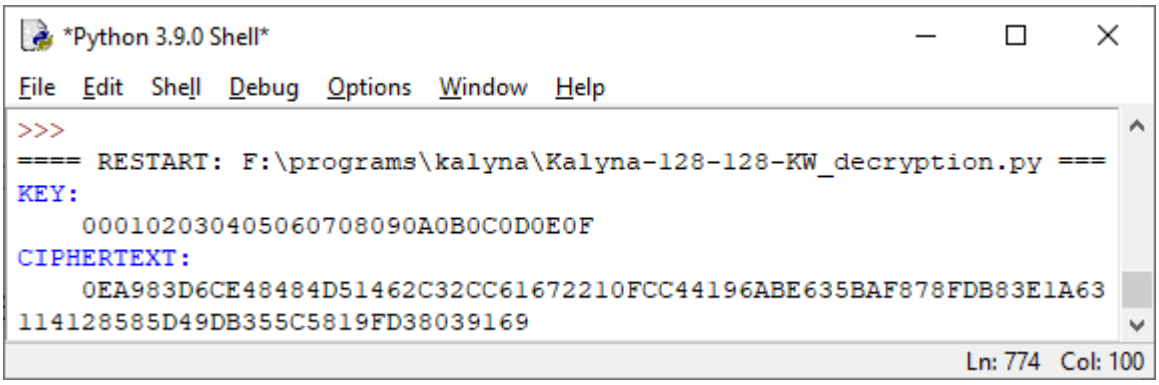

Рисунок 4.11 – Введення вхідних даних для дешифрування у режимі KW-p

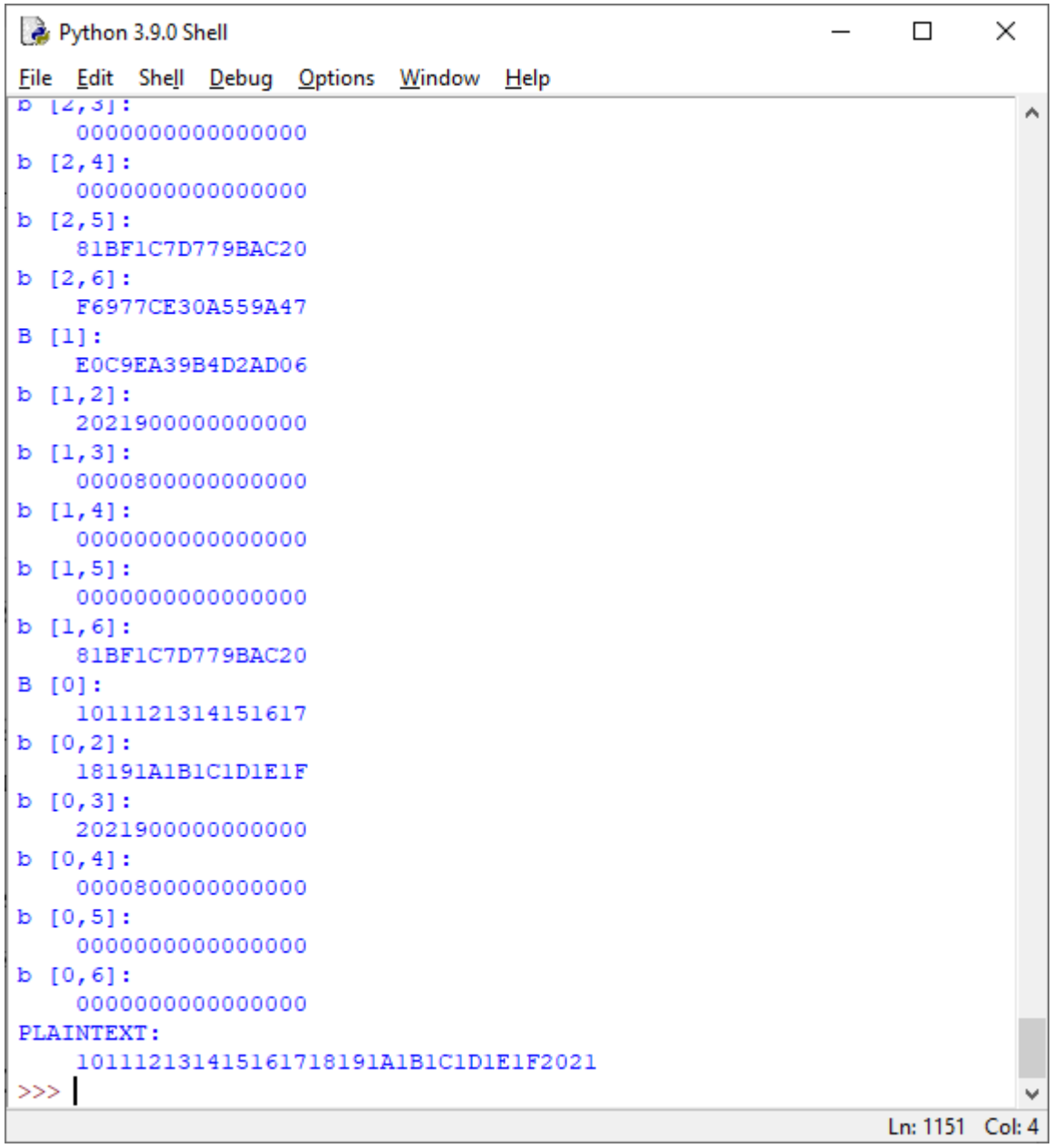

Рисунок 4.12 – Отримання результату для дешифрування у режимі KW-p

Повні приклади результатів роботи програм для дешифрування у режимі KW/KW-p подано у додатках К-Л.

ВИСНОВКИ

У роботі було розглянуто і проаналізовано алгоритми симетричного шифрування та їх режими.

Було розглянуто режими шифрування на основі національного стандарту ДСТУ 7624:2014 (Калина). З них було досліджено та проаналізовано:

– режим простої заміни (базове перетворення);

– режим гамування зі зворотнім зв’язком за шифртекстом;

– режим захисту ключових даних.

Для режиму гамування зі зворотнім зв’язком за шифртекстом та режиму захисту ключових даних було створено програмні реалізації алгоритмів з використанням відкритого коду для реалізації базового перетворення. Було створено програми для:

– зашифрування у режимі гамування зі зворотнім зв’язком за шифртекстом;

– дешифрування у режимі гамування зі зворотнім зв’язком за шифртекстом;

– зашифрування у режимі захисту ключових даних (без додатку та з додатком);

– дешифрування у режимі захисту ключових даних (без додатку та з додатком).

Для кожної з написаних програм було подано приклади її виконання (для режиму захисту ключових даних окремо варіанти без додатку та з додатком).

У підсумку, в дипломній роботі було досягнуто мету дослідження режимів шифрування у симетричних шифрах та їх програмної реалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гребенніков В.В. Історія криптології & секретного зв'язку / В.В. Гребенніков. – Ужгород: Ліра, 2012. – 664 с.

2. Щур Н.О. Основи криптології: навч. посібник / Н.О. Щур, О.А. Покотило. – Житомир: Державний університет «Житомирська політехніка», 2021. – 120 с.

3. Що таке криптографія з відкритим ключем? [Електронний ресурс] – Режим доступу: https://academy.binance.com/uk/articles/what-is-symmetric-key-cryptography

4. Козіна Г. Л. Криптографія від історії до сучасних стандартів: навч. посібник / Г. Л. Козіна. – Запоріжжя: НУ «Запорізька політехніка», 2020. – 192 с.

5. Diffie, W. and M. Hellman (1997). “Exhaustive cryptanalysis of the NBS data encryption standard.” Computer, 10 (6), 74–84.

6. A. Bogdanov, D. Khovratovich, and C. Rechberger, Biclique Cryptanalysis of the Full AES, Lecture Notes in Computer Science, vol.7073, pp.344-371, 2011.

7. Cryptography and Computer Privacy [Електронний ресурс] – Режим доступу: http://www.prism.net/user/dcowley/docs.html

8. Claude E. Shannon, "Communication Theory of Secrecy Systems" [Електронний ресурс] – Режим доступу: https:// netlab.cs.ucla.edu/wiki/files/shannon1949.pdf

9. Єфіменко А.А. Аналіз та порівняння алгоритму симетричного блокового перетворення «Калина» (ДСТУ 7624:2014) з міжнародним стандартом шифрування даних AES / Єфіменко А.А, Байлюк Є.М., Покотило Є.М. – Житомирський державний технологічний університет, 2018

10. Олійников Р.В. Принципи побудови і основні властивості нового національного стандарту блокового шифрування України / Олійников Р.В, Горбенко І.В, Казимиров О.В. та ін. – Захист інформації, 2015 – Т.1

11. ДСТУ 7624:2014. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. – Київ: Держстандарт України, 2014. – 238 с.

12. Олійников Р.В. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. Основні властивості. / Р.В. Олійников, І.В. Горбенко, О.В. Казимиров та ін. – Київ, 2015.

13. Python [Електронний ресурс] – Режим доступу: https://www.python.org/

14. Cryptography standards [Електронний ресурс] – Режим доступу: https://github.com/NeverWalkAloner/Cryptography-standards

ДОДАТОК А

Текст програми для зашифрування у режимі CFB

```
import dstu
def T(m,K):
    a=dstu.dstu2014(K)
    return a.encryption(m)
def Input():
    y=input()
    x=[0 for i in range (len(y)//2)]
    for i in range (0,len(y),2):
        x[i//2]=int(y[i]+y[i+1],16)
    return x
def output(x):
    for n in x:
        if (n//16==0):
            print(0, end='')
        print('%X'%(n), end='')
    print()

print("KEY:\n    ", end='')
K=Input()
print("IV:\n    ", end='')
S=Input()
print("PLAINTEXT:\n    ", end='')
M=Input()

n=len(M)//16+1
```

```
m=[[0 for j in range (16)] for i in range (n)]
for i in range (len(M)):
    m[i//16][i%16]=M[i]

c=[[0 for j in range (16)] for i in range (n)]
c0=[[0 for j in range (16)] for i in range (n-1)]

c0[0]=T(S,K)
print("gamma [1]:\n    ", end='')
output(c0[0])
for i in range (1,n):
    for j in range (16):
        c[i][j]=m[i-1][j]^c0[i-1][j]
    print("CIPHERTEXT [",i,"]:\n    ", sep='', end='')
    output(c[i])
    if (i<n-1):
        c0[i]=T(c[i],K)
        print("gamma [",i+1,"]:\n    ", sep='', end='')
        output(c0[i])

print("CIPHERTEXT:\n    ", end='')
for i in range (1,n):
    output(c[i])
    print("    ", end='')
```

ДОДАТОК Б

Приклад роботи програми для зашифрування у режимі CFB

KEY:

000102030405060708090A0B0C0D0E0F

IV:

101112131415161718191A1B1C1D1E1F

PLAINTEXT:

202122232425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F

gamma [1]:

81BF1C7D779BAC20E1C9EA39B4D2AD06

CIPHERTEXT [1]:

A19E3E5E53BE8A07C9E0C01298FF8329

gamma [2]:

2FBFD412152588089DFE16B39CBF6C34

CIPHERTEXT [2]:

1F8EE6212110BE3FA5C72C88A082520B

gamma [3]:

661432BD6C2D415E91FD0C155AD68D35

CIPHERTEXT [3]:

265570FE28680719D9B4465E169BC37A

CIPHERTEXT:

A19E3E5E53BE8A07C9E0C01298FF8329

1F8EE6212110BE3FA5C72C88A082520B

265570FE28680719D9B4465E169BC37A

ДОДАТОК В

Текст програми для дешифрування у режимі CFB

```
import dstu
def T(m,K):
    a=dstu.dstu2014(K)
    return a.encryption(m)
def Input():
    y=input()
    x=[0 for i in range (len(y)//2)]
    for i in range (0,len(y),2):
        x[i//2]=int(y[i]+y[i+1],16)
    return x
def output(x):
    for n in x:
        if (n//16==0):
            print(0, end='')
        print('%X'%(n), end='')
    print()

print("KEY:\n    ", end='')
K=Input()
print("IV:\n    ", end='')
S=Input()
print("CIPHERTEXT:\n    ", end='')
C=Input()

n=len(C)//16+1
```

```
c=[[0 for j in range (16)] for i in range (n)]
for i in range (len(C)):
    c[i//16][i%16]=C[i]

m=[[0 for j in range (16)] for i in range (n)]
c0=[[0 for j in range (16)] for i in range (n-1)]

c0[0]=T(S,K)
print("gamma [1]:\n    ", end='')
output(c0[0])
for i in range (1,n):
    for j in range (16):
        m[i][j]=c[i-1][j]^c0[i-1][j]
    print("PLAINTEXT [",i,"]:\n    ", sep='', end='')
    output(m[i])
    if (i<n-1):
        c0[i]=T(c[i-1],K)
        print("gamma [",i+1,"]:\n    ", sep='', end='')
        output(c0[i])

print("PLAINTEXT:\n    ", end='')
for i in range (1,n):
    output(m[i])
    print("    ", end='')
```

ДОДАТОК Г

Приклад роботи програми для дешифрування у режимі CFB

KEY:

000102030405060708090A0B0C0D0E0F

IV:

101112131415161718191A1B1C1D1E1F

CIPHERTEXT:

A19E3E5E53BE8A07C9E0C01298FF83291F8EE6212110BE3FA5C72C88A08
2520B265570FE28680719D9B4465E169BC37A

gamma [1]:

81BF1C7D779BAC20E1C9EA39B4D2AD06

PLAINTEXT [1]:

202122232425262728292A2B2C2D2E2F

gamma [2]:

2FBFD412152588089DFE16B39CBF6C34

PLAINTEXT [2]:

303132333435363738393A3B3C3D3E3F

gamma [3]:

661432BD6C2D415E91FD0C155AD68D35

PLAINTEXT [3]:

404142434445464748494A4B4C4D4E4F

PLAINTEXT:

202122232425262728292A2B2C2D2E2F

303132333435363738393A3B3C3D3E3F

404142434445464748494A4B4C4D4E4F

ДОДАТОК Д

Текст програми для зашифрування у режимі KW/KW-p

```
import dstu
def T(m,K):
    a=dstu.dstu2014(K)
    return a.encryption(m)
def Input():
    y=input()
    x=[0 for i in range (len(y)//2)]
    for i in range (0,len(y),2):
        x[i//2]=int(y[i]+y[i+1],16)
    return x
def output(x):
    for n in x:
        if (n//16==0):
            print(0, end='')
        print('%X'%(n), end='')
    print()


print("KEY:\n    ", end='')
K=Input()
print("PLAINTEXT:\n    ", end='')
M=Input()
L=len(M)
n=2*(L//16+1)
if (L%16):
    n+=2
```

```
m=[[0 for j in range (8)] for i in range (n)]
for i in range (L):
    m[i//8][i%8]=M[i]
if (L%16):
    for i in range (len(M),len(M)+8):
        if (L>0):
            m[i//8][i%8]=(L%256)*8
            L//=256
        else:
            break
    m[(len(M)+8)//8][(len(M)+8)%8]=128

V=(n-1)*6+1
print("V:\n   ",V-1)

B=[[0 for j in range (8)] for i in range (V)]
b=[[[0 for k in range (8)] for j in range (n)] for i in range (V)]

B[0]=[m[0][i] for i in range (8)]
print("B [0]:\n    ", end='')
output(B[0])
for i in range (1,n):
    b[0][i]=[m[i][j] for j in range (8)]
    print("b [0,",i+1,"]:\n    ",sep='', end='')
    output(b[0][i])

for j in range (1,V):
    t=T(B[j-1]+b[j-1][1],K)
    B[j]=[t[i+8] for i in range (8)]
```

```
    B[j][0]^=j
    print("B [",j,"]:\n    ", sep='', end='')
    output(B[j])
    b[j][n-1]=[t[i] for i in range (8)]
    print("b [",j,",",n,"]:\n    ",sep='', end='')
    output(b[j][n-1])
    for i in range (1,n-1):
        b[j][i]=b[j-1][i+1]
        print("b [",j,",",i+1,"]:\n    ",sep='', end='')
        output(b[j][i])

C=B[V-1]
print("CIPHERTEXT:\n    ", end='')
output(C)
print("    ", end='')
for i in range (1,n):
    output(b[V-1][i])
    print("    ", end='')
```

ДОДАТОК Е

Приклад роботи програми для зашифрування у режимі KW

KEY:

000102030405060708090A0B0C0D0E0F

PLAINTEXT:

101112131415161718191A1B1C1D1E1F

V:

18

B [0]:

1011121314151617

b [0,2]:

18191A1B1C1D1E1F

b [0,3]:

0000000000000000

b [0,4]:

0000000000000000

B [1]:

E0C9EA39B4D2AD06

b [1,4]:

81BF1C7D779BAC20

b [1,2]:

0000000000000000

b [1,3]:

0000000000000000

B [2]:

9616FF2161DC1C80

b [2,4]:

846CB27DF9E5EA62

b [2,2]:

0000000000000000

b [2,3]:

81BF1C7D779BAC20

B [3]:

16D4D8040BAB9770

b [3,4]:

E8FADC96C0164FC7

b [3,2]:

81BF1C7D779BAC20

b [3,3]:

846CB27DF9E5EA62

B [4]:

2FA7BABA8BEF881E

b [4,4]:

97641AE1751E3668

b [4,2]:

846CB27DF9E5EA62

b [4,3]:

E8FADC96C0164FC7

B [5]:

46287D9B32DB6EED

b [5,4]:

E39D6755CA9909FA

b [5,2]:

E8FADC96C0164FC7

b [5,3]:

97641AE1751E3668

B [6]:

36C7CBAA83163361

b [6,4]:

8C3D01910930FE9D

b [6,2]:

97641AE1751E3668

b [6,3]:

E39D6755CA9909FA

B [7]:

30B3F2DA71B0794A

b [7,4]:

2F548F690FAACF71

b [7,2]:

E39D6755CA9909FA

b [7,3]:

8C3D01910930FE9D

B [8]:

6683C7C62F4C11D9

b [8,4]:

94C3DD624FDC7F14

b [8,2]:

8C3D01910930FE9D

b [8,3]:

2F548F690FAACF71

B [9]:

2C70C839E442B4F4

b [9,4]:

8BC00C8CE891058E

b [9,2]:

2F548F690FAACF71

b [9,3]:

94C3DD624FDC7F14

B [10]:

38F627141D055364

b [10,4]:

02A65B3741515391

b [10,2]:

94C3DD624FDC7F14

b [10,3]:

8BC00C8CE891058E

B [11]:

55FB3C5537D958A5

b [11,4]:

96A46AED96B9B634

b [11,2]:

8BC00C8CE891058E

b [11,3]:

02A65B3741515391

B [12]:

D0B8BCDBE47B86B2

b [12,4]:

7D1555B543A661ED

b [12,2]:

02A65B3741515391

b [12,3]:

96A46AED96B9B634

B [13]:

37F77355E52DECF1

b [13,4]:

8C547E14C1F348D4

b [13,2]:

96A46AED96B9B634

b [13,3]:

7D1555B543A661ED

B [14]:

A356546DE6BE1DA9

b [14,4]:

6AE0EBC1ED2B33E4

b [14,2]:

7D1555B543A661ED

b [14,3]:

8C547E14C1F348D4

B [15]:

413E178CFBED9493

b [15,4]:

92EBD515D48E1BDB

b [15,2]:

8C547E14C1F348D4

b [15,3]:

6AE0EBC1ED2B33E4

B [16]:

E96F98BF552828A3

b [16,4]:

DBED25ADDA95A1B6

b [16,2]:

6AE0EBC1ED2B33E4

b [16,3]:

92EBD515D48E1BDB

B [17]:

871D4C7C47E05D4C

b [17,4]:

AD3E15056E489738

b [17,2]:

92EBD515D48E1BDB

b [17,3]:

DBED25ADDA95A1B6

B [18]:

1DC91DC6E52575F6

b [18,4]:

972C199FB9EE2913

b [18,2]:

DBED25ADDA95A1B6

b [18,3]:

AD3E15056E489738

CIPHERTEXT:

1DC91DC6E52575F6

DBED25ADDA95A1B6

AD3E15056E489738

972C199FB9EE2913

ДОДАТОК Ж

Приклад роботи програми для зашифрування у режимі KW-p

KEY:

000102030405060708090A0B0C0D0E0F

PLAINTEXT:

101112131415161718191A1B1C1D1E1F2021

V:

30

B [0]:

1011121314151617

b [0,2]:

18191A1B1C1D1E1F

b [0,3]:

2021900000000000

b [0,4]:

0000800000000000

b [0,5]:

0000000000000000

b [0,6]:

0000000000000000

B [1]:

E0C9EA39B4D2AD06

b [1,6]:

81BF1C7D779BAC20

b [1,2]:

2021900000000000

b [1,3]:

0000800000000000

b [1,4]:

0000000000000000

b [1,5]:

0000000000000000

B [2]:

49E1B34F2252F725

b [2,6]:

F6977CE30A559A47

b [2,2]:

0000800000000000

b [2,3]:

0000000000000000

b [2,4]:

0000000000000000

b [2,5]:

81BF1C7D779BAC20

B [3]:

23F5AA5A742021D9

b [3,6]:

4579EEFF1114EA5C

b [3,2]:

0000000000000000

b [3,3]:

0000000000000000

b [3,4]:

81BF1C7D779BAC20

b [3,5]:

F6977CE30A559A47

B [4]:

51E814CC36FE61ED

b [4,6]:

BB69999C77DAE340

b [4,2]:

0000000000000000

b [4,3]:

81BF1C7D779BAC20

b [4,4]:

F6977CE30A559A47

b [4,5]:

4579EEFF1114EA5C

B [5]:

8CDACDD1879C719E

b [5,6]:

766917CBCF818775

b [5,2]:

81BF1C7D779BAC20

b [5,3]:

F6977CE30A559A47

b [5,4]:

4579EEFF1114EA5C

b [5,5]:

BB69999C77DAE340

B [6]:

F66A1F42869C478F

b [6,6]:

B69D32DE9BD1B443

b [6,2]:

F6977CE30A559A47

b [6,3]:

4579EEFF1114EA5C

b [6,4]:

BB69999C77DAE340

b [6,5]:

766917CBCF818775

B [7]:

C2A43362ECC5D96C

b [7,6]:

A038B49CDD27FC42

b [7,2]:

4579EEFF1114EA5C

b [7,3]:

BB69999C77DAE340

b [7,4]:

766917CBCF818775

b [7,5]:

B69D32DE9BD1B443

B [8]:

288402A774EAA7AB

b [8,6]:

7DC7C817C8FB2CCB

b [8,2]:

BB69999C77DAE340

b [8,3]:

766917CBCF818775

b [8,4]:

B69D32DE9BD1B443

b [8,5]:

A038B49CDD27FC42

B [9]:

4E34878D74038FE5

b [9,6]:

8AFF89CBC82A212C

b [9,2]:

766917CBCF818775

b [9,3]:

B69D32DE9BD1B443

b [9,4]:

A038B49CDD27FC42

b [9,5]:

7DC7C817C8FB2CCB

B [10]:

C75580E079751E68

b [10,6]:

1B5020A335FF3C19

b [10,2]:

B69D32DE9BD1B443

b [10,3]:

A038B49CDD27FC42

b [10,4]:

7DC7C817C8FB2CCB

b [10,5]:

8AFF89CBC82A212C

B [11]:

8BC289527FF39859

b [11,6]:

AC1914BE9BD3172F

b [11,2]:

A038B49CDD27FC42

b [11,3]:

7DC7C817C8FB2CCB

b [11,4]:

8AFF89CBC82A212C

b [11,5]:

1B5020A335FF3C19

B [12]:

96EE68BFD45379AE

b [12,6]:

3F47705E397060F2

b [12,2]:

7DC7C817C8FB2CCB

b [12,3]:

8AFF89CBC82A212C

b [12,4]:

1B5020A335FF3C19

b [12,5]:

AC1914BE9BD3172F

B [13]:

F9262A142B14AD02

b [13,6]:

393C439991A1A8AA

b [13,2]:

8AFF89CBC82A212C

b [13,3]:

1B5020A335FF3C19

b [13,4]:

AC1914BE9BD3172F

b [13,5]:

3F47705E397060F2

B [14]:

B913BE35711FAEC5

b [14,6]:

B762B0C88D553400

b [14,2]:

1B5020A335FF3C19

b [14,3]:

AC1914BE9BD3172F

b [14,4]:

3F47705E397060F2

b [14,5]:

393C439991A1A8AA

B [15]:

AA273AC09E311670

b [15,6]:

930D9357D70245DE

b [15,2]:

AC1914BE9BD3172F

b [15,3]:

3F47705E397060F2

b [15,4]:

393C439991A1A8AA

b [15,5]:

B762B0C88D553400

B [16]:

3F259D57242C2E54

b [16,6]:

63CFC1FA25F3A9F0

b [16,2]:

3F47705E397060F2

b [16,3]:

393C439991A1A8AA

b [16,4]:

B762B0C88D553400

b [16,5]:

930D9357D70245DE

B [17]:

36882C52ECFF1254

b [17,6]:

5625CD1C29CC5E87

b [17,2]:

393C439991A1A8AA

b [17,3]:

B762B0C88D553400

b [17,4]:

930D9357D70245DE

b [17,5]:

63CFC1FA25F3A9F0

B [18]:

87686F0E18E5E42E

b [18,6]:

CDE7D2E418AB0552

b [18,2]:

B762B0C88D553400

b [18,3]:
930D9357D70245DE
b [18,4]:
63CFC1FA25F3A9F0
b [18,5]:
5625CD1C29CC5E87
B [19]:
2278D847C42DA539
b [19,6]:
78A975308CA46DC8
b [19,2]:
930D9357D70245DE
b [19,3]:
63CFC1FA25F3A9F0
b [19,4]:
5625CD1C29CC5E87
b [19,5]:
CDE7D2E418AB0552
B [20]:
2EEEC05489E18B27
b [20,6]:
9A7067BA70B830D0
b [20,2]:
63CFC1FA25F3A9F0
b [20,3]:
5625CD1C29CC5E87
b [20,4]:
CDE7D2E418AB0552
b [20,5]:

78A975308CA46DC8

B [21]:

8BA4431022F35CE3

b [21,6]:

3634D54C50735C6D

b [21,2]:

5625CD1C29CC5E87

b [21,3]:

CDE7D2E418AB0552

b [21,4]:

78A975308CA46DC8

b [21,5]:

9A7067BA70B830D0

B [22]:

E511ECFBCD9E0AD1

b [22,6]:

27B591C258997200

b [22,2]:

CDE7D2E418AB0552

b [22,3]:

78A975308CA46DC8

b [22,4]:

9A7067BA70B830D0

b [22,5]:

3634D54C50735C6D

B [23]:

09F8138B09720CE5

b [23,6]:

FA6BC7D20275319C

b [23,2]:
78A975308CA46DC8
b [23,3]:
9A7067BA70B830D0
b [23,4]:
3634D54C50735C6D
b [23,5]:
27B591C258997200
B [24]:
29A46BEF35CD5393
b [24,6]:
968670153F639760
b [24,2]:
9A7067BA70B830D0
b [24,3]:
3634D54C50735C6D
b [24,4]:
27B591C258997200
b [24,5]:
FA6BC7D20275319C
B [25]:
286B8E1658F91336
b [25,6]:
A96A9DF056B3A3B4
b [25,2]:
3634D54C50735C6D
b [25,3]:
27B591C258997200
b [25,4]:

FA6BC7D20275319C

b [25,5]:

968670153F639760

B [26]:

9CE7B10D4C473147

b [26,6]:

51462C32CC616722

b [26,2]:

27B591C258997200

b [26,3]:

FA6BC7D20275319C

b [26,4]:

968670153F639760

b [26,5]:

A96A9DF056B3A3B4

B [27]:

DE2DA537B021C3C0

b [27,6]:

10FCC44196ABE635

b [27,2]:

FA6BC7D20275319C

b [27,3]:

968670153F639760

b [27,4]:

A96A9DF056B3A3B4

b [27,5]:

51462C32CC616722

B [28]:

E9E55D5BDCA67862

b [28,6]:
BAF878FDB83E1A63
b [28,2]:
968670153F639760
b [28,3]:
A96A9DF056B3A3B4
b [28,4]:
51462C32CC616722
b [28,5]:
10FCC44196ABE635
B [29]:
7B6A2D9181C71169
b [29,6]:
114128585D49DB35
b [29,2]:
A96A9DF056B3A3B4
b [29,3]:
51462C32CC616722
b [29,4]:
10FCC44196ABE635
b [29,5]:
BAF878FDB83E1A63
B [30]:
0EA983D6CE48484D
b [30,6]:
5C5819FD38039169
b [30,2]:
51462C32CC616722
b [30,3]:

10FCC44196ABE635

b [30,4]:

BAF878FDB83E1A63

b [30,5]:

114128585D49DB35

CIPHERTEXT:

0EA983D6CE48484D

51462C32CC616722

10FCC44196ABE635

BAF878FDB83E1A63

114128585D49DB35

5C5819FD38039169

ДОДАТОК И

Текст програми для дешифрування у режимі KW/KW-p

```
import dstu
def U(c,K):
    a=dstu.dstu2014(K)
    return a.decryption©
def Input():
    y=input()
    x=[0 for i in range (len(y)//2)]
    for i in range (0,len(y),2):
        x[i//2]=int(y[i]+y[i+1],16)
    return x
def output(x):
    for n in x:
        if (n//16==0):
            print(0, end=’’)
        print(‘%X’%(n), end=’’)
    print()

print(«KEY:\n    «, end=’’)
K=Input()
print(«CIPHERTEXT:\n    «, end=’’)
C=Input()
n=2*(len©//16)
c=[[0 for j in range (8)] for i in range (n)]
for i in range (len©):
    c[i//8][i%8]=C[i]
```

```
V=(n-1)*6+1
print(«V:\n  «,V-1)

B=[[0 for j in range (8)] for i in range (V)]
b=[[[0 for k in range (8)] for j in range (n)] for i in range (V)]

B[V-1]=[c[0][i] for i in range (8)]
print(«B [«,V-1,»]:\n  «,sep='', end='')
output(B[V-1])
for i in range (1,n):
    b[V-1][i]=[c[i][j] for j in range (8)]
    print(«b [«,V-1,»,»,i+1,»]:\n  «,sep='', end='')
    output(b[V-1][i])

for j in range (V-1,0,-1):
    u=b[j][n-1]+B[j]
    u[8]^=j
    u=U(u,K)
    B[j-1]=[u[i] for i in range (8)]
    print(«B [«,j-1,»]:\n  «, sep='', end='')
    output(B[j-1])
    b[j-1][1]=[u[i+8] for i in range (8)]
    print(«b [«,j-1,»,2]:\n  «,sep='', end='')
    output(b[j-1][1])
    for i in range (1,n-1):
        b[j-1][i+1]=b[j][i]
        print(«b [«,j-1,»,»,i+2,»]:\n  «,sep='', end='')
        output(b[j-1][i+1])
```

```
M=B[0]
error=0
p=0
ln=0
for i in range(8):
    if (b[0][n-1][i]!=0):
        error=1
        break
for i in range(8):
    if (b[0][n-2][i]==128):
        p=1
        ln=i
        b[0][n-2][i]=0
        break
for i in range(8):
    if (b[0][n-2][i]!=0):
        error=1
        break
for i in range(8):
    if (b[0][n-3][i]==128):
        p=2
        ln=i
        b[0][n-3][i]=0
        break

if (error==1):
    print(«WRONG CIPHERTEXT»)
else:
```

```
for i in range (1,n-4):
    M+=b[0][i]
if (n==4):
    M+=b[0][1]
if (p==1):
    M+=b[0][n-4]+b[0][n-3][:ln]
else:
        M+=b[0][n-4][:ln]
print(«PLAINTEXT:\n    «, end=’’)
output(M)
```

ДОДАТОК К

Приклад роботи програми для дешифрування у режимі KW

KEY:

000102030405060708090A0B0C0D0E0F

CIPHERTEXT:

1DC91DC6E52575F6DBED25ADDA95A1B6AD3E15056E489738972C199FB9EE2913

V:

18

B [18]:

1DC91DC6E52575F6

b [18,2]:

DBED25ADDA95A1B6

b [18,3]:

AD3E15056E489738

b [18,4]:

972C199FB9EE2913

B [17]:

871D4C7C47E05D4C

b [17,2]:

92EBD515D48E1BDB

b [17,3]:

DBED25ADDA95A1B6

b [17,4]:

AD3E15056E489738

B [16]:

E96F98BF552828A3

b [16,2]:
6AE0EBC1ED2B33E4
b [16,3]:
92EBD515D48E1BDB
b [16,4]:
DBED25ADDA95A1B6
B [15]:
413E178CFBED9493
b [15,2]:
8C547E14C1F348D4
b [15,3]:
6AE0EBC1ED2B33E4
b [15,4]:
92EBD515D48E1BDB
B [14]:
A356546DE6BE1DA9
b [14,2]:
7D1555B543A661ED
b [14,3]:
8C547E14C1F348D4
b [14,4]:
6AE0EBC1ED2B33E4
B [13]:
37F77355E52DECF1
b [13,2]:
96A46AED96B9B634
b [13,3]:
7D1555B543A661ED
b [13,4]:

8C547E14C1F348D4

B [12]:

D0B8BCDBE47B86B2

b [12,2]:

02A65B3741515391

b [12,3]:

96A46AED96B9B634

b [12,4]:

7D1555B543A661ED

B [11]:

55FB3C5537D958A5

b [11,2]:

8BC00C8CE891058E

b [11,3]:

02A65B3741515391

b [11,4]:

96A46AED96B9B634

B [10]:

38F627141D055364

b [10,2]:

94C3DD624FDC7F14

b [10,3]:

8BC00C8CE891058E

b [10,4]:

02A65B3741515391

B [9]:

2C70C839E442B4F4

b [9,2]:

2F548F690FAACF71

b [9,3]:
94C3DD624FDC7F14
b [9,4]:
8BC00C8CE891058E
B [8]:
6683C7C62F4C11D9
b [8,2]:
8C3D01910930FE9D
b [8,3]:
2F548F690FAACF71
b [8,4]:
94C3DD624FDC7F14
B [7]:
30B3F2DA71B0794A
b [7,2]:
E39D6755CA9909FA
b [7,3]:
8C3D01910930FE9D
b [7,4]:
2F548F690FAACF71
B [6]:
36C7CBAA83163361
b [6,2]:
97641AE1751E3668
b [6,3]:
E39D6755CA9909FA
b [6,4]:
8C3D01910930FE9D
B [5]:

46287D9B32DB6EED

b [5,2]:

E8FADC96C0164FC7

b [5,3]:

97641AE1751E3668

b [5,4]:

E39D6755CA9909FA

B [4]:

2FA7BABA8BEF881E

b [4,2]:

846CB27DF9E5EA62

b [4,3]:

E8FADC96C0164FC7

b [4,4]:

97641AE1751E3668

B [3]:

16D4D8040BAB9770

b [3,2]:

81BF1C7D779BAC20

b [3,3]:

846CB27DF9E5EA62

b [3,4]:

E8FADC96C0164FC7

B [2]:

9616FF2161DC1C80

b [2,2]:

0000000000000000

b [2,3]:

81BF1C7D779BAC20

b [2,4]:

846CB27DF9E5EA62

B [1]:

E0C9EA39B4D2AD06

b [1,2]:

0000000000000000

b [1,3]:

0000000000000000

b [1,4]:

81BF1C7D779BAC20

B [0]:

1011121314151617

b [0,2]:

18191A1B1C1D1E1F

b [0,3]:

0000000000000000

b [0,4]:

0000000000000000

PLAINTEXT:

101112131415161718191A1B1C1D1E1F

ДОДАТОК Л

Приклад роботи програми для дешифрування у режимі KW-p

KEY:

000102030405060708090A0B0C0D0E0F

CIPHERTEXT:

0EA983D6CE48484D51462C32CC61672210FCC44196ABE635BAF878FDB83E1A6
3114128585D49DB355C5819FD38039169

V:

30

B [30]:

0EA983D6CE48484D

b [30,2]:

51462C32CC616722

b [30,3]:

10FCC44196ABE635

b [30,4]:

BAF878FDB83E1A63

b [30,5]:

114128585D49DB35

b [30,6]:

5C5819FD38039169

B [29]:

7B6A2D9181C71169

b [29,2]:

A96A9DF056B3A3B4

b [29,3]:

51462C32CC616722

b [29,4]:

10FCC44196ABE635

b [29,5]:

BAF878FDB83E1A63

b [29,6]:

114128585D49DB35

B [28]:

E9E55D5BDCA67862

b [28,2]:

968670153F639760

b [28,3]:

A96A9DF056B3A3B4

b [28,4]:

51462C32CC616722

b [28,5]:

10FCC44196ABE635

b [28,6]:

BAF878FDB83E1A63

B [27]:

DE2DA537B021C3C0

b [27,2]:

FA6BC7D20275319C

b [27,3]:

968670153F639760

b [27,4]:

A96A9DF056B3A3B4

b [27,5]:

51462C32CC616722

b [27,6]:
10FCC44196ABE635
B [26]:
9CE7B10D4C473147
b [26,2]:
27B591C258997200
b [26,3]:
FA6BC7D20275319C
b [26,4]:
968670153F639760
b [26,5]:
A96A9DF056B3A3B4
b [26,6]:
51462C32CC616722
B [25]:
286B8E1658F91336
b [25,2]:
3634D54C50735C6D
b [25,3]:
27B591C258997200
b [25,4]:
FA6BC7D20275319C
b [25,5]:
968670153F639760
b [25,6]:
A96A9DF056B3A3B4
B [24]:
29A46BEF35CD5393
b [24,2]:

9A7067BA70B830D0

b [24,3]:

3634D54C50735C6D

b [24,4]:

27B591C258997200

b [24,5]:

FA6BC7D20275319C

b [24,6]:

968670153F639760

B [23]:

09F8138B09720CE5

b [23,2]:

78A975308CA46DC8

b [23,3]:

9A7067BA70B830D0

b [23,4]:

3634D54C50735C6D

b [23,5]:

27B591C258997200

b [23,6]:

FA6BC7D20275319C

B [22]:

E511ECFBCD9E0AD1

b [22,2]:

CDE7D2E418AB0552

b [22,3]:

78A975308CA46DC8

b [22,4]:

9A7067BA70B830D0

b [22,5]:
3634D54C50735C6D
b [22,6]:
27B591C258997200
B [21]:
8BA4431022F35CE3
b [21,2]:
5625CD1C29CC5E87
b [21,3]:
CDE7D2E418AB0552
b [21,4]:
78A975308CA46DC8
b [21,5]:
9A7067BA70B830D0
b [21,6]:
3634D54C50735C6D
B [20]:
2EEEC05489E18B27
b [20,2]:
63CFC1FA25F3A9F0
b [20,3]:
5625CD1C29CC5E87
b [20,4]:
CDE7D2E418AB0552
b [20,5]:
78A975308CA46DC8
b [20,6]:
9A7067BA70B830D0
B [19]:

2278D847C42DA539

b [19,2]:

930D9357D70245DE

b [19,3]:

63CFC1FA25F3A9F0

b [19,4]:

5625CD1C29CC5E87

b [19,5]:

CDE7D2E418AB0552

b [19,6]:

78A975308CA46DC8

B [18]:

87686F0E18E5E42E

b [18,2]:

B762B0C88D553400

b [18,3]:

930D9357D70245DE

b [18,4]:

63CFC1FA25F3A9F0

b [18,5]:

5625CD1C29CC5E87

b [18,6]:

CDE7D2E418AB0552

B [17]:

36882C52ECFF1254

b [17,2]:

393C439991A1A8AA

b [17,3]:

B762B0C88D553400

b [17,4]:
930D9357D70245DE
b [17,5]:
63CFC1FA25F3A9F0
b [17,6]:
5625CD1C29CC5E87
B [16]:
3F259D57242C2E54
b [16,2]:
3F47705E397060F2
b [16,3]:
393C439991A1A8AA
b [16,4]:
B762B0C88D553400
b [16,5]:
930D9357D70245DE
b [16,6]:
63CFC1FA25F3A9F0
B [15]:
AA273AC09E311670
b [15,2]:
AC1914BE9BD3172F
b [15,3]:
3F47705E397060F2
b [15,4]:
393C439991A1A8AA
b [15,5]:
B762B0C88D553400
b [15,6]:

930D9357D70245DE

B [14]:

B913BE35711FAEC5

b [14,2]:

1B5020A335FF3C19

b [14,3]:

AC1914BE9BD3172F

b [14,4]:

3F47705E397060F2

b [14,5]:

393C439991A1A8AA

b [14,6]:

B762B0C88D553400

B [13]:

F9262A142B14AD02

b [13,2]:

8AFF89CBC82A212C

b [13,3]:

1B5020A335FF3C19

b [13,4]:

AC1914BE9BD3172F

b [13,5]:

3F47705E397060F2

b [13,6]:

393C439991A1A8AA

B [12]:

96EE68BFD45379AE

b [12,2]:

7DC7C817C8FB2CCB

b [12,3]:

8AFF89CBC82A212C

b [12,4]:

1B5020A335FF3C19

b [12,5]:

AC1914BE9BD3172F

b [12,6]:

3F47705E397060F2

B [11]:

8BC289527FF39859

b [11,2]:

A038B49CDD27FC42

b [11,3]:

7DC7C817C8FB2CCB

b [11,4]:

8AFF89CBC82A212C

b [11,5]:

1B5020A335FF3C19

b [11,6]:

AC1914BE9BD3172F

B [10]:

C75580E079751E68

b [10,2]:

B69D32DE9BD1B443

b [10,3]:

A038B49CDD27FC42

b [10,4]:

7DC7C817C8FB2CCB

b [10,5]:

8AFF89CBC82A212C

b [10,6]:

1B5020A335FF3C19

B [9]:

4E34878D74038FE5

b [9,2]:

766917CBCF818775

b [9,3]:

B69D32DE9BD1B443

b [9,4]:

A038B49CDD27FC42

b [9,5]:

7DC7C817C8FB2CCB

b [9,6]:

8AFF89CBC82A212C

B [8]:

288402A774EAA7AB

b [8,2]:

BB69999C77DAE340

b [8,3]:

766917CBCF818775

b [8,4]:

B69D32DE9BD1B443

b [8,5]:

A038B49CDD27FC42

b [8,6]:

7DC7C817C8FB2CCB

B [7]:

C2A43362ECC5D96C

b [7,2]:
4579EEFF1114EA5C
b [7,3]:
BB69999C77DAE340
b [7,4]:
766917CBCF818775
b [7,5]:
B69D32DE9BD1B443
b [7,6]:
A038B49CDD27FC42
B [6]:
F66A1F42869C478F
b [6,2]:
F6977CE30A559A47
b [6,3]:
4579EEFF1114EA5C
b [6,4]:
BB69999C77DAE340
b [6,5]:
766917CBCF818775
b [6,6]:
B69D32DE9BD1B443
B [5]:
8CDACDD1879C719E
b [5,2]:
81BF1C7D779BAC20
b [5,3]:
F6977CE30A559A47
b [5,4]:

4579EEFF1114EA5C

b [5,5]:

BB69999C77DAE340

b [5,6]:

766917CBCF818775

B [4]:

51E814CC36FE61ED

b [4,2]:

0000000000000000

b [4,3]:

81BF1C7D779BAC20

b [4,4]:

F6977CE30A559A47

b [4,5]:

4579EEFF1114EA5C

b [4,6]:

BB69999C77DAE340

B [3]:

23F5AA5A742021D9

b [3,2]:

0000000000000000

b [3,3]:

0000000000000000

b [3,4]:

81BF1C7D779BAC20

b [3,5]:

F6977CE30A559A47

b [3,6]:

4579EEFF1114EA5C

B [2]:

49E1B34F2252F725

b [2,2]:

0000800000000000

b [2,3]:

0000000000000000

b [2,4]:

0000000000000000

b [2,5]:

81BF1C7D779BAC20

b [2,6]:

F6977CE30A559A47

B [1]:

E0C9EA39B4D2AD06

b [1,2]:

2021900000000000

b [1,3]:

0000800000000000

b [1,4]:

0000000000000000

b [1,5]:

0000000000000000

b [1,6]:

81BF1C7D779BAC20

B [0]:

1011121314151617

b [0,2]:

18191A1B1C1D1E1F

b [0,3]:

2021900000000000

b [0,4]:

0000800000000000

b [0,5]:

0000000000000000

b [0,6]:

0000000000000000

PLAINTEXT:

101112131415161718191A1B1C1D1E1F2021