МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-28

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.28.15.000.КРБ

МУРЗІНА
ОЛЕКСАНДДРА
ВОЛОДИМОРОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: **«Комп'ютерна інженерія»**

Група: **2БКС-28**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи бакалавра на тему:

Дослідження блокових криптографічних систем шифрування інформації на основі мови Python

Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах).

Дипломник _____ (Мурзін О.В.)

Керівник _____ (Кільдішев В.Й)

Консультанти:

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувачка кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «24» 06 2024 р. Протокол ДК № 1

Оцінка ДК 4 (добре) 85 б

Секретар ДК _____

АНОТАЦІЯ

У випускній кваліфікаційній роботі на тему: "Дослідження блочних криптографічних систем шифрування інформації на мові програмування Python" розглянуто методи шифрування та режими блочного шифрування, також були розроблені блочні режими шифрування на мові програмування Python. Проаналізовано, як кожен з цих режимів впливає на безпеку та ефективність шифрування. На основі проведених досліджень було розроблено власні реалізації цих алгоритмів на Python, що дозволило зробити висновки щодо їх продуктивності та безпеки.

Було виконано аналіз переваг та недоліків блочних режимів шифрування. Було виконане зрівняння між Python, Java, C++ мовами програмування для виконання роботи. Крім того, у роботі були розроблені блок-схеми алгоритмів, що ілюструють процес шифрування та дешифрування для кожного з розглянутих режимів шифрування. Це допомогло візуалізувати складні процеси та полегшити їх розуміння. Загалом, виконана робота сприяє глибшому розумінню блочних криптографічних систем та їх реалізації на сучасних мовах програмування.

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС.28.16.000.00 КРБ. ПЗ	Кваліфікаційна робота		
А4			БКС.28.16.000.00 КРБ. ПЗ	Пояснювальна записка	1	
			БКС 28.16.000.00 КРБ МП	Мультимедійна		
				презентація		

Ізм.	Арк.	№ докум.	Підпис	Дата	БКС.28.16.000.00 КРБ. ПЗ			
Розробив		Мурзін О.В.			Дослідження блокових криптографічних систем шифрування інформації на основі мови програмування Python	Літ.	Аркуш	Аркушів
Перевірив		Кільдішев В.Й.				Н Д П	4	
						ВСП ОТФК ОНТУ 2БКС-28		
Н. Контр.		Петрашова В.І.		12.06				
Затверд.		Іванова Л. В.						

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **Комп'ютерних систем** Комісія ***КТ та ПІ***

Спеціальність **123 «Комп'ютерна інженерія»**

Освітня програма **«Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

"15" 01 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти **Мурзін Олександр Володимирович**

(прізвище, ім'я, по батькові)

1. Тема проекту **Дослідження блокових криптографічних систем шифрування інформації на основі мови програмування Python**

затверджена наказом по коледжу від "2" 11 2023 р. № 244-А2-ОД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту

Існуючи методи шифрування; Існуючи режими блочного шифрування та іх порівняння;

Реалізований код блочних режимів шифрування на мові програмування Python

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. ОСНОВНИЙ РОЗДІЛ. 2. Опис блочного метода шифрування. 3. Види атак на блочне шифрування. 4. Вибір мови програмування. 5. Опис функціонування коду. 6. Блок-схема алгоритмів режимів шифрування. 7. РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ. 8. ВИСНОВКИ 9. ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

1. Титулка. 2. Шифрування? 3. Що таке шифрування. 4. Методи шифрування. 5. Блочне шифрування. 6. Режими блочного шифрування. 7. Режим ECB. 8. Режим CBC. 9. Режим CFB. 10. Режим OFB. 11 Режим CRT 12. Таблиця зрівняння режимів блочного шифрування. 13. Реалізація ECB на мові програмування Python 14. Блок-схема алгоритм режима шифрування ECB реалізованих реалізованого на Python. 15. Кінець

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	Кільдішев В. Й.		
Розділ охорона праці	Чорновол. Н. І.		
Нормоконтроль	Петрашова В. І.		
Старший консультант	Кривченко Ю. В.		

7. Дата видачі завдання 15.01.2024

Керівник Кільдішев В. Й. _____
(Підпис)

Завдання прийняв до виконання Мурзін О. В. _____
(Підпис)

КАЛЕНДАРНИЙ ПЛАН

№ З/р	Назва етапів кваліфікаційної роботи	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	Огляд літератури. Огляд існуючих рішень.	04.05.2024	
2.	Консультація та створення питань до роботи	10.05.2024	
3.	ВСТУП. Формування кінцевого завдання на розробку	12.05.2024	
4.	Проаналізувати та порівняти існуючи режими блочного шифрування	16.05.2024	
5.	Розробка коду для режимів блочного шифрування	20.05.2024	
6.	Тестування коду для режимів шифрування	25.05.2024	
7.	Виконання розділу «Охорона праці»	29.05.2024	
8.	Перевірка роботи на плагіат	06.06.2024	
9.	Отримання рецензії, відповіді на зауваження рецензента	08.06.2024	
10	Підготовка до малого захисту	09.06.2024	
11	Підготовка доповіді та презентації для захисту	14.06.2024	

Дипломник _____
(підпис)

Керівник _____
(підпис)

ЗМІСТ

ВСТУП

З появою письмового збереження інформації з'явилася потреба шифрувати її. Перші приклади шифрування були знайдені в давніх манускриптах Індії та Єгипту. Шифрування неспішно розвивалось на протязі довгих століть, основою яких стали прості методи перестанови та підстановки символі. Важливим шагом в цьому напрямі стала використання ключа шифрування, який знайшов широке використання у епоху відродження. З появою механічних приладів, шифрування ще на крок просунулася вперед завдяки роторним машинам. Їх створили у XIX столітті, але широке використання отримали лише у XX ст. та використовувались до появи ЕОМ. На початку XX ст. криптографія стала, як самостійна наука, яка використовує математичні методи для шифрування.

В 1970-х роках IBM проводила дослідження у галузі шифрування, які очолював криптограф Хорст Фейстель. У 1973 році Фейстель розробив шифр Фейстеля, який став основою багатьох наступних блокових шифрів. Цей шифр поділяв кожний блок на дві частини та застосовує серію операцій к кожній частині, чергуючи їх між собою. Цей метод дозволив створити симетричний алгоритм шифрування, який чинив опір криптоаналізу.

У 1974 IBM представила шифр, заснований на структурі Фейстеля, який був прийнятий Національним інститутом стандартів і технологій США (NIST) в 1977 як Data Encryption Standard (DES).

1997 року NIST оголосив конкурс на розробку нового стандарту блокового шифрування, який мав замінити DES. У 2001 році було обрано алгоритм Rijndael, розроблений двома бельгійськими криптографами, Жоаном Дайменом та Вінсентом Рейменом. AES став новим стандартом та використовує блоки даних розміром 128 біт з ключами довжиною 128, 192 та 256 біт. AES відрізняється високою швидкістю та ефективністю на різних платформах, а також стійкістю до різних методів криптоаналізу.

В сучасному світі широко використовується шифрування інформації. Шифрування – це зворотне перетворення інформації, мета якого є захист цієї інформації від третіх осіб та передачі авторизованому користувачеві доступ до неї. Важливою особливістю любого алгоритму шифрування є використання ключа, який стверджує вибір конкретного перетворення з сукупності можливих для даного алгоритму.

Методи шифрування поділяються на: симетричні та асиметричні, метод хешування, гомоморфне шифрування та блочні, поточні шифрування

В цій роботі буде розглядатися методи блочного шифрування та зрівнюватися між собою. Це буде реалізовано на Python із-за ця мова найлегша для програмістів-початківців.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Загальний опис методів шифрування

1.1.1 Симетричний метод шифрування

Симетричний метод шифрування використовує один ключ для шифрування та розшифрування. Алгоритм та ключ обирається заздалегідь та відомий обом сторонам.

Принцип роботи симетричного шифрування заснований на математичних операціях, таких як заміни символів або перестановка бітів в повідомленні з відповідно з ключем. Цей процес робить дані незрозумілими для сторонніх осіб без знання ключа.

Одним з найпопулярніших алгоритмів симетричного шифрування є AES (Advanced Encryption Standard). AES використовує ключі різної довжини для розшифрування та шифрування даних. Він широко застосовується в різних областях, таких як шифрування файлів, захист даних в мережі і т. д.

У симетричного шифрування основною перевагою є його висока швидкість роботи, що робить його найкращим для шифрування великих об'ємів даних. Однак головний недолік полягає в необхідності передачі ключа меж відправником та одержувачем безпечним способом, що може представить складність в випадку передачі даних через відкриті мережі.

Також прикладами цього метода є також: DES (Data Encryption Standard) та IDEA (International Data Encryption Algorithm).

1.1.2 Асиметричний метод шифрування

Також відомий як криптографія з відкритим ключем – цей метод шифрування використовує два ключа: закритий та відкритий. Кожний учасник мережі має пару таких ключів. Відкритий ключ використовує для шифрування даних, а закритий ключ використовує для розшифрування.

Процес шифрування з окритим ключем має такий вигляд: відправник получає від отримувача його відкритий ключ та використовує його для шифрування повідомлення перед відправкою. Зашифроване повідомлення можу бути розшифроване тільки з допомогою відповідного закритого ключа, який знаходиться у отримувача. Таким чином, навіть якщо хто перехватить зашифроване повідомлення та має відкритий ключ, без закритого ключа він не зможе його розшифрувати.

Переваги асиметричного шифрування полягає в забезпечені безпечну передачу даних в відкритих мережах, не вимагаючи попереднього обміну секретними ключами. Однак він більш повільний та потребує більше обчислюваних ресурсів зі порівнянні з симетричним шифруванням.

Для цього метода прикладами є: RSA (Rivest-Shamir-Adleman) и ECC (Elliptic Curve Cryptography)

1.1.3 Метод Хешування

Хешування – це процес перетворення вхідних даних в фіксований набор символів та фіксований довжини, який називається хешем. Головна особливість хеш-функції полягає в том, що навіть невелика зміна у вхідних даних повинні приводить до значних змін в хеші. Вони роблять швидко, що забезпечують унікальність хешу для кожного унікального вхідного значення.

Хеш-функції незворотні, тому неможливо відновити вихідні дані з хешу. Вони роблять швидко, що забезпечує ефективність при обробці великих об’ємів даних.

Прикладом широкого використання алгоритмів хешування включають MD5 та SHA (Secure Hash Algorithm), такі як SHA-256, які забезпечують різні рівні безпеки та характеризуються різний довжиною хеша.

Хеш-функції використовуються для різних цілей, включаючи перевірку цілісності даних, збереження паролів та створення цифрових підписів. Однак

важливо вибирати підходящі алгоритми в залежності від конкретних потреб захисту та продуктивності.

1.1.4 Шифрування методом потока

Шифрування потоку: в цьому методі дані шифруються побайтно або побітно без розбиття на блоки. Кожний символ або біт відкритого тексту XOR-иться з символом або бітом ключа для отримання шифротексту. Це дозволить шифрувати та дешифрувати дані безперервним потоком, що особливо корисно для шифрування потокових даних, таких як передача звука або відео.

Прикладом алгоритмів шифрування блоків включають DES, AES та IDEA, та потокові шифри включають RC4 и ChaCha20.

1.1.5 Гомоморфне шифрування

Гомоморфне шифрування – це метод шифрування, який дозволяє виконувати операції над зашифрованими даними, не розшифровуючи їх. Це метод означає, що виконувати операції над зашифрованими даними та отримувати зашифрований результат, який буде відповідати результату операції над оригінальними даними.

Є два основних виду гомоморфного шифрування:

Повністю гомоморфне шифрування (FHE): Це тип гомоморфного шифрування який дозволяє виконувати любі операції над зашифрованими даними, включаючи додавання та множення. Однак (FHE) зазвичай вимагає значних обчислювальних ресурсів і може бути повільним у роботі.

Частково гомоморфне шифрування (PHE): Цей вид гомомрфного шифрування дозволяє виконувати тільки певні операції, наприклад: тільки додавання або тільки множення. (PHE) може бути більш ефективним з точки зору обчислень, ніж FHE, та мати менше потреби к ресурсам.

Гомоморфне шифрування має ряд потенційних застосувань, включаючи безпечну обробку конфіденційних даних в хвилі, забезпечення конфіденційності в обчислювальних процесах на стороні клієнта та забезпечуючи конфіденційності при обробці даних в медичних та фінансових додатках.

Однак гомоморфне шифрування також має свої обмеження та виклики, включаючи складність реалізації, потребу в обчислювальних ресурсах та можливі вразливості. Тим не менш, з розвитком криптографії та обчислювальної потужності існує надія на те, що гомоморфне шифрування стане більш практичним і широко застосовним у майбутньому.

1.1.6 Криптографічні ситеми

Також потрібно знати поняття **Криптографічних систем**

Криптографічні системи – це сукупність програмно-апаратних систем, та криптографічних алгоритмів або криптографічних схем, об’єднаних в єдину систему для розв’язання конкретної задачі захисту інформації. Криптографічні системи використовують для забезпечення конфіденційності, цілісності та автентичності даних. Вони відіграють важливу роль у захисті інформації в мережах передачі даних, електронній комерції, а також у різних аспектах інформаційної безпеки.

Є кілька основних типів криптосистем, наприклад: **Цифрові підписи**, вони використовуються для автентифікації відправника повідомлення та забезпечення цілісності та автентичності даних; **Протоколи обміну ключами**, ці протоколи дозволяють двом або більше сторонам безпечно домовитися про секретний ключ для використання у симетричному шифруванні; а також нове направлення **Квантова криптографія,** яке використовує принципи квантової механіки для забезпечення безпечного обміну ключами та захисту інформації від злому квантовими комп'ютерами.

1.2 Опис Блочного метода шифрування

Блочний метод шифрування даних розбирається на блоки фіксованого розміру, які шифруються окремо від одного. Одинакові блоки даних в одному повідомленні можуть бути зашифровані по різному, в залежності від використовуваного алгоритму в режимі шифрування. Режими шифрування визначаються, як блоки даних, об'єднуються та обробляються перед та після шифрування.

Популярні режими шифрування блоків включають:

- ECB (Electronic Codebook): Кожний блок даних шифрується окремо та незалежно. Це може привести к небажаним патернам в зашифрованих даних, якщо одним з тих же блоків даних мають однакові значення.
- CBC (Cipher Block Chaining): Кожний блок даних XOR-иться з попереднім зашифрованим блоком перед шифруванням. Це запобігає появу однакових шифротекстів для однакових блоків даних в повідомленні та робить "атаку" підміною більш важкою.
- CFB (Cipher Feedback): Шифротекст попереднього блока XOR-иться зі наступним блоком відкритого тексту перед шифруванням. Це дозволяє використовувати блочний шифр як потоковий шифр.
- OFB (Output Feedback): Генерує псевдовипадкова послідовність, яка XOR-иться зі окритим текстом для створення шифротексту. Це перетворює блочний шифр в потоковий.
- Counter (CTR): Кожний блок відкритого тексту XOR-иться з унікальним ключовим потоком, що генерується шляхом шифрування лічильника (counter) з використанням ключа. Цей режим як і OFB 0перетворює з блочного шифру в потоковий.

1.2.1 Опис режиму Electronic Codebook (ECB)

Цей режим є одним з найпростіших та основних режимів блочного шифрування. В цьому режимі кожний блок відкритого тексту шифрується окремо та незалежно, використовуючи один ключ.

У Electronic Codebook переваги є в простій реалізації, має можливість паралельно обробляти блоки, що прискорює шифрування та дешифрування, незалежність дає змогу ефективно використовувати багатопоточність. Недоліки цього режиму є слабкий захист інформації, при однаковому вхідному тексті створюються однакові шифротексти, це може призвести к появі видимих шаблонам в шифротексті. Також через незалежність блоків має вразливість к атакам на основі зміни блоків.

Приклад роботи:

Ми маємо текст: “HELLO WORLD”. Ми ділимо цей текст на блоки: “HELL” “O WO” “RLD”. На кінець блоки шифруються окремо з заданим ключем: Шифротекст 1 = Шифр (блок 1), Шифротекст 2 = Шифр (блок 2), Шифротекст 3 = Шифр (блок 3).

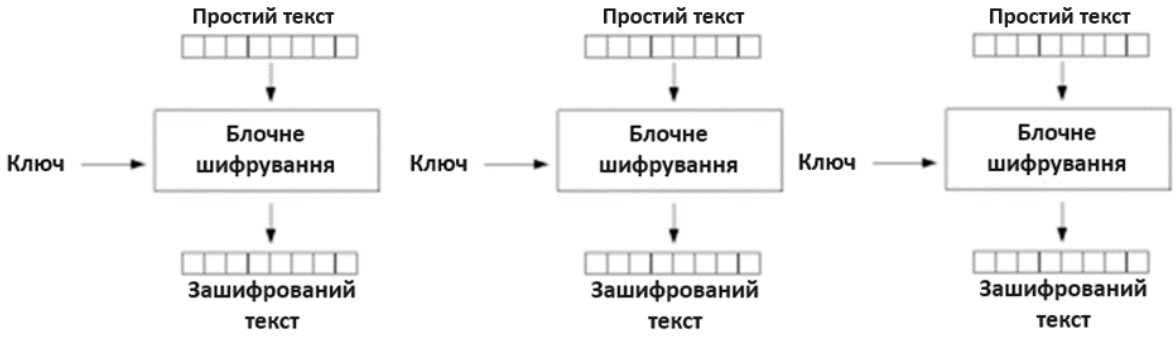


Рисунок 1.1. Приклад роботи ECB (шифрування)

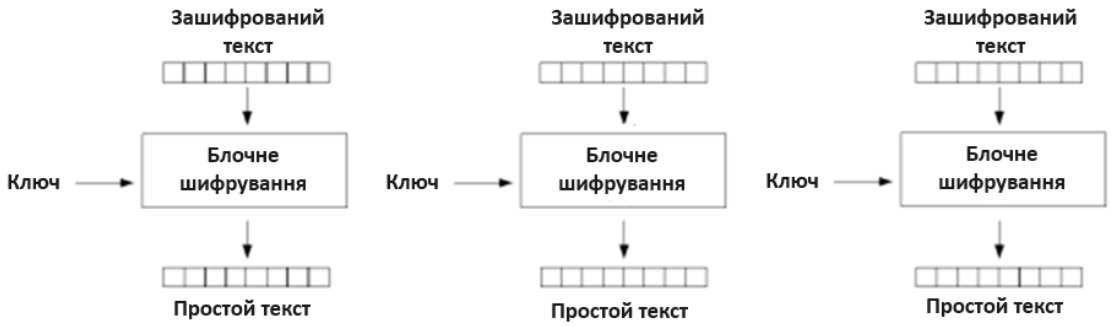


Рисунок 1.2. Приклад роботи ECB (розшифрування)

1.2.2 Опис режиму Cipher Block Chaining (CBC)

Режим Cipher Block Chaining забезпечує біль великий захист між Electronic Codebook, через використання вектора ініціалізації та XOR-операції.

Вектор ініціалізації (IV) – це випадкове або особливий вектор, який XOR-иться з першим блоком відкритого тексту перед шифруванням. Цій вектор також повинен бути відомий одержувачу, тому він часто передається з шифрованим текстом.

З переваг у CBC дає великий рівень безпеки через запобігання появі однакових шифротекстів для однакових блоків відкритого тексту. Також має надійність проти атак основаних на повторенні блоків, через використання XOR та залежності блоків, що створює більш складні патерн в шифротексті, утрудняючи аналіз та атак. Недоліками є потрібність в векторі для кожного повідомлення який має бути відомий відправнику та одержувачу, також повторне використання вектора призведе до послаблення безпеки. Має низьку ефективність при паралельній обробці, через залежність кожного блоку попереднього, також якщо з’явиться помилка в блоці шифротексту – це може вплинути на дешифровку наступних блоків.

Приклад роботи:

Маємо відкритий текст: "HELLO WORLD" Маємо вектор: випадковий чи унікальний вектор ініціалізації Ділимо текст на блоки: Блок 1: "HELL", Блок 2: "O WO", Блок 3: "RLD " Кожен блок обробляється XOR-м з IV потім шифрується:

Блок 1: XOR(IV, "HELL") -> Шифрується Блок 2: XOR(Зашифрований блок 1, "O WO") -> Шифрується Блок 3: XOR(Зашифрований блок 2, "RLD") -> Шифрується

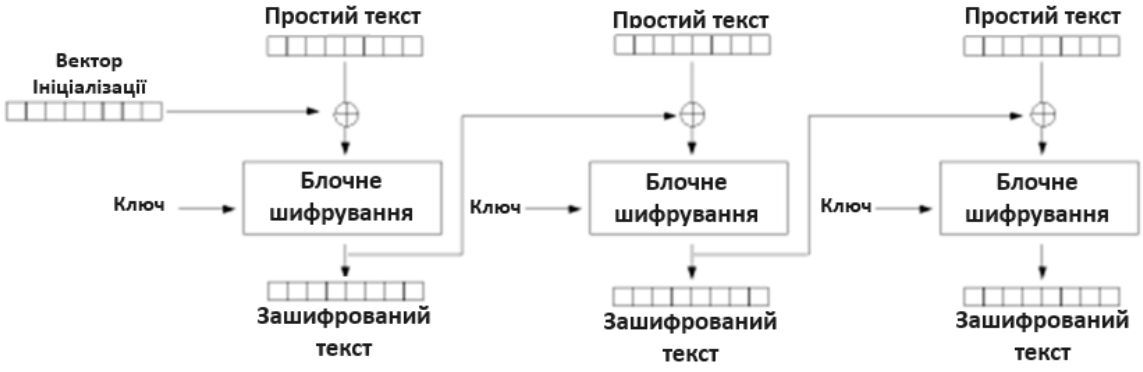


Рисунок 1.3. Приклад роботи CBC (шифрування)

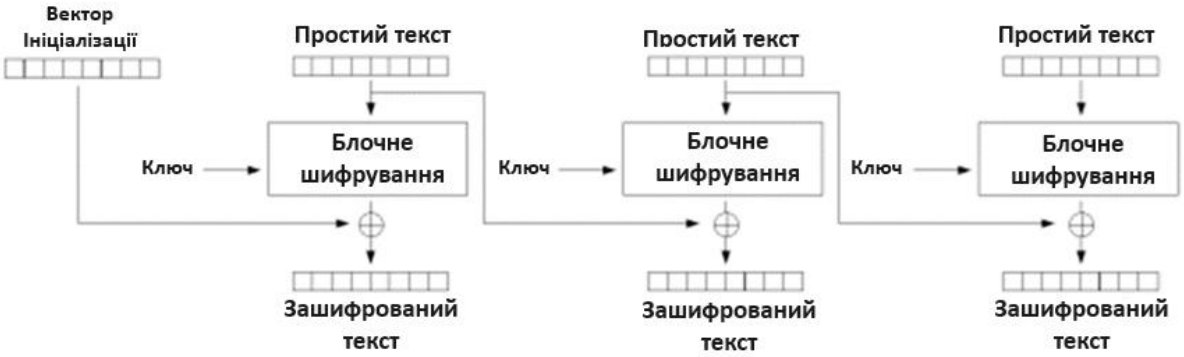


Рисунок 1.4. Приклад роботи CBC (розшифрування)

1.2.3 Опис режиму Cipher Feedback (CFB)

Cipher Feedback (CFB) – це режим, який дозволяє блочним шифрам робити як поточні шифри. Це робиться через використання шифротексту попереднього блоку для шифрування наступного блоку відкритого тексту. Він використовує векторну ініціалізацію з унікальним або випадковим для кожного шифротексту. Блоки достатньо цікаво шифруються, VI шифрується та результат XOR-иться з першим блоком відкритого тексту, щоб получити перший блок шифротексту. Потім кожний наступний блок шифрується з шифротекстом, результат XOR-иться з поточний блок відкритого тексту, щоб отримати поточний блок шифротексту. На кінець після кожного шифрування регістр зсувається, також останній блок шифротексту.

Перевагами Cipher Feedback є використання блочних шифрів для потокового шифрування даних, також підходить для передачі даних в у реальному часі. Недоліками як і у всіх режимах які використовують вектор це те, що унікальний вектор потрібен для кожного повідомлення та має бути відомий і відправнику і одержувач, та неправильне використовування вектора може послабити надійність. Також при помилці в блоці шифротексту може вплинути на подальше шифрування блоків до відновлення синхронізації.

Приклад роботи:

У нас є відкритий текст: “HELLO WORLD”; Створюється унікальний або випадковий вектор ініціалізації; Ділимо текст на блоки: Блок 1: "HELL", Блок 2: "O WO", Блок 3: "RLD "

На кінець блоки обробляються:

Шифруємо з вектором, потім XOR з "HELL" -> Шифротекст 1 Шифрування Шифротексту 1, потім XOR з "O WO" -> Шифротекст 2 Шифрування Шифротексту 2, потім XOR з "RLD" -> Шифротекст 3

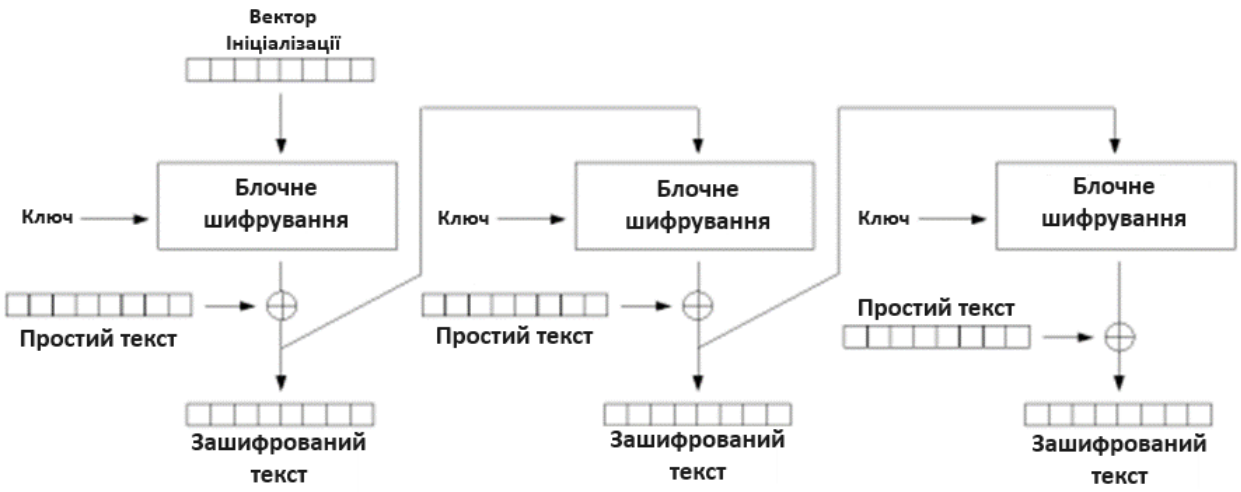


Рисунок 1.5. Приклад роботи CFB (шифрування)

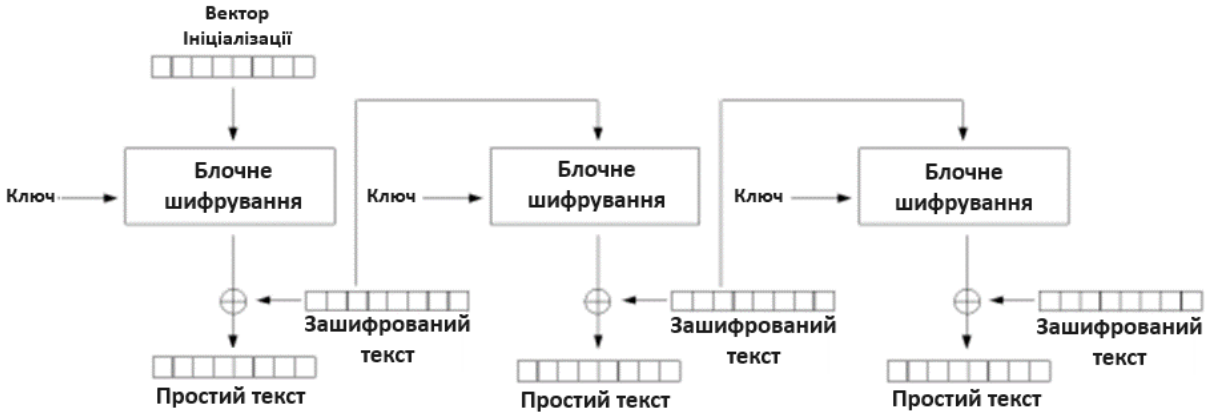


Рисунок 1.6. Приклад роботи CFB (розшифрування)

1.2.4 Опис режиму Output Feedback (OFB)

Output Feedback – цей режим блочного шифрування, який перетворює блочні шифри в поточні. Він забезпечує безпеку шифрування та дозволяє зашифрувати дані по блочно, як в блочних шифрах, но при цьому уникає кілька недоліків других режимів, таких як поява однакових шифротекстів для однакових блоків даних.

Спочатку вектор ініціалізації, який має бути унікальним або випадковим для кожного шифрування, при цьому вектор не повинен повторюватися для надійності. IV шифрується з використанням вибраного блочного шифру для отримання псевдовипадкового виходу (keystream block). Цей keystream block XOR-ится з першим блоком відкритого тексту для отримання першого блока шифротексту. В цьому режимі вихід попереднього шифрування використовується в якості входу для наступного шифрування, генеруючи наступний keystream block. Цей процес повторюється для кожного блоку відкритого тексту.

Перевагами цього режиму є: потокове шифрування, яке ґрунтується на перетворюванні блочних шифрів в потоковий, що робить його підходящим для шифрування потоків даних або даних невідомого розміру. При помилці в одному блоці шифротексту не впливає на розшифровку других блоків. В цьому плані OFB перевершує CBC, помилка не поширюється. OFB потрібен унікальний IV для кожного повідомлення, но IV може бути відкритим і передаватися разом з шифротекстом.

Недоліками є: повторне використання IV при шифруванні різні повідомлення можуть призвести до компрометації безпеки. Хоча у OFB помилка в одному блоці не впливає на інші блоки, але помилка в передачі може спотворити дані.

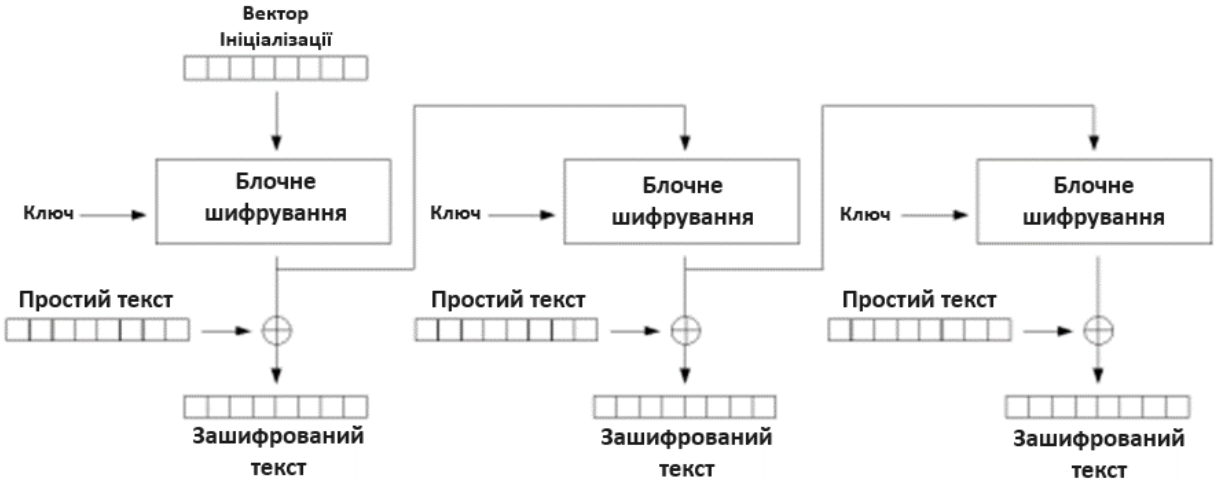


Рисунок 1.7. Приклад роботи OFB (шифрування)

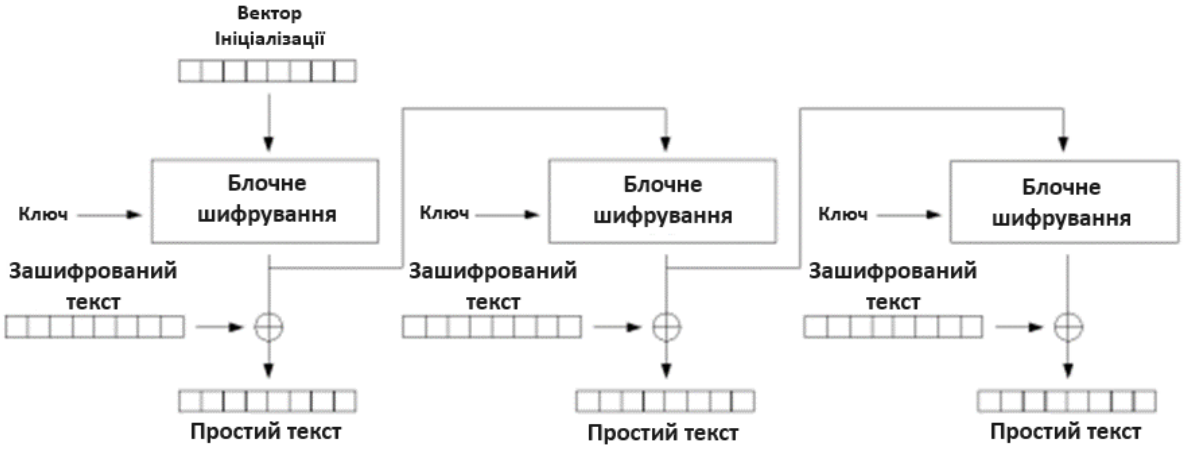


Рисунок 1.8. Приклад роботи OFB (розшифрування)

1.2.5 Опис режиму Counter (CTR)

Counter або режим лічильника – це один з режимів роботи блочних шифрів таких як AES. В цьому режимі блочний шифр перетворюється на потоковий шифр, що робить його ефективним та гнучким для шифрування даним любого розміру. Основними характеристиками цього режиму є: по-перше потоковий режим, який дозволяє блочним шифрам робити як поточні, генеруючи ключовий потік, який потім використовується для шифрування та розшифрування даних любого розміру, включно потоки даних. Також є паралельна обробка, в цьому режимі блоки даних можливо шифрувати та розшифровувати паралельно, що робить його підходящим для високопродуктивних додатків та багатоядерних процесорів.

Лічильник починається з деякого значення та збільшується для кожного наступного блоку. Тому він дає унікальність вектора ініціалізації для кожного блоку, що запобігає повторному використанню ключового потоку.

CTR режим забезпечує високу безпеку за умови, що лічильник ніколи не повторюється для того самого ключа, але повторне використання лічильника може призвести до вразливості атак.

Ще лічильник не вимагає складних механізмів зворотного зв'язку, що робить його простим у реалізації та перевірці.

Приклад роботи:

Спочатку лічильник ініціалізується початковим значенням (IV). Потім кожен блок лічильника шифрується з використанням блокового шифру та ключа для генерації ключового потоку. На кінець відкритий текст XOR-іється з ключовим потоком для отримання шифротексту, або для розшифрування шифротекст XOR-иться з тим же ключовим потоком для отримання відкритого тексту.

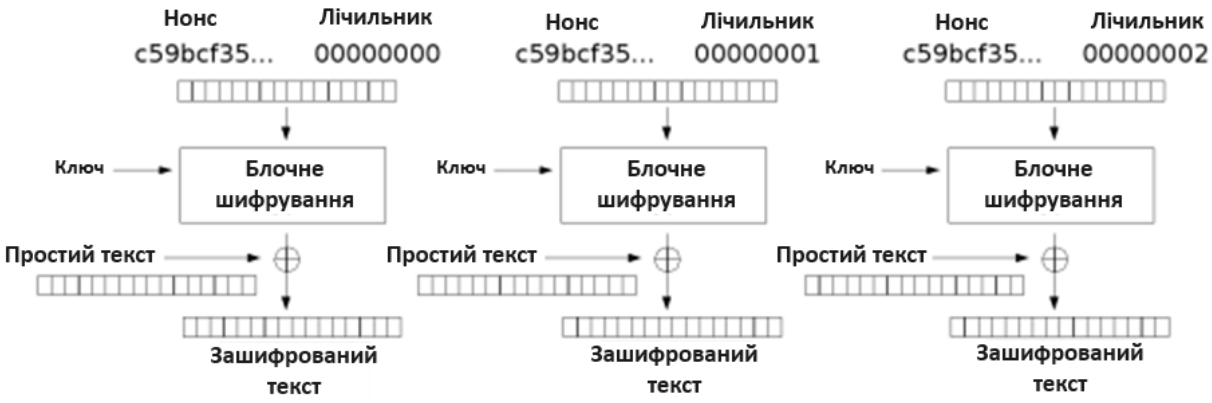


Рисунок 1.9. Приклад роботи CTR (шифрування)

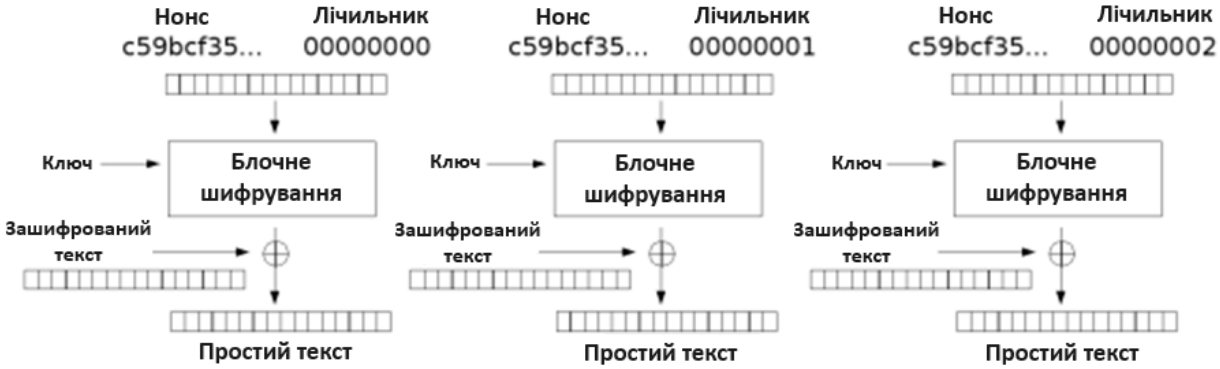


Рисунок 1.10. Приклад роботи CTR (розшифрування)

1.2.6 Зрівняння режимів блочного шифрування

Для цієї роботи потрібно зробити зрівняння між режимами блочного шифрування за такими характеристиками, а це: принцип роботи, паралельна обробка, обробка помилок, патерни в шифротексті, складність реалізації, необхідність IV, ініціалізація та за недоліками.

Таблиця 1.1. – Зрівняння режимів блочного шифрування

Характеристика	ECB (Electronic Codebook)	CBC (Cipher Block Chaining)	CFB (Cipher Feedback)	OFB (Output Feedback)	CTR (Counter)
Принцип роботи	Кожен блок шифрується незалежно	Кожен блок XOR-иться з попереднім шифротекстом перед шифруванням	Шифротекст попереднього блоку XOR-иться з наступним блоком відкритого тексту перед шифруванням	Ключовий потік генерується шифруванням вектора ініціалізації або зашифрованого попереднього блоку	Ключовий потік генерується шифруванням лічильника
Паралельная обработка	Так	Ні	Ні	Так	Так
Обробка помилок	Помилка впливає лише на один блок	Помилка поширюється на наступний блок	Помилка поширюється на наступний блок	Помилка не поширюється	Помилка не поширюється
Патерни в шифротексті	Може мати патерни за однакових блоків даних	Не має	Не має	Не має	Не має
Складність реалізації	Проста	Помірна	Помірна	Проста	Проста
Необхідність IV	Ні	Так	Так	Так	Так (nonce + лічильник)

Характеристика	ECB (Electronic Codebook)	CBC (Cipher Block Chaining)	CFB (Cipher Feedback)	OFB (Output Feedback)	CTR (Counter)
Ініціалізація	Не потребує	Потрібен випадковий IV	Потрібен випадковий IV	Потрібен випадковий IV	Потрібен випадковий nonce
Недоліки	Низька безпека через можливі патерни.	Складніше реалізація помилки поширюються на наступний блок	Складніше реалізація помилки поширюються на наступний блок	Можливі атаки на основі потоків, складність управління IV	Потрібен унікальний лічильник для кожного блоку, складність управління nonce

1.3 Види атак на блочне шифрування

Є різні види та методи шифрування, тому є і різні види атак для розшифрування (злому) цієї інформації. Для блочного шифрування можуть застосовані всі види атак, також є кілька специфічних. Ось кілька прикладів основних видів атак:

1. Аналітичні атаки, які поділяються на:
   - Диференціальний крипоаналіз – він досліджує, як відмінності у вхідних даних можуть впливати на відмінності у вихідних даних, щоб виявити ключ.
   - Лінійний криптоаналіз - цей використовує апроксимацію нелінійних операцій шифру лінійними рівняннями для виявлення ключа.
   - Інтегральний криптоаналіз - аналізує як певні набори вхідних даних призводять до певних вихідних даних, щоб знайти ключ.

2. Атаки методом грубої сили - це перебір усіх можливих ключів наприклад: випробовування всіх можливих комбінації ключів, поки не знайде правильний ключ. Практично неефективний для сучасних алгоритмів із великими довжинами ключів.
3. Алгебраїчні атаки - вони застосовуються до шифрів, які можуть бути описані рівняннями алгебри. Ці атаки намагаються вирішити такі рівняння для знаходження ключа.
4. Криптографічні атаки поділяються на:
    - Атаки які основані на структурі - вони використовує знання про внутрішню структуру шифру, такі як слабкості в S-блоках, схемах перестановок і підстановок.
    - Атаки пов'язані з ключами: Використовує кілька ключів, пов'язаних між собою будь-яким чином, щоб спростити завдання злому.
5. Атаки на основі вибору тексту поділяються на:
    - Відомий відкритий текст – зломщик який знає деякі пари відкритих текстів (шифротекст) та використовує цю інформацію для злому ключа.
    - Вибраний відкритий текст - атакуючий може вибрати відкритий текст та отримати відповідний шифротекст, що надає більше інформації для аналізу.
    - Вибраний шифротекст – в цій атаці, атакуючий може вибрати шифротекст та отримати відповідний відкритий текст, який допомагає в аналізі.
6. Атаки з витоком побічних каналів:
    - Атаки з використанням часу – основане на вимірюванні часу виконання криптографічних операцій, щоб отримати інформацію про ключ.
    - Електромагнітні атаки: Аналізують електромагнітне випромінювання пристрою під час виконання шифрування.

- Атаки споживаної потужності: Вимірюють споживану потужність пристрою під час шифрування, що може розкрити інформацію про ключ.

Ось кілька специфічних видів атак:

1. Атака зі зв'язними ключами (related-key attack) – це вид криптографічної атаки, в якій криптоаналітик вибирає зв'язок між парою ключів, але самі ключи залишаються йому невідомими. Ця атака має таку формулу: $K_B=F(K_A)$ де F – функція обрана атакуючим, K_B та K_A – зв'язані ключи. Якщо зрівнювати зі іншими атаками, в яких атаками в яких атакуючий може маніпулювати тільки відкритим шифротекстом або зашифрованим текстом, вибір між ключами дає додаткову ступінь свободи для зловмисника.
2. Атака з обраним ключем (chosen-key attack) — цей тип криптографічної атаки, при якій атакуючий може вибрати певні ключі для шифрування і спостерігати за результатами цих шифрувань. Основна мета таких атак – виявити слабкості у криптографічному алгоритмі, які залежать від особливостей використання різних ключів. Ці атаки зазвичай застосовуються до шифрів і систем, уразливих до специфічних залежностей між ключами та шифротекстом або між різними ключами. Має такий принцип роботи: зловмисник вибирає ключ, потім шифрує відкритий текст обраним ключем, та аналізує отриманий шифротекст, щоб виявити закономірність або слабкість, які допоможуть зламати шифр.
3. Усічений диференціальний криптоаналіз (truncated differential cryptanalysis) - це розширення класичного диференціального криптоаналізу, розроблений для аналізу блокових шифрів. На відміну від класичного диференціального криптоаналізу, який використовує точні

відмінності між парами вхідних і вихідних текстів, усічений диференціальний криптоаналіз працює з частковими відмінностями, ігноруючи деякі біти. Має такий принцип роботи: атакуючий вибирає пари відкритих текстів із певними частковими відмінностями. Ці відмінності можуть стосуватися лише деяких бітів або позицій усередині блоку. Потім ці пари текстів шифруються з використанням одного і того самого ключа, та аналізуються отримані шифротексти. На кінець досліджуються різниці між отриманими шифротекстами, щоб виявити закономірності. Якщо деякі часткові відмінності на вході призводять до певних часткових відмінностей на виході з високою ймовірністю, це використовується для атаки на шифр.

1.4 Вибір мови програмування

1.4.1 Опис Python

Python — це мова програмування високого рівня загального призначення, створена Гвідо ван Россумом і вперше випущена в 1991 році. Python набув широкої популярності завдяки своїй простоті, легкої читальності та гнучкості, що робить його ідеальним для різних завдань, від веб-розробки до наукових обчислень та штучного інтелекту.

Він має такі особливості:

1. Простоту та легку читаність - синтаксис Python інтуїтивно зрозумілий та легкий для читання, що сприяє швидкому вивченню та зручності написання коду.
2. Інтерпретована мова - Python інтерпретується, а не компілюється, що означає, що код виконується рядок за рядком. Це спрощує тестування та налагодження програм.

3. Велика стандартна бібліотека - Python поставляється з великою стандартною бібліотекою, яка включає модулі для роботи з файлами, мережами, інтернет-протоколами, базами даних та багатьма іншими завданнями.
4. Розширюваність - Python легко розширюється та інтегрується з іншими мовами, такими як C і C++, що дозволяє використовувати його як "клей" для зв'язування різних систем та компонентів.

Python має такі основні області застосування:

1. Веб-розробка – Python часто використовується для створення веб-застосунків. Фреймворки, такі як Django і Flask, спрощують розробку і дозволяють швидко створювати масштабовані та безпечні веб-сайти.
2. Наукові обчислення та аналіз даних – бібліотеки, такі як NumPy, SciPy, pandas та Matplotlib, роблять Python потужним інструментом для наукових досліджень та аналізу даних.
3. Ігрова розробка - бібліотеки, такі як Pygame, дозволяють створювати прості 2D-ігри та прототипи ігор.
4. Машинне навчання та штучний інтелект – Python є основною мовою для розробки алгоритмів машинного навчання та штучного інтелекту. Бібліотеки TensorFlow, Keras, PyTorch та Scikit-learn надають всі необхідні інструменти для цієї галузі.

1.4.2 Опис мови програмування C++

C++ - це високорівнева мова програмування загального призначення, розроблена Бйорном Страуструп в 1979 як доповнення до мови C. Він має в собі міць низькорівневого керування апаратним забезпеченням із зручністю високорівневого абстрактного програмування, що робить його одним з найпопулярніших та найбільш використовуваних мов програмування.

C++ має такі переваги над Python:

1. Має швидшу продуктивність: із-за того що він є компілюється мовою, що робить його значно швидше, ніж інтерпретований Python. Це особливо важливо для обчислювально-інтенсивних завдань, таких як криптографічні операції. Також має оптимізацію низького рівня, де: C++ дозволяє розробникам писати високоефективний код з використанням оптимізації низького рівня, таких як управління пам'яттю та асемблерні вставки.
2. Має прямий контроль над пам'яттю: C++ надає механізми для прямого управління пам'яттю (наприклад, за допомогою покажчиків), що може бути корисним для оптимізації криптографічних операцій. Також відсутність складання: у C++ немає автоматичного складання сміття, що дозволяє уникнути додаткових затримок, пов'язаних з керуванням пам'яттю, які можуть виникнути в Python.
3. Має паралелізм та багатопоточність, де: C++ має потужні засоби для багатопоточності та паралельних обчислень, що дозволяє ефективніше використовувати багатоядерні процесори. Має Реалізацію потокобезпечного коду може бути більш передбачуваною та ефективною у C++.

В прикладі буде використовуватись режим ECB, на C++, код має такий вигляд (див. Додаток А ст. 72).

1.4.3 Опис мови програмування Java

Java - це високорівнева мова програмування, розроблена в 1995 році компанією Sun Microsystem. Він спочатку створювався для розробки програмного забезпечення для вбудованих систем, але згодом став широко

використовуваним для створення різноманітних програм, веб-серверів, мобільних програм та корпоративних систем.

У Java є такі переваги над Python:

1. Має швидшу швидкість виконання: Java, будучи компілюваним у байт-код і виконуваним на віртуальній машині Java (JVM), часто забезпечує кращу продуктивність порівняно з Python, що інтерпретується.
2. Має строгу типізацію: Java є строго типізованою мовою, що допомагає виявляти помилки на етапі компіляції, роблячи код більш надійним та захищеним. А також: Сувора типізація також дозволяє JVM виконувати оптимізацію на основі типів даних, що може підвищити продуктивність.
3. Має багату екосистему бібліотек та інструментів: криптографічні бібліотеки Java надають потужні та добре документовані криптографічні бібліотеки, такі як Java Cryptography Extension (JCE), які полегшують реалізацію безпечних та ефективних криптографічних алгоритмів. Java має багатий набір бібліотек для інтеграції з різними системами та сервісами, що робить його зручним для розробки комплексних програм.

В прикладі буде використовуватись режим ECB, на Java, код має такий вигляд (див. Додаток А ст. 74).

1.4.4 Зрівняння Python, C++, Java

У цій таблиці можна подивитися зрівняння між мовами програмування, які зрівнюються за такими характеристиками: призначення, підтримка ООП, підтримка функціонального програмування, підтримка багатопоточності, складність, швидкодія, підтримка платформо незалежності, статична типізація, гнучкість, популярність.

Таблиця 1.2. – Зрівняння мов програмування Python, C++, Java

Характеристика	**Python**	**C++**	**Java**
Призначення	Загального призначення, особливо веб-розробка, науков обчислення, штучний інтелект	Загального призначення, ігрова розробка, вбудовані системи, системне програмування	Загального призначення, корпоративні системи мобільні додатки (Android), веб-розробка
Підтримка ООП	Так	Так	Так
Підтримка Функціонального програмування	Так	Частково	Ні
Підтримка багатопоточності	Так	Так	Так
Складність	Низька	Висока	Середня
Швидкодія	Повільна	Швидка	Середня
Підтримка Платформо незалежності	Так	Частково	Так
Статична типізація	Ні	Так	Так
Гнучкість	Висока	Низька	Середня
Популярність	Дуже висока	Висока	Висока

1.5 Опис функціонування коду

1.5.1 Початок коду та його кінець

В коді на початок та кінець зустрічаються одні та теж самі строки коду, де:

```
import tkinter as tk
from tkinter import messagebox
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import os
```

В цих строках імпортуються необхідні бібліотеки:

- tkinter для створення графічного інтерфейсу
- messagebox з tkinter для відображення повідомлень про помилки
- AES і pad, unpad з Crypto.Cipher та Crypto.Util.Padding для шифрування та додавання відступів.
- os для створення випадкового ключа.

У коді використовується бібліотека PyCryptodome для різних дій шифрування.

Та створення графічного виду:

```
root = tk.Tk()
root.title("ECB Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Вхідний текст:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Шифротекст:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Зашифрувати", command=encrypt_text)
```

```
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)
btn_decrypt = tk.Button(root, text="Розшифрувати", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)
# Запуск основного циклу GUI
root.mainloop()
```

Створення GUI:

- Створюємо основне вікно із заголовком "ECB Система шифрування".
- Додаємо мітки та поля введення для відкритого тексту та шифротексту.
- Додаємо кнопки для шифрування та розшифрування тексту.
- Запускаємо основний цикл GUI за допомогою root.mainloop().

1.5.2 Опис коду ECB (Electronic codebook)

```
class ECBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)
```

Тут ініціалізується класу ECBCipher:

Створення ключа шифрування та встановлення розміру блоку для AES.

```
    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Electronic Codebook (ECB)."""
        cipher = AES.new(self.key, AES.MODE_ECB) # Ініціалізація AES шифру в режимі ECB
        padded_plaintext = pad(plaintext, self.block_size) # Додавання відступів до відкритого тексту
        ciphertext = cipher.encrypt(padded_plaintext) # Шифрування тексту
        return ciphertext
    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Electronic Codebook (ECB)."""
        cipher = AES.new(self.key, AES.MODE_ECB) # Ініціалізація AES шифру в режимі ECB
        decrypted_padded_text = cipher.decrypt(ciphertext)  # Розшифровка тексту
```

```
        plaintext = unpad(decrypted_padded_text, self.block_size)  # Видалення відступів із
розшифрованого тексту

        return plaintext
```

Функції encrypt_text и decrypt_text:

- encrypt_text: Отримує текст із поля введення, шифрує його та виводить зашифрований текст у поле введення для шифротексту.
- decrypt_text: Отримує шифротекст із поля введення, розшифровує його та виводить вихідний текст у поле введення для відкритого тексту. Якщо розшифровка не вдається, відображається повідомлення про помилку.

Результат:

При запуске програми відкриться вікно, в якому можно буде ввести текст для шифрування та розшифрування.

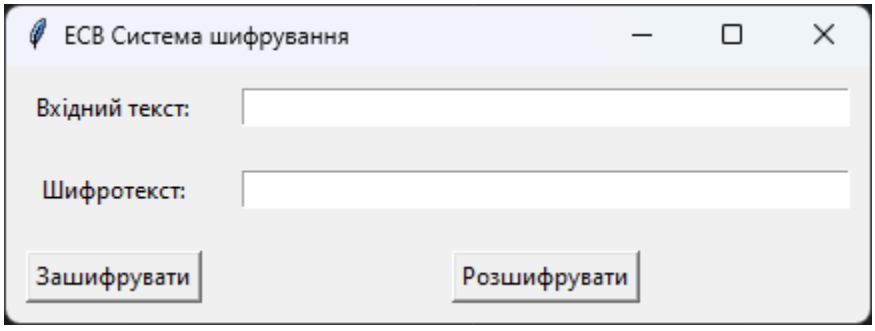


Рисунок 1.11. Вікно програми – ECB режиму шифрування

При шифруванні тексту "Sasha" ми отримуємо шифротекст "474ac53ac6cda4c45acdac4769666d13"

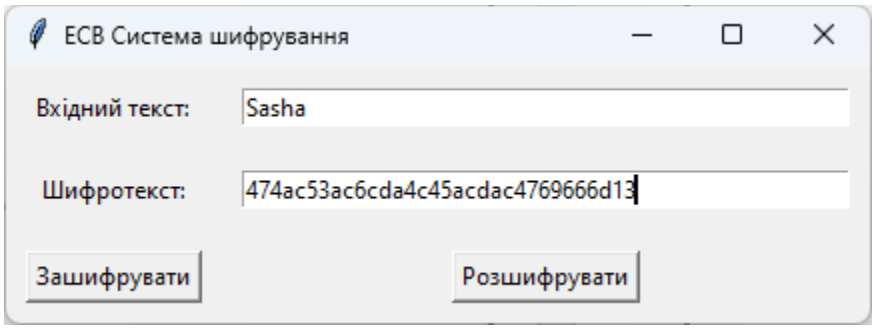


Рисунок 1.12. Шифрування слова та його результат – ECB режиму шифрування

При розшифруванні шифротексту "de0e3c5ec22379b7048ff588a410496b" ми отримуємо текст "Murzin"

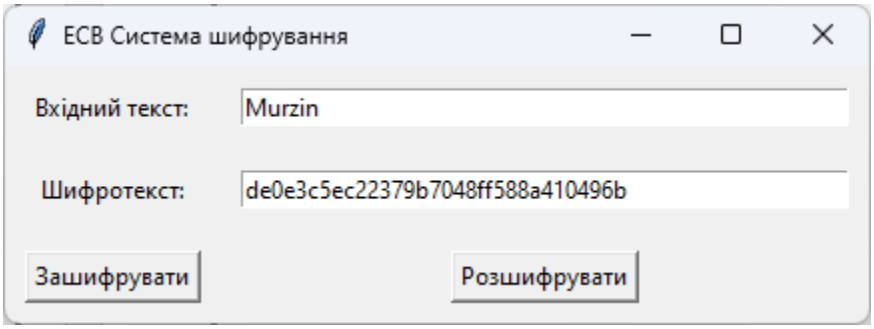


Рисунок 1.13. Розшифрування шифротексту та його результат – ECB режиму шифрування

1.5.3 Опис коду CBC (Cipher Block Chaining)

```
class BlockCipherSystem:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)
    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Cipher Block Chaining (CBC)."""
        iv = os.urandom(self.block_size)  # Генерація випадкового вектора ініціалізації (IV)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)  # Ініціалізація AES шифру в режимі CBC
        padded_text = pad(plaintext, self.block_size)  # Доповнення тексту до кратності розміру блоку
        ciphertext = cipher.encrypt(padded_text)  # Шифрування тексту
        return iv + ciphertext  # Повертаємо IV разом із шифротекстом
    def decrypt(self, ciphertext):
        """Розшифровування тексту в режимі Cipher Block Chaining (CBC)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_CBC, iv)  # Ініціалізація AES шифру в режимі CBC
        padded_text = cipher.decrypt(actual_ciphertext)  # Розшифровування тексту
        plaintext = unpad(padded_text, self.block_size)  # Видалення доповнення
        return plaintext
```

Клас BlockCipherSystem:

- Ініціалізується випадковим ключем та розміром блоку AES (16 байт).
- Метод encrypt: Спочатку генерує випадковий IV, потім ініціалізує шифр AES як CBC. Доповнює текст до кратності розміру блоку та шифрує його. На кінець повертає IV, доданий до зашифрованого тексту.
- Метод decrypt: Витягує IV із початку шифротексту. Потім розшифровує частину шифротексту, що залишилася. Видаляє додаток та повертає вихідний текст.

```
def encrypt_text():
    plaintext = entry_plaintext.get().encode()  # Отримуємо текст із поля введення та кодуємо в байти
    ciphertext = bcs.encrypt(plaintext)  # Шифруємо текст
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())  # Вставляємо зашифрований текст у полі введення

def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())  # Отримуємо шифротекст із поля введення та декодуємо з hex
        plaintext = bcs.decrypt(ciphertext)  # Розшифровуємо текст
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())  # Вставляємо розшифрований текст у полі введення
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")
```

Функції encrypt_text та decrypt_text:

encrypt_text: Отримує текст із поля введення, шифрує його та виводить зашифрований текст у поле введення для шифротексту.

decrypt_text: Отримує шифротекст із поля введення, розшифровує його та виводить вихідний текст у поле введення для відкритого тексту. Якщо розшифровка не вдається, відображається повідомлення про помилку.

Результат:

При запуске програми відкриться вікно, в якому можно буде ввести текст для шифрування та розшифрування.

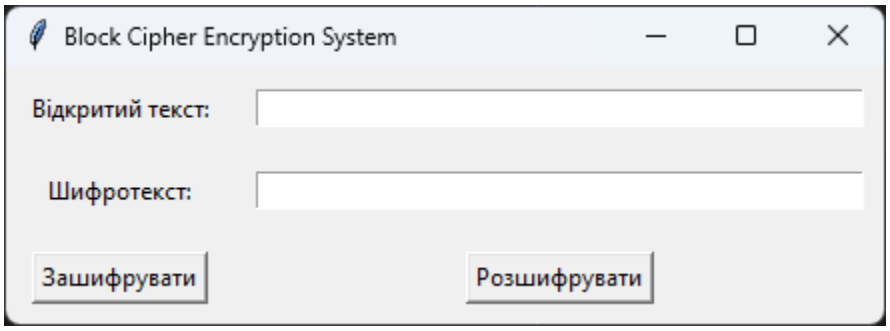


Рисунок 1.14. Вікно програми – CBC режиму шифрування

При шифруванні тексту "Sasha" ми отримуємо шифротекст “7fbaffe80a259f2d357270cfbfe0cf8f347767e4068e689aff924dbfe2805105”, але при повторному натисканні кнопки "Зашифрувати" шифротекст зміниться на інший, це обумовлено тим що в нас при натисканні випадковий вектор.

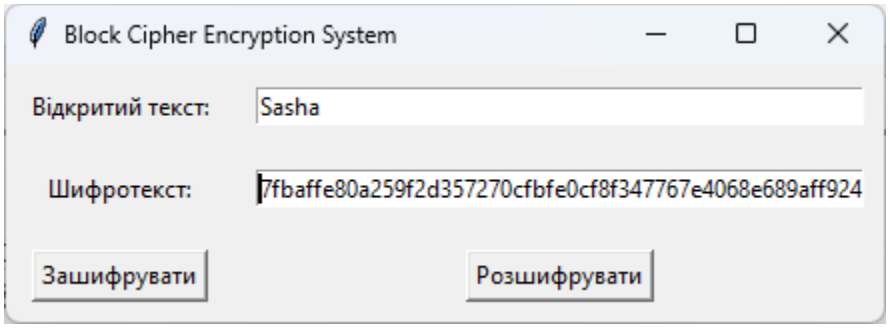


Рисунок 1.15. Шифрування слова та його результат – CBC режиму шифрування

При розшифруванні шифротексту:

“1d1734bd7d94cc9f47e92cd02c8bd7d200c8d813e78bc137324ff4798eb1de89” ми отримуємо текст "Murzin"

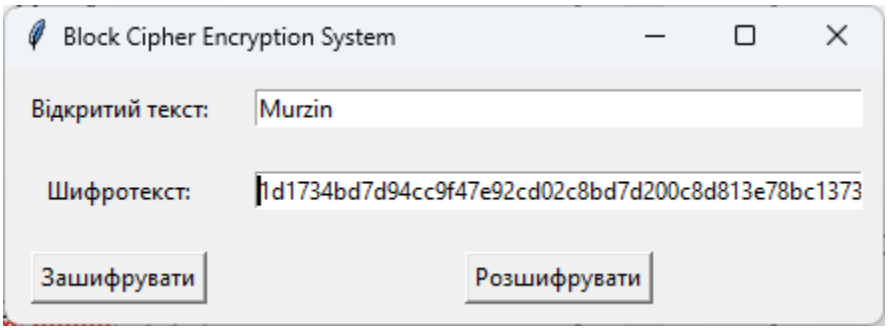


Рисунок 1.16. Розшифрування шифротексту та його результат – CBC режиму шифрування

1.5.4 Опис коду CFB (Cipher Feedback)

```
class CFBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.iv = os.urandom(AES.block_size)  # Генерація випадкового вектора ініціалізації
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)

    def encrypt(self, plaintext):
        """Шифрование текста в режиме Cipher Feedback (CFB)."""
        cipher = AES.new(self.key, AES.MODE_CFB, self.iv)  # Ініціалізація AES шифру в режимі CFB
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return self.iv + ciphertext  # Повертаємо IV разом із шифротекстом

    def decrypt(self, ciphertext):
        """Розшифрування тексту у режимі Cipher Feedback (CFB)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_CFB, iv)  # Ініціалізація AES шифру в режимі CFB
```

```
        plaintext = cipher.decrypt(actual_ciphertext)  # Розшифрування тексту
        return plaintext
```

Клас CFBCipher:

__init__: Генерує випадковий ключ та вектор ініціалізації, задає розмір блоку AES (16 байт).

encrypt: Ініціалізує шифр AES у режимі CFB, шифрує текст та повертає IV, доданий до зашифрованого тексту.

decrypt: Витягує IV з початку шифротексту, ініціалізує шифр AES у режимі CFB, розшифровує текст та повертає вихідний текст.

```
def encrypt_text():
    plaintext = entry_plaintext.get().encode()
    ciphertext = cfb_cipher.encrypt(plaintext)
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())

def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())
        plaintext = cfb_cipher.decrypt(ciphertext)
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")
```

Функції encrypt_text та decrypt_text:

encrypt_text: Отримує текст із поля введення, шифрує його та виводить зашифрований текст у поле введення для шифротексту.

decrypt_text: Отримує шифротекст із поля введення, розшифровує його та виводить вихідний текст у поле введення для відкритого тексту. Якщо розшифровка не вдається, відображається повідомлення про помилку.

Результат:

При запуске програми відкриться вікно, в якому можно буде ввести текст для шифрування та розшифрування.

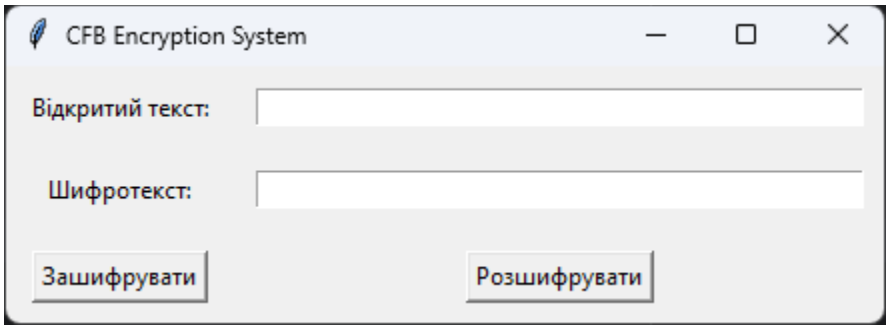


Рисунок 1.17. Вікно програми – CFB режиму шифрування

При шифруванні тексту "Sasha" ми отримуємо шифротекст "8b95e1ad964b5bd758c4ff5e255312d58fcb3cb7fd", але запуске програми та введені того ж самого тексту, шифротекст буде іншим. Це тому що при запуске генерується новий вектор.

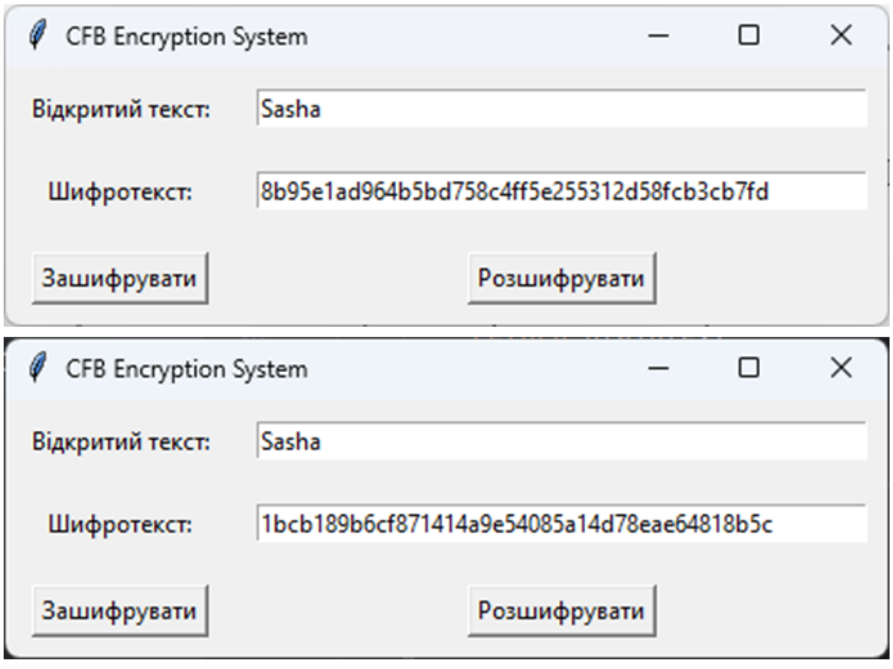


Рисунок 1.18а, 1.18б. Шифрування слова та його результат – CFB режиму шифрування

При розшифруванні шифротексту:

“b1d366a674ea8786d2ade180a3a54c2af9dc25255060” ми отримуємо текст "Murzin"

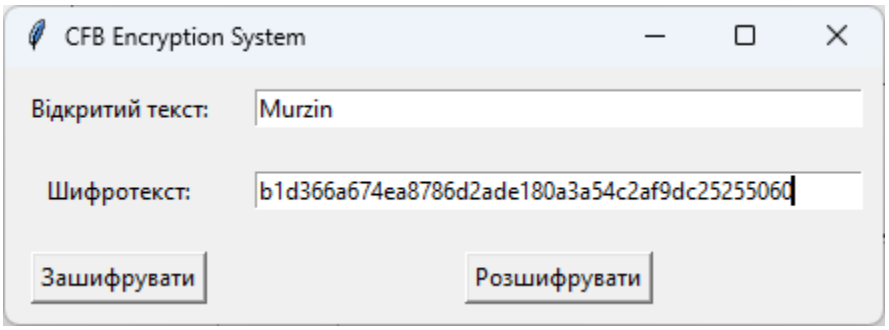


Рисунок 1.19. Розшифрування шифротексту та його результат – CFB режиму шифрування

1.5.5 Опис коду OFB (Output Feedback)

```
class OFBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)
    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Output Feedback (OFB)."""
        iv = os.urandom(self.block_size)  # Генерація випадкового вектора ініціалізації (IV)
        cipher = AES.new(self.key, AES.MODE_OFB, iv)  # Ініціалізація AES шифру в режимі OFB
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return iv + ciphertext  # Повертаємо IV разом із шифротекстом
    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Output Feedback (OFB)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_OFB, iv)  # Ініціалізація AES шифру в режимі OFB
        plaintext = cipher.decrypt(actual_ciphertext)  # Розшифрування тексту
        return plaintext
```

Клас OFBCipher:

- Ініціалізується випадковим ключем та розміром блоку AES (16 байт).

Метод encrypt:

- Генерує випадковий IV.
- Ініціалізує шифр AES як OFB.
- Шифрує текст та повертає IV, доданий до зашифрованого тексту.

Метод decrypt:

- Витягує IV із початку шифротексту.
- Розшифровує частину шифротексту, що залишилася, і повертає вихідний текст.

```
def encrypt_text():
    plaintext = entry_plaintext.get().encode()  # Отримуємо текст із поля введення та кодуємо в байти
    ciphertext = ofb_cipher.encrypt(plaintext)  # Шифруємо текст
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())  # Вставляємо зашифрований текст у поле введення


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())  # Отримуємо шифротекст з поля введення та декодуємо з hex
        plaintext = ofb_cipher.decrypt(ciphertext)  # Розшифровуємо текст
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())  # Вставляємо розшифрований текст у полі введення
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")
```

Функції encrypt_text та decrypt_text:

- encrypt_text: Отримує текст із поля введення, шифрує його та виводить зашифрований текст у поле введення для шифротексту.
- decrypt_text: Отримує шифротекст із поля введення, розшифровує його та виводить вихідний текст у поле введення для відкритого тексту. Якщо розшифровка не вдається, відображається повідомлення про помилку.

Результат:

При запуске програми відкриться вікно, в якому можно буде ввести текст для шифрування та розшифрування.

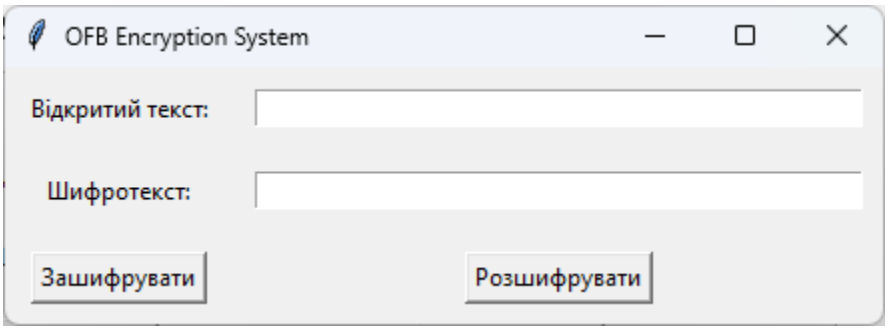


Рисунок 1.20. Вікно програми – OFB режиму шифрування

При шифруванні тексту "Sasha" ми отримуємо шифротекст “620b4705518d7878eefe7b82bada88e4333e4d14a1”, але при повторному натисканні кнопки "Зашифрувати" шифротекст зміниться на інший, це обумовлено тим що в нас при натисканні випадковий вектор

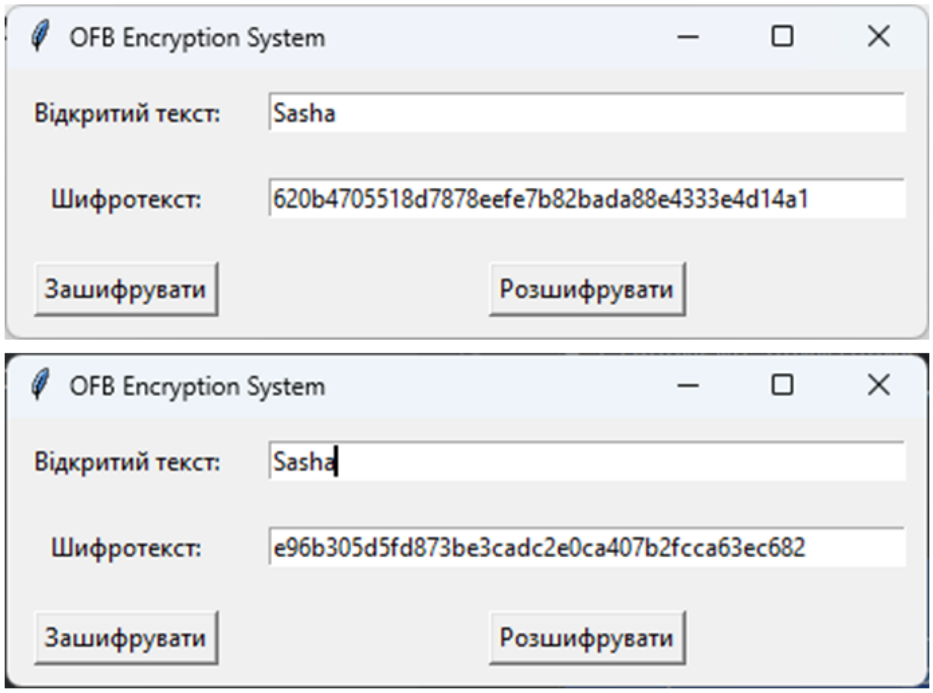


Рисунок 1.21а, 1.21б. Шифрування слова та його результат – OFB режиму шифрування

При розшифруванні шифротексту:

“c324dc90da9e4243a9374e7fb51b4bb1616bbf7b0725” отримуємо текст "Murzin"

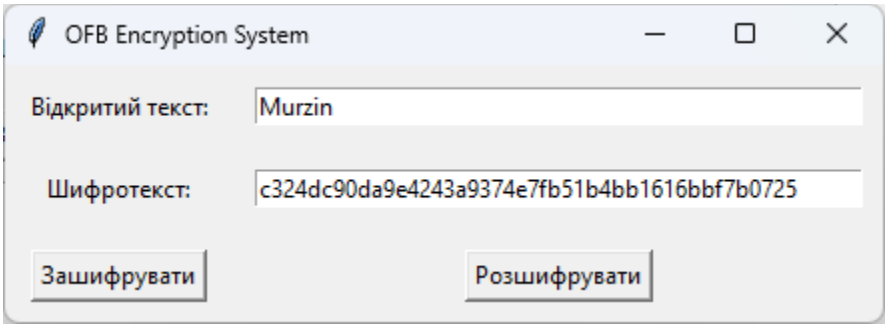


Рисунок 1.22. Розшифрування шифротексту та його результат – OFB режиму шифрування

1.5.6 Опис коду CTR (Counter)

```
class CTRCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.nonce = os.urandom(8)  # Генерація випадкового значення nonce (8 байт)
    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Counter (CTR)."""
        cipher = AES.new(self.key, AES.MODE_CTR, nonce=self.nonce)  # Ініціалізація AES шифру в режимі CTR
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return ciphertext
    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Counter (CTR)."""
        cipher = AES.new(self.key, AES.MODE_CTR, nonce=self.nonce)  # Ініціалізація AES шифру в режимі CTR
        plaintext = cipher.decrypt(ciphertext)  # Розшифрування тексту
        return plaintext
```

Клас CTRCipher:

- __init__: Генерує випадковий ключ та значення nonce, які використовуються для ініціалізації шифру.

- encrypt: Ініціалізує шифр AES у режимі CTR та шифрує текст.
- decrypt: Ініціалізує шифр AES у режимі CTR та розшифровує текст.

```
def encrypt_text():
    plaintext = entry_plaintext.get().encode()
    ciphertext = ctr_cipher.encrypt(plaintext)
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())
def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())
        plaintext = ctr_cipher.decrypt(ciphertext)
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")
```

Функції encrypt_text та decrypt_text:

- encrypt_text: Отримує текст із поля введення, шифрує його та виводить зашифрований текст у поле введення для шифротексту.
- decrypt_text: Отримує шифротекст із поля введення, розшифровує його та виводить вихідний текст у поле введення для відкритого тексту. Якщо розшифровка не вдається, відображається повідомлення про помилку.

Результат:

При запуске програми відкриться вікно, в якому можно буде ввести текст для шифрування та розшифрування.

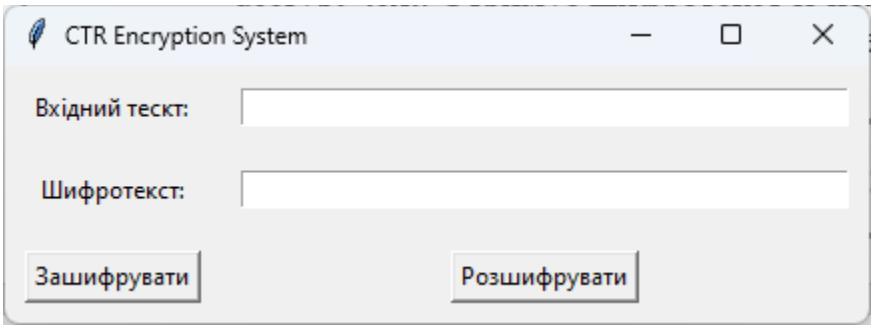


Рисунок 1.23. Вікно програми – CTR режиму шифрування

При шифруванні тексту "Sasha" ми отримуємо шифротекст " 6a3640424a", але запуске програми та введені того ж самого тексту, шифротекст буде іншим. Це тому що при запуске генерується новий ключ та лічильник.

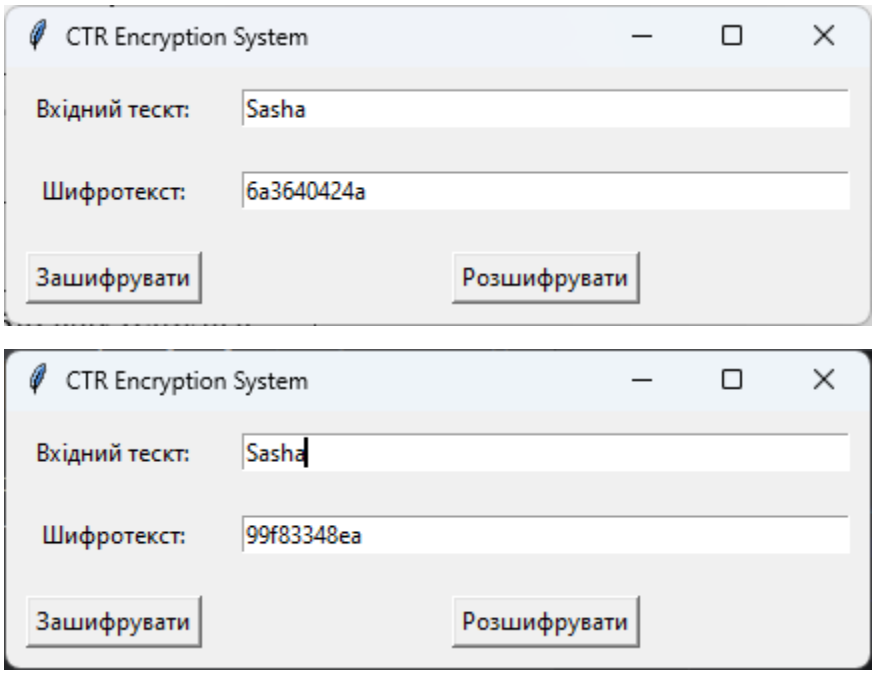


Рисунок 1.24а, 1.24б. Шифрування слова та його результат – CTR режиму шифрування

При розшифруванні шифротексту: “87ec325ae2c8” отримуємо текст "Murzin"

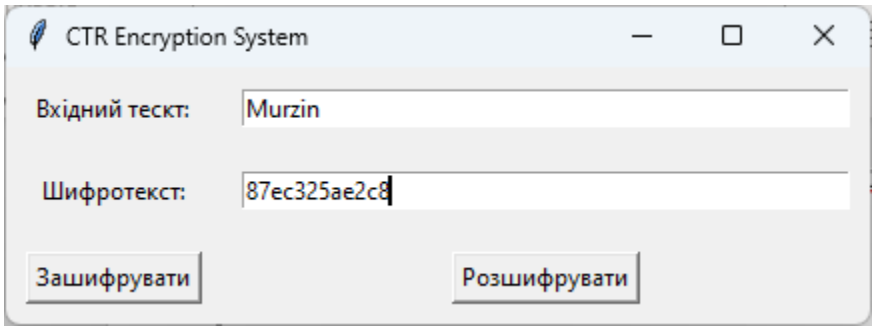


Рисунок 1.25. Розшифрування шифротексту та його результат – CTR режиму шифрування

1.6 Блок-схема алгоритмів режимів шифрування реалізованих на Python

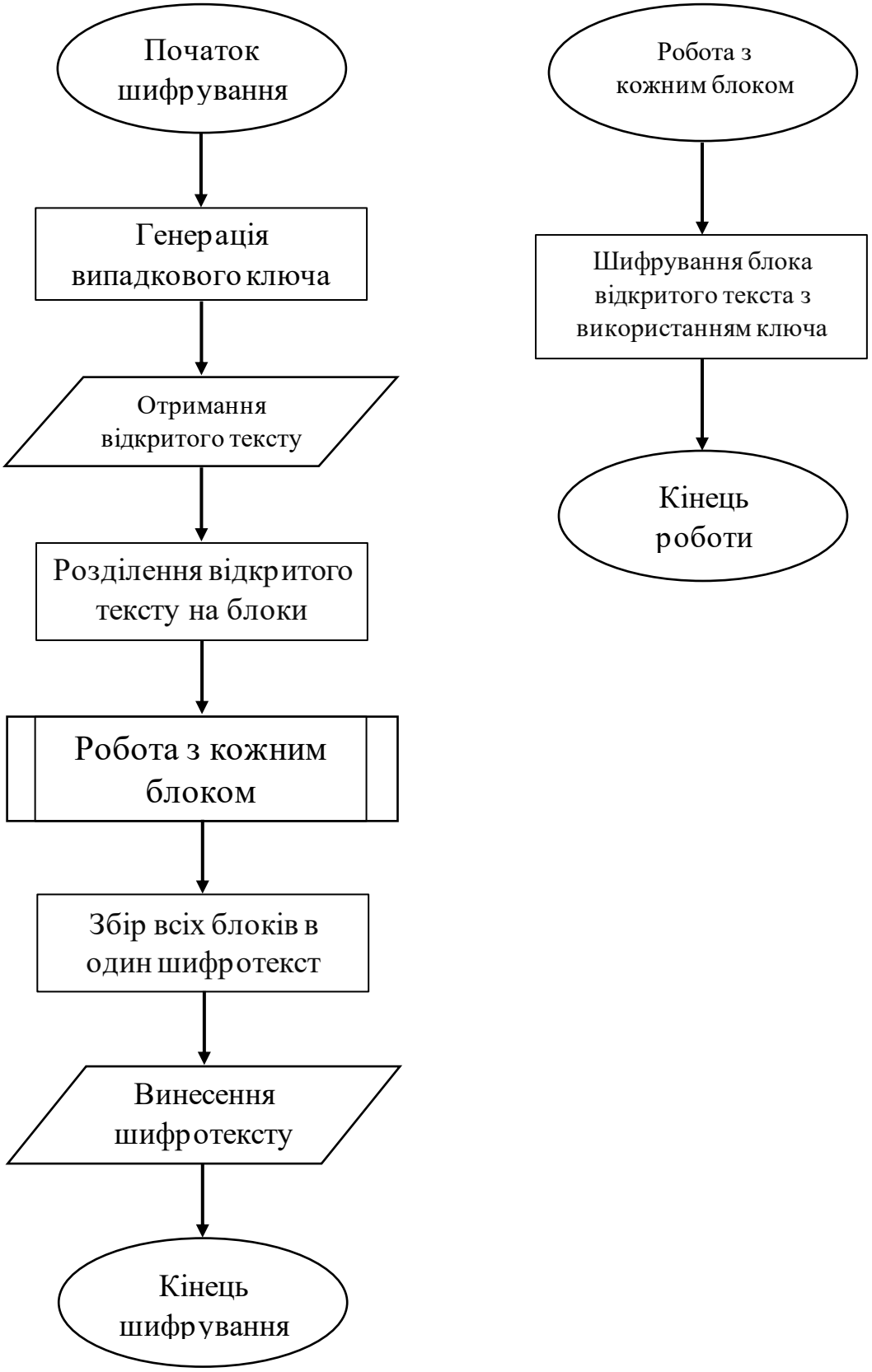


Рисунок 1.26. шифрування ECB на Python

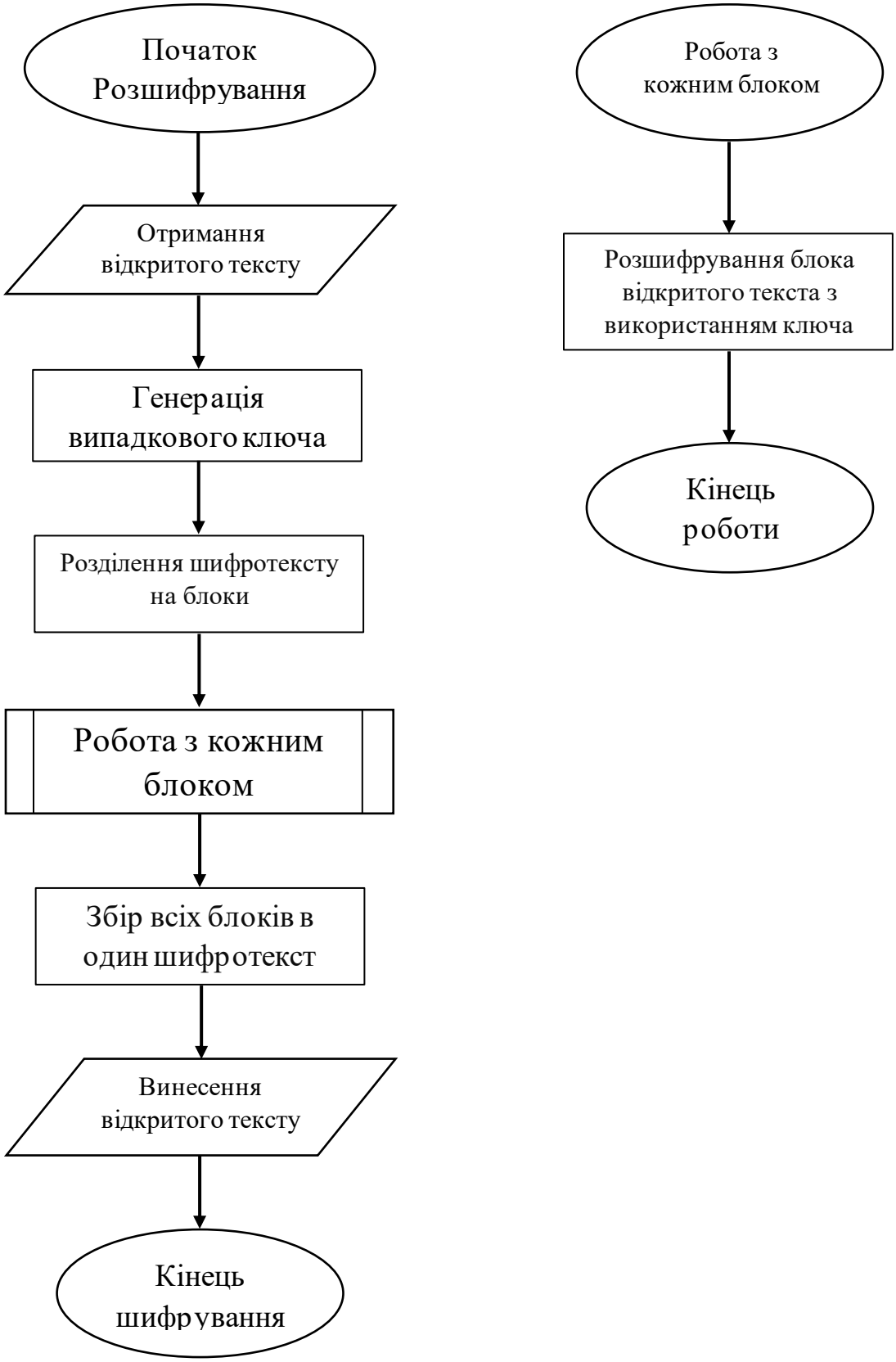


Рисунок 1.27. розшифрування ECB на Python

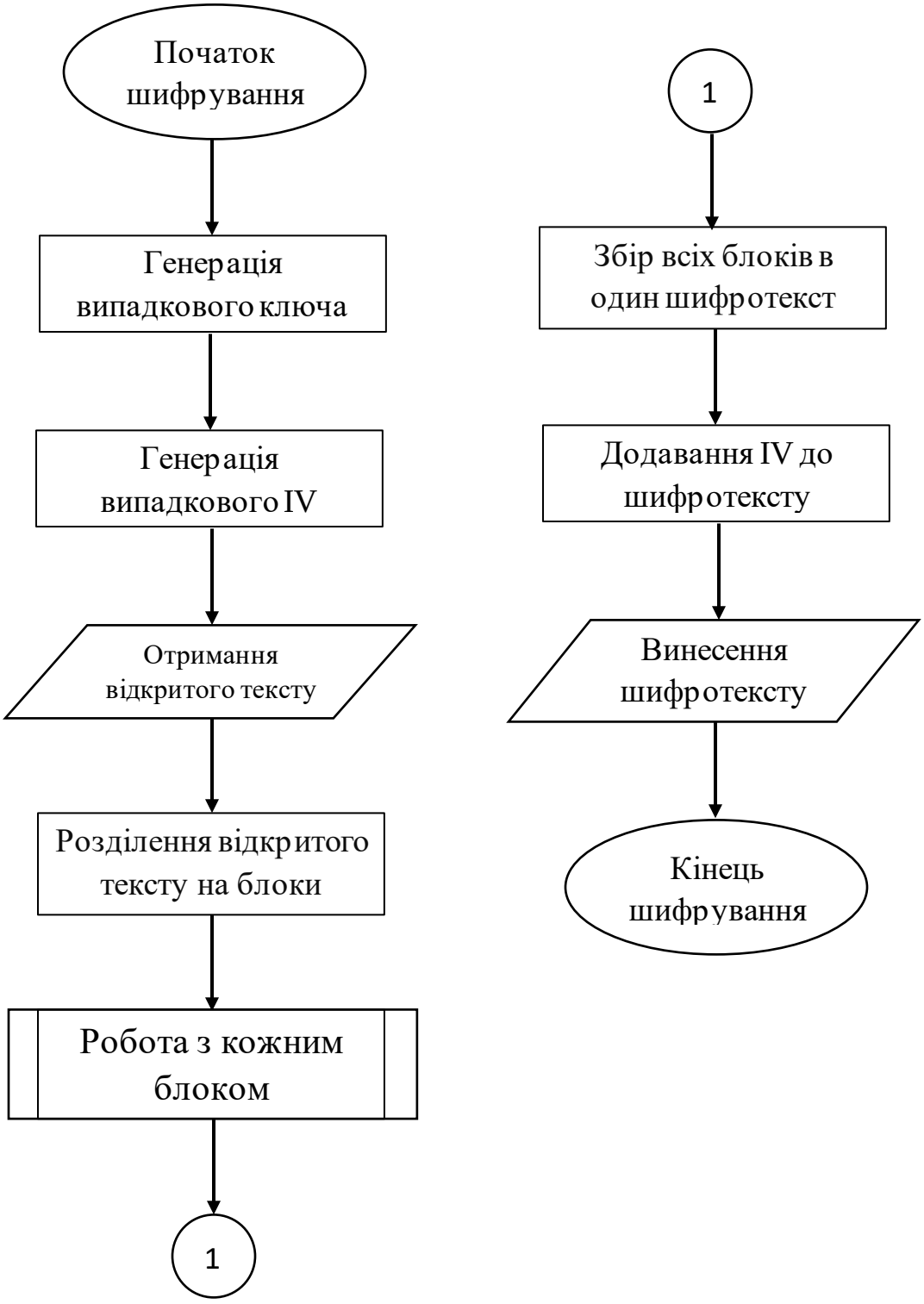


Рисунок 1.28. шифрування CBC на Python

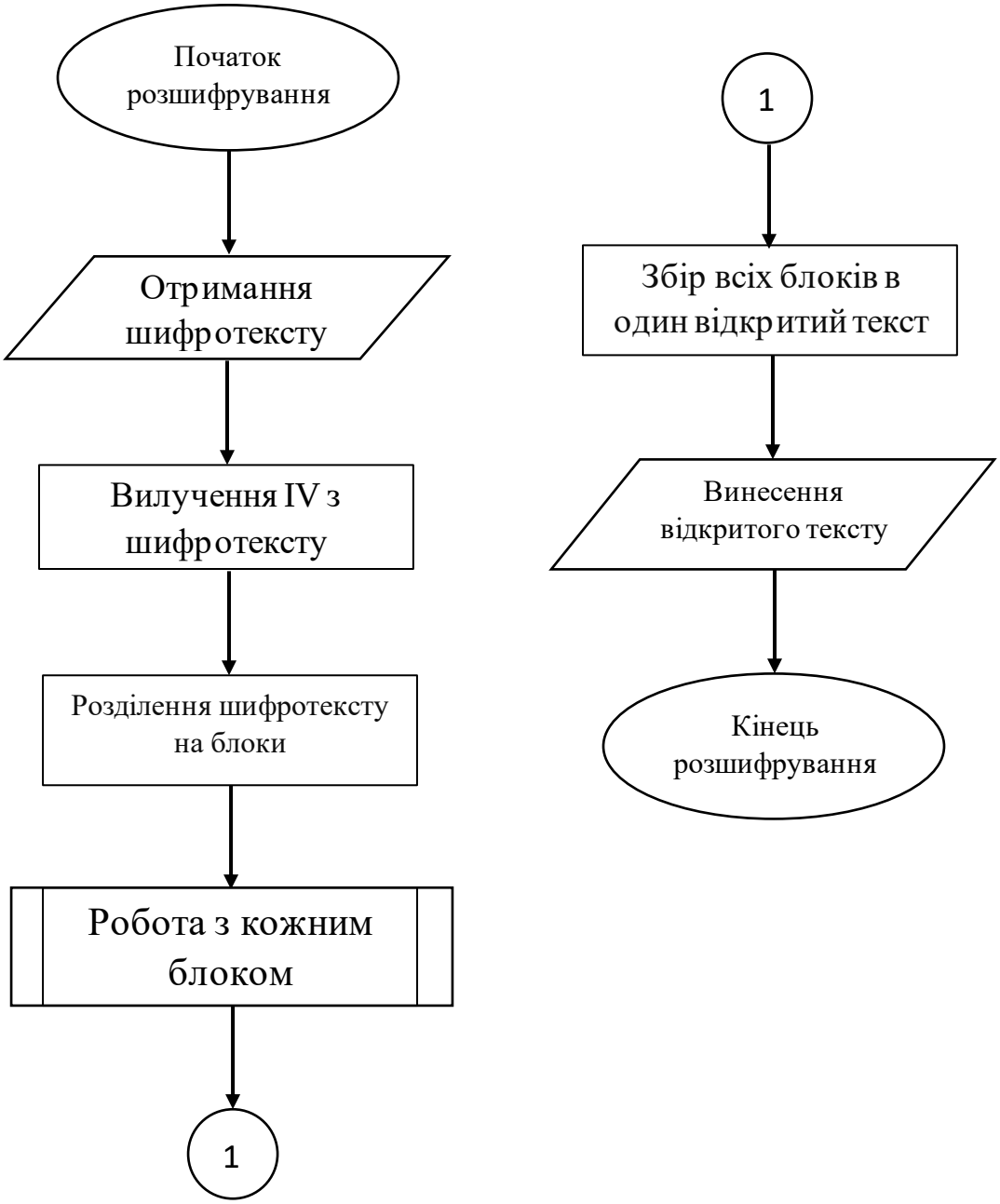


Рисунок 1.29. шифрування CBC на Python

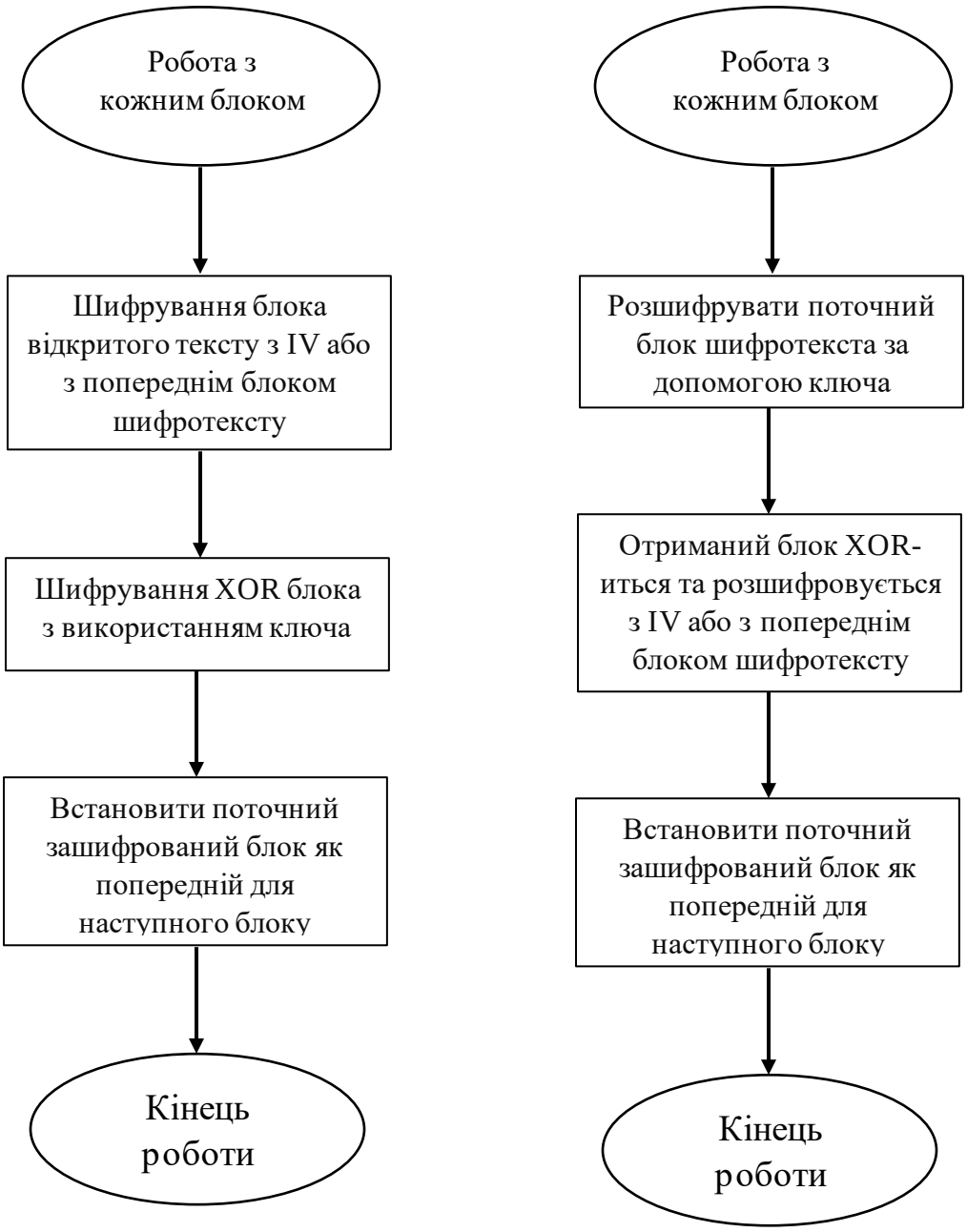


Рисунок 1.30. Процес шифрування та розшифрування з кожним блоком відкритого тексту та шифротекстом, в алгоритмі CBC

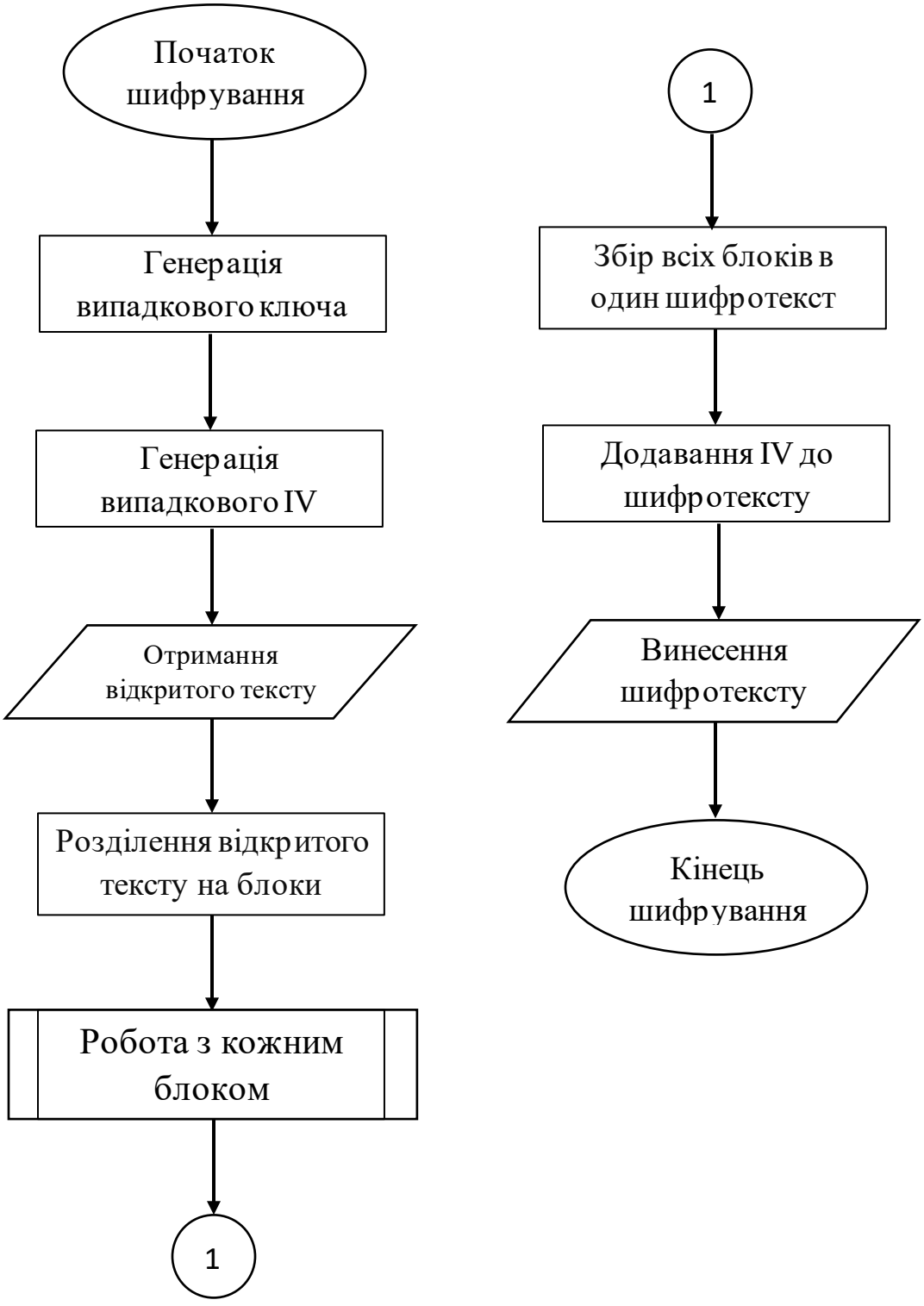


Рисунок 1.31. шифрування OFB на Python

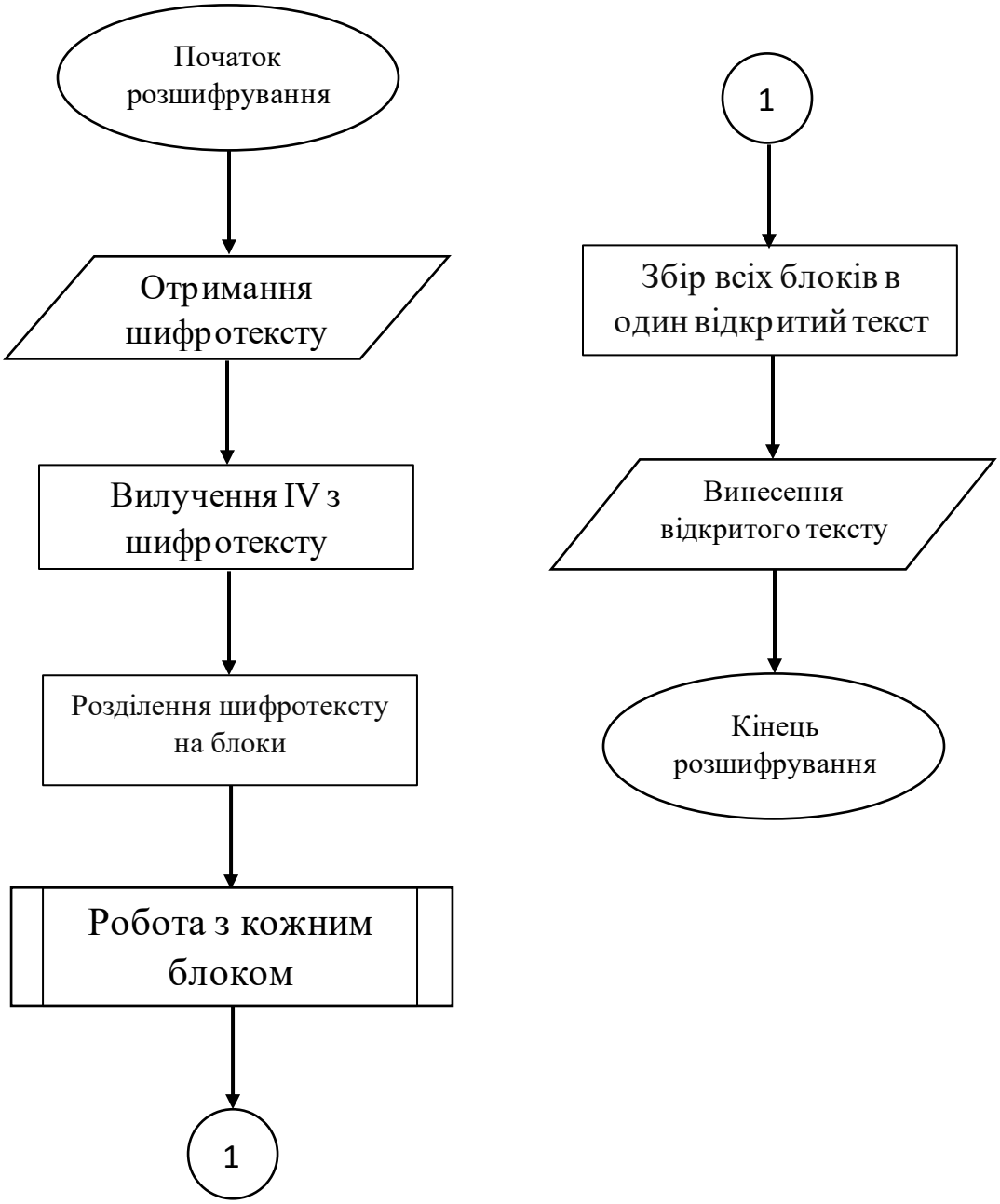


Рисунок 1.32. шифрування OFB на Python

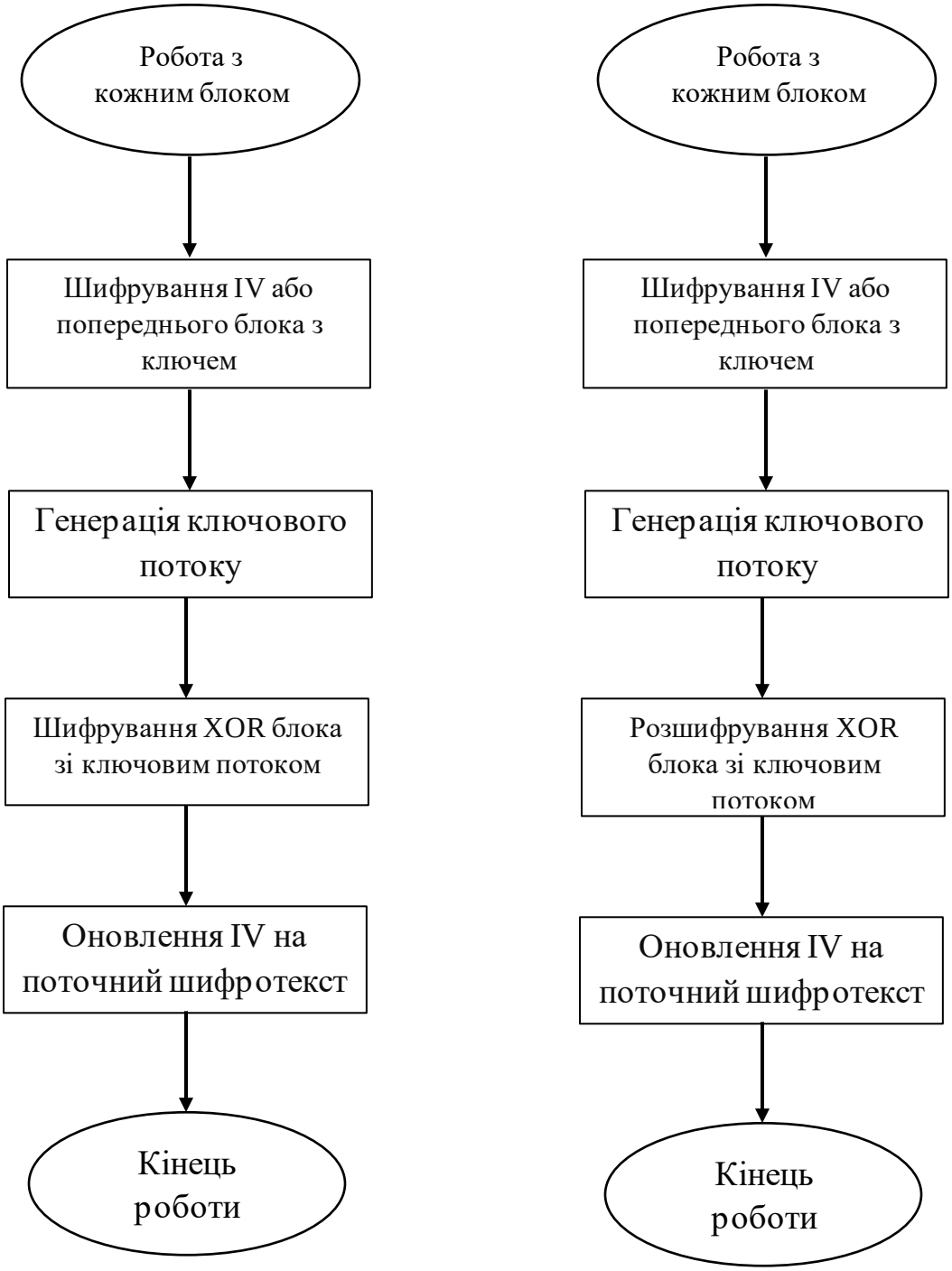


Рисунок 1.33. Процес шифрування та розшифрування з кожним блоком відкритого тексту та шифротекстом, в алгоритмі OFB

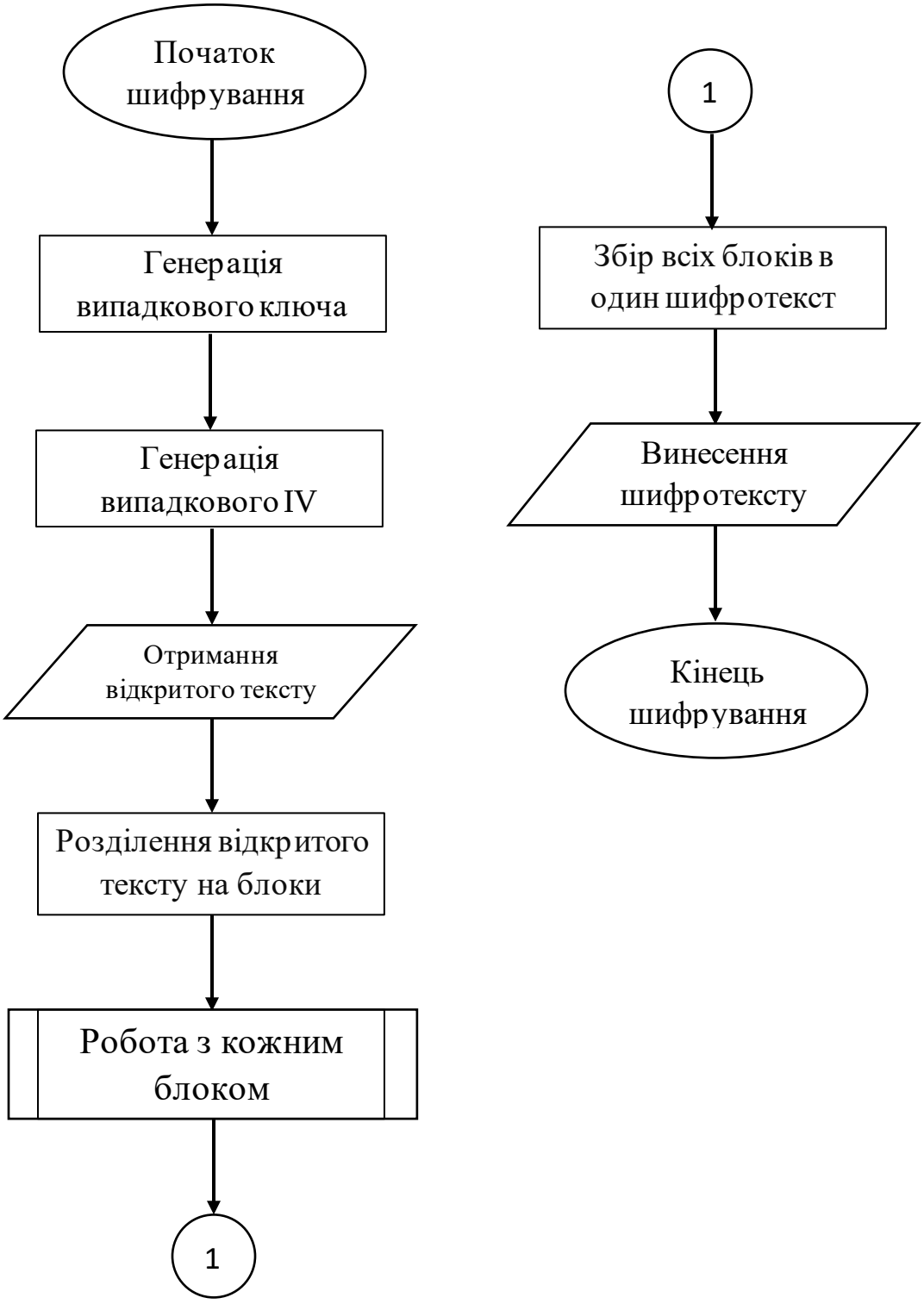


Рисунок 1.34. шифрування CFB на Python

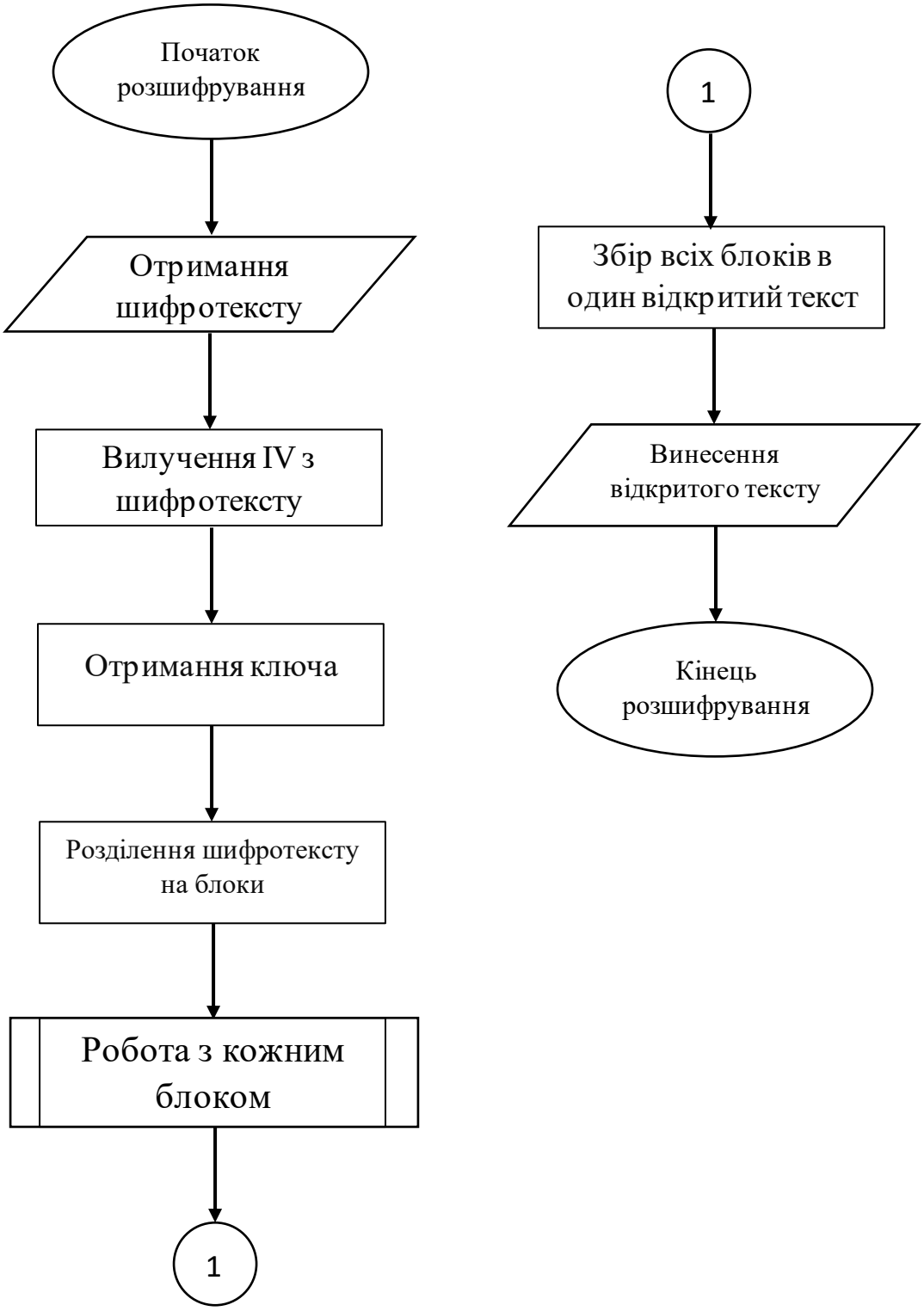


Рисунок 1.35. розшифрування CFB на Python

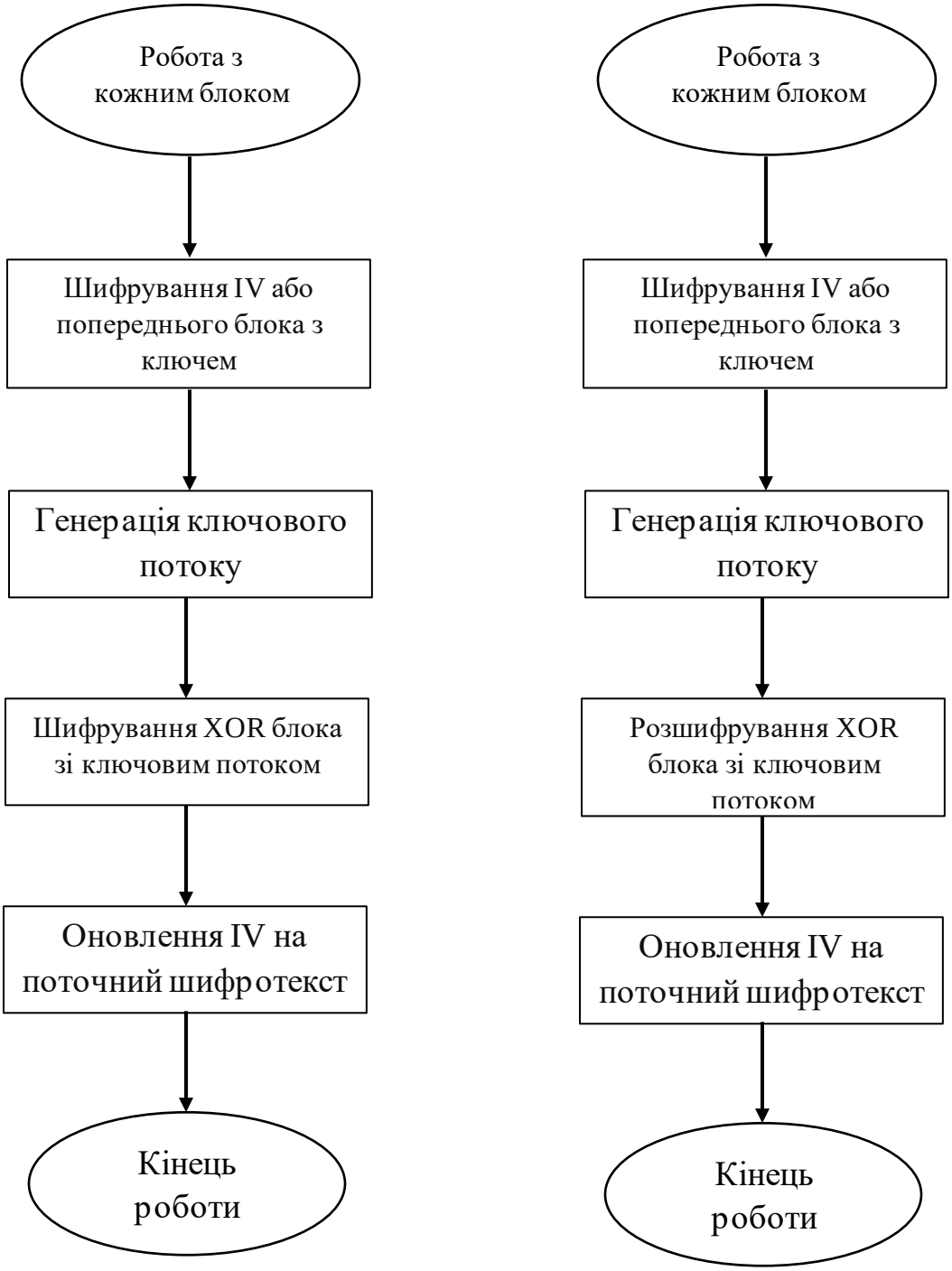


Рисунок 1.36. Процес шифрування та розшифрування з кожним блоком відкритого тексту та шифротекстом, в алгоритмі CFB

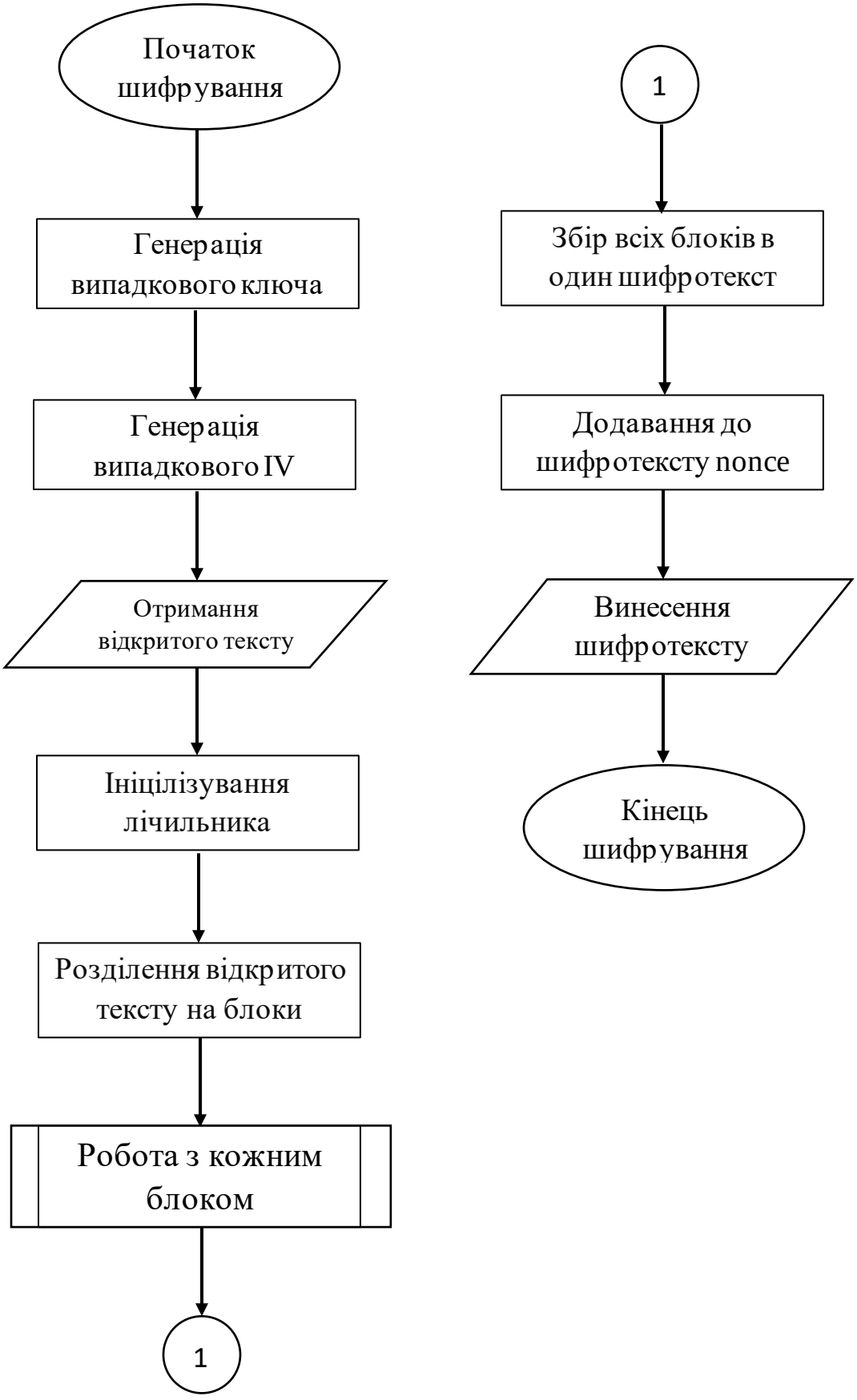


Рисунок 1.37. шифрування CTR на Python

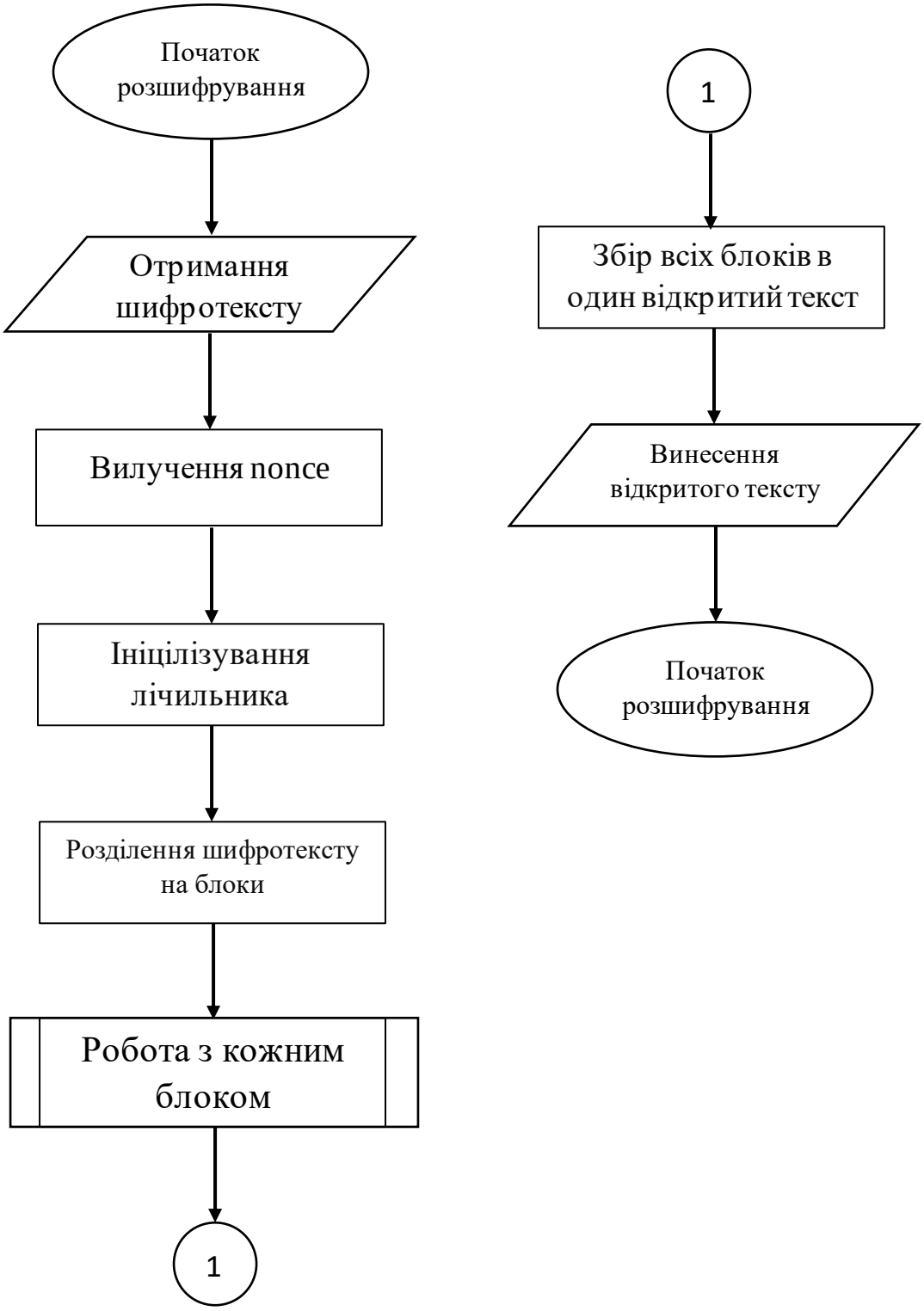


Рисунок 1.38. розшифрування CTR на Python

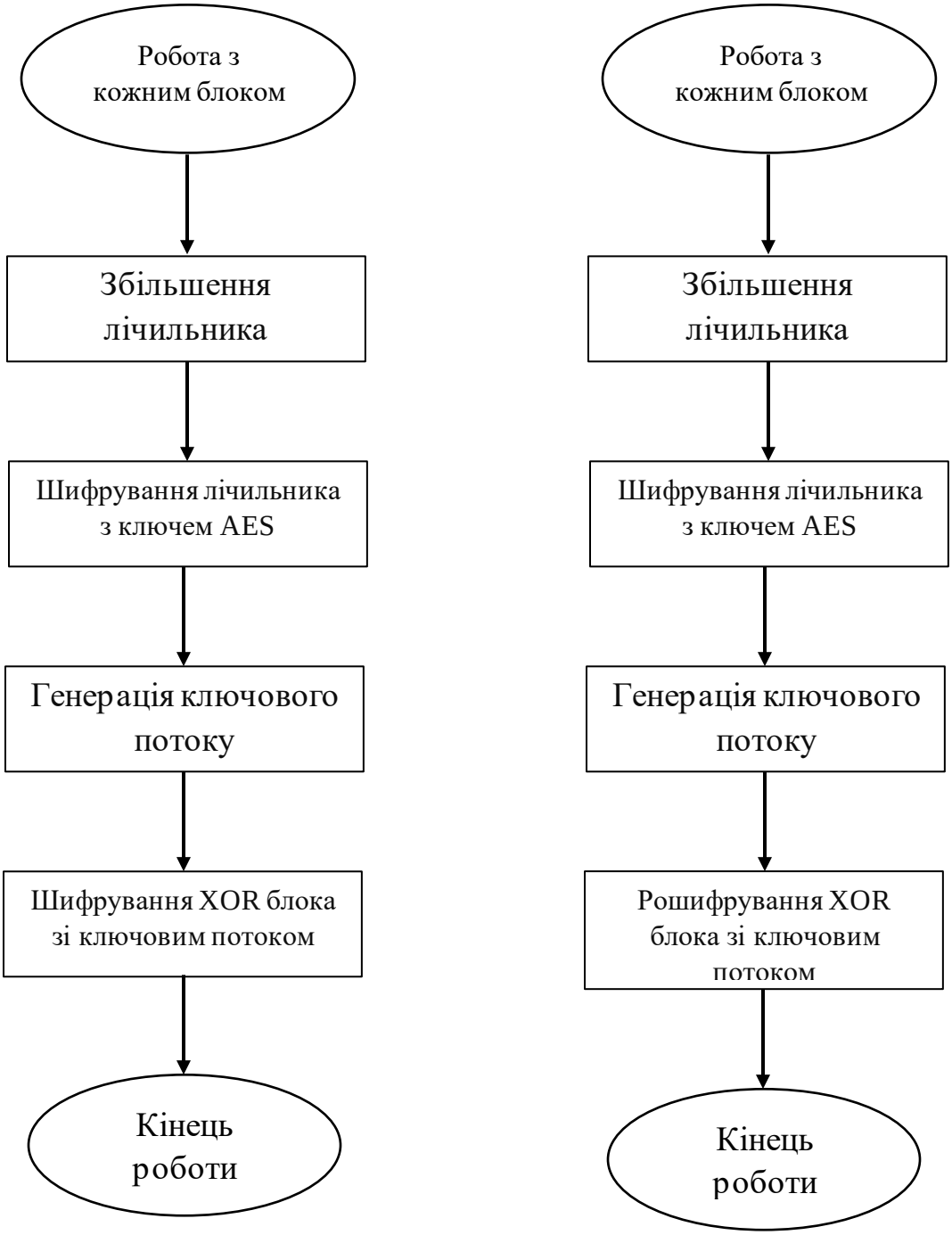


Рисунок 1.39. Процес шифрування та розшифрування з кожним блоком відкритого тексту та шифротекстом, в алгоритмі CTR

ВИСНОВКИ

У даній роботі виконано дослідження блочного метода шифрування на мові програмування Python. У роботі було розглянуто методи шифрування та також п’ять режимів блочного шифрування, та проведене зрівняння режимів шифрування, та реалізація їх на мові програмуванні Python, таких режимів як: Elecronic Codeboock, Cipher Block Chaining, Cipher Feedback, Output Feedback, Counter. В роботі також є аналіз мов програмування та їх реалізація коду режиму шифрування ЕВС.

У практичній частині є розробка програмного забезпечення на Python, яке дає змогу дізнатися про процес шифрування та розшифрування тексту, у програм також був реалізований простий графічний інтерфейс для користувача.

Треба зазначити, що Python є найбільш простим та зручним для розробки криптографічного програмного забезпечення, який дає змогу навчитися створювати програми для захисту інформації. Python має багатий набір бібліотек, таких як cryptography та PyCrypto, що полегшує реалізацію складних криптографічних алгоритмів. Крім того, у роботі було детально проаналізовано ефективність цих бібліотек у контексті різних режимів блочного шифрування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Ю.А. Тарнавський – «ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ»

2. Н.О. ЩУР, О.А. ПОКОТИЛО – «ОСНОВИ КРИПТОЛОГІЇ»

3. Сергій Козлов - “Криптографічні протоколи: від теорії до практики”

4. Шифрование — Вікіпедія (wikipedia.org)

5. Режими блокового шифрування — Вікіпедія [Веб-сайт] (wikipedia.org)

6. Криптографія — Вікіпедія [Веб-сайт] (wikipedia.org)

7. Java — Вікіпедія [Веб-сайт] (wikipedia.org)

8. C++ — Вікіпедія [Веб-сайт] (wikipedia.org)

9. Python — Вікіпедія [Веб-сайт] (wikipedia.org)

10. Бауер Ф. – "Розшифровані таємниці. Методи та принципи криптології."

11. Криптографічна система [Веб-сайт] (https://wiki.tntu.edu.ua)

12. Криптографія [Веб-сайт] (https://uk.wikipedia.org)

13. Crypto101 - Block Ciphers [Веб-сайт] (https://www.crypto101.io)

14. Coursera - Cryptography Courses [Веб-сайт] (https://www.coursera.org/)

15. Schneier on Security [Веб-сайт] (https://www.schneier.com/)

16. NIST - Block Cipher Modes [Веб-сайт] (https://csrc.nist.gov/)

17. Криптографія та мережева безпека Вільяма Столлінгса (книга)

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

2.1 Вступ

Охорона праці – це не лише набір правил і вказівок для працівників, а повноцінна система заходів, спрямованих на забезпечення безпечних та комфортних умов праці. Вона включає в себе регулювання таких параметрів, як освітлення, рівень шуму, мікроклімат, розмір робочого простору та інші важливі аспекти. Забезпечення комфорту та безпеки працівників здійснюється через дотримання нормативних актів, законів та положень. Недотримання цих стандартів може призвести до швидкого втомлення працівників, появи помилок та загрози професійних захворювань або травм.

2.2 Аналіз та безпека умов праці працівника на робочому місці

Тема диплома "Дослідження блокових криптографічних систем шифрування інформації на основі мови програмування Python". Тому будем аналізувати з точки зору програміста в офісі.

Основною метою заходів щодо охорони праці - ліквідація травматизму й професіональних захворювань. Завдяки цим заходам можливо зменшити витрати на відновлення втраченої працездатності та підвищити продуктивність праці.

Всі заходи щодо охорони праці проводяться з метою захисту учасників трудового процесу від впливу небезпечних і шкідливих факторів. У дипломному проекті розглядається робота працівника-програміста, тому особливо небезпечних факторів мало, але кілька малих шкідливих факторів є, які при тривалому впливі зустрічаються. Наприклад довгий час сидіння за комп’ютером може спричинити болі у хребті.

2.3 Організація робочого місця

У дипломному проекті розглядається праця дослідника у сфері ІТ. В цій сфері загалом не потрібно особливих умов для роботи, тому робоче місце виглядить так: Стіл на якому розмінюється персональний комп'ютер, або ноутбук; Ергономічне крісло на якому працівник зможе правильно сидіти; Персональний комп'ютер або ноутбук який використовує працівник для роботи.

Для того щоб правильно сидіти, працівник повинен дотримуватись таких правил:

1. Сидіти прямо, держати спину рівно.
2. Розмістити комп'ютер на відстані 70 см.
3. Не закидати ногу на ногу, стопи мають твердо стояти на підлозі.
4. Між працівником та краєм столу має бути вільний простір хоча б у 5 см.
5. Відстань від екрана до очей повинна бути не менше 50 см.
6. Монітор повинен бути трохи нижче горизонтальної лінії погляду в межах 30 градусів.
7. Лікті повинні повністю лежати на столі, не звисаючи і розташовуватися перпендикулярно до підлоги.
8. Руки повинні бути покладені рівно на клавіатурі, не згинаючи їх у зап'ястях.
9. Працівнику потрібно зменшити навантаження на очі

2.4 Перерви та паузи

Для тих хто працює за комп'ютером важливо робити перерви на відпочинок, для того щоб уникнути втомленість очей та зап'ястного синдрому. За статтею 66 КЗпП передбачає надання працівникам перерви для відпочинку і харчування тривалістю не більше 2 годин, цей час не включається у робочий час. Перерва на відпочинок надається через 4 години після початку роботи. Час початку та кінця встановлюється правилами внутрішнього трудового розпорядку.

2.5 Шум

Шум один з важливих факторів роботи в приміщені. Робота у шумному приміщені може робити працівників більш дратівливими та швидше втомлювати людей. Тому важливо щоб в приміщені було тихо та сторонні шуми не проходили в приміщення. В цьому допоможе металопластикові вікна. Верхня межа шуму в офісі не повинна переважати 55 дБ для комфортної праці.

2.6 Мікроклімат

Санітарно-гігієнічне нормування умов мікроклімату здійснюється за ДСН 3.3.6.042-99 . Мікроклімат повинен відповідати нормативам.

Таблиця 2.1. Рівень іонів у повітрі на сантиметр кубічний

Пора року	Температура повітря у градусах Цельсія.	Відносна вологість повітря у відсотках.	Швидкість руху повітря у метрах за секунду.
Холодна	22-24	40-60	0,1
Тепла	23-25	40-60	0,1

Для забезпечення оптимального складу повітря в приміщенні, яке відповідає гігієнічним нормам, використовується система вентиляції. Природна вентиляція через вікна дозволяє повітрю потрапляти в приміщення та видалятися за рахунок теплового обміну, але вона має свої обмеження. Тому частіше використовується штучна загальна вентиляція, яка очищає повітря та постачає його до робочих місць. Перед подачею повітря можна піддавати різним обробкам, таким як підігрів, зволоження, охолодження і т.д.

Рівень іонів у повітрі що повинні відповідати санітарно-гігієнічним нормам № 2152-80. Наведені у таблиці 3.3

Таблиця 2.2. Рівень іонів у повітрі на сантиметр кубічний

Рівні	Кількість на сантиметр кубічний.	
	n+	n-
Мінімальне	400	600
Оптимальне	1500-300	3000-5000
Максимальне	50000	50000

Для підтримки мікроклімату слід використовувати кондиціонери та зволожувачі повітря або інші прилади.

2.7 Електробезпека

В сучасному світі майже в будь-якому приміщені є розетки та електрика, тому всі працівники на виробництві мають знати як себе поводити з електроприладами та що потрібно робити, щоб запобігти створенню екстремальних ситуацій. Для того що б запобігти цьому достатньо провести інструктаж з охорони праці по правильному використанні електронного обладнання. Також для запобігання потрібно правильно розподілити електроживлення щоб запобігти короткому замиканню та іншим електричним пригодам.

2.8 Пожежна безпека

Пожежна безпека є необхідною складовою системи заходів з охорони праці на робочих місцях, що охоплює широкий спектр заходів. Ці заходи включають:

- Створення безпечних умов праці.

- Мінімізацію ризику виникнення пожеж.

- Забезпечення технічними засобами для запобігання пожежам та їхнього усунення.

- Контроль за дотриманням протипожежних вимог і норм законодавства.

- Розробку та впровадження процедур щодо гасіння пожеж, евакуації та порятунку людей і майна.

- Проведення навчання з питань пожежної безпеки для співробітників.

Первинні засоби пожежогасіння використовуються для боротьби з пожежами на початковій стадії. Це можуть бути пожежні кран-комплекти, вогнегасники, а також переносний пожежний інструмент.

Промисловість виробляє різні типи вогнегасників, такі як водопінні, водяні, газові і порошкові, причому порошкові вважаються більш перспективними за ефективністю та економічністю. В офісної будівлі прийнято тримати вогнегасники класів Е та В.

Вогнегасники класу Е (ОУ - вуглекислий) використовують для запобігання пожежі з електроносіїв, а клас В (ВП – порошковий) для запобігання пожежі з горючими рідинами (зазвичай лаки, фарби, аерозолі) та матеріалами які плавляться (основна небезпека від пластику, пластмаси).

Первинні засоби пожежогасіння розміщуються на пожежних щитах, розташованих на виробничій території з розрахунку один щит на 5000 м2, і мають червоний колір.

Вогнегасники маркуються буквами, що вказують на їх тип, та цифрами, які вказують на їх об'єм.

2.9 Висновки охорони праці

На сьогоднішній день охорона праці є ключовим елементом забезпечення безпеки на роботі. Щоб працівники могли працювати в здоровому та комфортному середовищі, було розроблено безліч документів, норм, заходів і інших засобів. За останні роки кількість трагічних випадків на робочих місцях зменшилася, але до досягнення нульових показників ще далеко. Тому постійно вдосконалюються заходи та розробляються нові рішення для забезпечення безпечної та комфортної праці.

ДОДАТОК А

Приклад коду ECB (Electronic Codebook) реалізований на C++

```
#include <openssl/aes.h>
#include <openssl/rand.h>
#include <iostream>
#include <cstring>
// Функція для обробки помилок
void handleErrors() {
    std::cerr << "Сталася помилка!" << std::endl;
    exit(1);
}
// Функція для шифрування в режимі ECB
void encryptECB(const unsigned char *plaintext, unsigned char *ciphertext, const unsigned char *key) {
    AES_KEY encryptKey;
    // Ініціалізація ключа для шифрування
    if (AES_set_encrypt_key(key, 128, &encryptKey) < 0) {
        handleErrors();
    }
    // Шифрування блоку даних
    AES_ecb_encrypt(plaintext, ciphertext, &encryptKey, AES_ENCRYPT);
}
// Функція для розшифрування в режимі ECB
void decryptECB(const unsigned char *ciphertext, unsigned char *plaintext, const unsigned char *key) {
    AES_KEY decryptKey;
    // Ініціалізація ключа для розшифрування
    if (AES_set_decrypt_key(key, 128, &decryptKey) < 0) {
```

```
        handleErrors();
    }
    // Розшифрування блоку даних
    AES_ecb_encrypt(ciphertext, plaintext, &decryptKey, AES_DECRYPT);
}
int main() {
    // Ключ AES повинен бути 16 байт (128 біт)
    unsigned char key[AES_BLOCK_SIZE];
    // Генерація випадкового ключа
    if (!RAND_bytes(key, AES_BLOCK_SIZE)) {
        handleErrors();
    }
    // Відкритий текст повинен бути кратний розміру AES_BLOCK_SIZE (16 байт)
    unsigned char plaintext[AES_BLOCK_SIZE] = "HELLO WORLD 123"; // 16 байт
    unsigned char ciphertext[AES_BLOCK_SIZE];
    unsigned char decryptedtext[AES_BLOCK_SIZE];
    std::cout << "Відкритий текст: " << plaintext << std::endl;
    // Шифрування відкритого тексту
    encryptECB(plaintext, ciphertext, key);
    std::cout << "Зашифрований текст: ";
    for (int i = 0; i < AES_BLOCK_SIZE; ++i) {
        std::cout << std::hex << (int)ciphertext[i];
    }
    std::cout << std::endl;
    // Розшифрування зашифрованого тексту
    decryptECB(ciphertext, decryptedtext, key);
    std::cout << "Розшифрований текст: " << decryptedtext << std::endl;
```

```
    return 0;
}
```

Приклад коду ECB (Electronic Codebook) реалізований на Java

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class EcbEncryption {
    // Метод для створення секретного ключа AES
    public static SecretKey generateKey(int n) throws Exception {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(n);
        SecretKey secretKey = keyGen.generateKey();
        return secretKey;
    }
    // Метод шифрування даних у режимі ECB
    public static String encrypt(String plainText, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
    // Метод розшифрування даних у режимі ECB
    public static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {
```

```
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        return new String(decryptedBytes);
    }
    public static void main(String[] args) {
        try {
            // Генерація ключа AES розміром 128 біт
            SecretKey secretKey = generateKey(128);
            // Оригінальний текст
            String plainText = "HELLO WORLD";
            System.out.println("Original text: " + plainText);
            // Шифрування тексту
            String encryptedText = encrypt(plainText, secretKey);
            System.out.println("Encrypted text: " + encryptedText);
            // Розшифрування тексту
            String decryptedText = decrypt(encryptedText, secretKey);
            System.out.println("Decrypted text: " + decryptedText);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Код 1.2.1 ECB (Electronic Codebook)

```
import tkinter as tk
from tkinter import messagebox
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
```

```
import os


class ECBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)

    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Electronic Codebook (ECB)."""
        cipher = AES.new(self.key, AES.MODE_ECB)  # Ініціалізація AES шифру в режимі ECB
        padded_plaintext = pad(plaintext, self.block_size)  # Додавання відступів до відкритого тексту
        ciphertext = cipher.encrypt(padded_plaintext)  # Шифрування тексту
        return ciphertext

    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Electronic Codebook (ECB)."""
        cipher = AES.new(self.key, AES.MODE_ECB)  # Ініціалізація AES шифру в режимі ECB
        decrypted_padded_text = cipher.decrypt(ciphertext)  # Розшифровка тексту
        plaintext = unpad(decrypted_padded_text, self.block_size)  # Видалення відступів із
розшифрованого тексту
        return plaintext


# Створюємо екземпляр системи блочного шифрування
ecb_cipher = ECBCipher()


def encrypt_text():
    plaintext = entry_plaintext.get().encode()
    ciphertext = ecb_cipher.encrypt(plaintext)
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())
        plaintext = ecb_cipher.decrypt(ciphertext)
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# Створення GUI за допомогою Tkinter
root = tk.Tk()
root.title("ECB Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Plaintext:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
```

```
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Ciphertext:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Encrypt", command=encrypt_text)
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)

btn_decrypt = tk.Button(root, text="Decrypt", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)

# Запуск основного циклу GUI
root.mainloop()
```

Код 1.2.2 CBC (Cipher Block Chaining)

```
import tkinter as tk
from tkinter import messagebox
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import os


class BlockCipherSystem:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)

    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Cipher Block Chaining (CBC)."""
        iv = os.urandom(self.block_size)  # Генерація випадкового вектора ініціалізації (IV)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)  # Ініціалізація AES шифру в режимі CBC
        padded_text = pad(plaintext, self.block_size)  # Доповнення тексту до кратності розміру блоку
        ciphertext = cipher.encrypt(padded_text)  # Шифрування тексту
        return iv + ciphertext  # Повертаємо IV разом із шифротекстом

    def decrypt(self, ciphertext):
        """Розшифровування тексту в режимі Cipher Block Chaining (CBC)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_CBC, iv)  # Ініціалізація AES шифру в режимі CBC
        padded_text = cipher.decrypt(actual_ciphertext)  # Розшифровування тексту
        plaintext = unpad(padded_text, self.block_size)  # Видалення доповнення
        return plaintext
```

```
def encrypt_text():
    plaintext = entry_plaintext.get().encode()  # Отримуємо текст із поля введення та кодуємо в байти
    ciphertext = bcs.encrypt(plaintext)  # Шифруємо текст
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())  # Вставляємо зашифрований текст у полі введення


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())  # Отримуємо шифротекст із поля введення та декодуємо з hex
        plaintext = bcs.decrypt(ciphertext)  # Розшифровуємо текст
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())  # Вставляємо розшифрований текст у полі введення
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# Створюємо екземпляр системи блочного шифрування
bcs = BlockCipherSystem()

# Створення GUI за допомогою Tkinter
root = tk.Tk()
root.title("Block Cipher Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Plaintext:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Ciphertext:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Encrypt", command=encrypt_text)
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)

btn_decrypt = tk.Button(root, text="Decrypt", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)

# Запуск основного циклу GUI
root.mainloop()
```

Код 1.2.3 CFB (Cipher Feedback)

```
import tkinter as tk
from tkinter import messagebox
from Crypto.Cipher import AES
import os


class CFBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.iv = os.urandom(AES.block_size)  # Генерація випадкового вектора ініціалізації
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)

    def encrypt(self, plaintext):
        """Шифрование текста в режиме Cipher Feedback (CFB)."""
        cipher = AES.new(self.key, AES.MODE_CFB, self.iv)  # Ініціалізація AES шифру в режимі CFB
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return self.iv + ciphertext  # Повертаємо IV разом із шифротекстом

    def decrypt(self, ciphertext):
        """Розшифрування тексту у режимі Cipher Feedback (CFB)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_CFB, iv)  # Ініціалізація AES шифру в режимі CFB
        plaintext = cipher.decrypt(actual_ciphertext)  # Розшифрування тексту
        return plaintext


# Створюємо екземпляр системи блочного шифрування
cfb_cipher = CFBCipher()


def encrypt_text():
    plaintext = entry_plaintext.get().encode()
    ciphertext = cfb_cipher.encrypt(plaintext)
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())
        plaintext = cfb_cipher.decrypt(ciphertext)
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# Створення GUI за допомогою Tkinter
root = tk.Tk()
```

```
root.title("CFB Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Plaintext:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Ciphertext:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Encrypt", command=encrypt_text)
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)

btn_decrypt = tk.Button(root, text="Decrypt", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)

# Запуск основного циклу GUI
root.mainloop()
```

Код 1.2.4 OFB (Output Feedback)

```
import tkinter as tk
from tkinter import messagebox
from Crypto.Cipher import AES
import os


class OFBCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.block_size = AES.block_size  # Розмір блоку AES (16 байт)

    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Output Feedback (OFB)."""
        iv = os.urandom(self.block_size)  # Генерація випадкового вектора ініціалізації (IV)
        cipher = AES.new(self.key, AES.MODE_OFB, iv)  # Ініціалізація AES шифру в режимі OFB
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return iv + ciphertext  # Повертаємо IV разом із шифротекстом

    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Output Feedback (OFB)."""
        iv = ciphertext[:self.block_size]  # Вилучення IV з початку шифротексту
        actual_ciphertext = ciphertext[self.block_size:]  # Частина, що залишилася, - сам шифротекст
        cipher = AES.new(self.key, AES.MODE_OFB, iv)  # Ініціалізація AES шифру в режимі OFB
        plaintext = cipher.decrypt(actual_ciphertext)  # Розшифрування тексту
        return plaintext
```

```
# Створюємо примірник системи блочного шифрування
ofb_cipher = OFBCipher()


def encrypt_text():
    plaintext = entry_plaintext.get().encode()  # Отримуємо текст із поля введення та кодуємо в байти
    ciphertext = ofb_cipher.encrypt(plaintext)  # Шифруємо текст
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())  # Вставляємо зашифрований текст у поле введення


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())  # Отримуємо шифротекст з поля введення та декодуємо з hex
        plaintext = ofb_cipher.decrypt(ciphertext)  # Розшифровуємо текст
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())  # Вставляємо розшифрований текст у полі введення
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# Створення GUI за допомогою Tkinter
root = tk.Tk()
root.title("OFB Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Plaintext:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Ciphertext:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Encrypt", command=encrypt_text)
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)

btn_decrypt = tk.Button(root, text="Decrypt", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)

# Запуск основного циклу GUI
root.mainloop()
```

Код 1.2.5 CTR (Counter)

```
import tkinter as tk
from tkinter import messagebox
```

```
from Crypto.Cipher import AES
import os


class CTRCipher:
    def __init__(self, key_size=16):
        """Ініціалізація системи блокового шифрування."""
        self.key = os.urandom(key_size)  # Генерація випадкового ключа
        self.nonce = os.urandom(8)  # Генерація випадкового значення nonce (8 байт)

    def encrypt(self, plaintext):
        """Шифрування тексту в режимі Counter (CTR)."""
        cipher = AES.new(self.key, AES.MODE_CTR, nonce=self.nonce)  # Ініціалізація AES шифру в
режимі CTR
        ciphertext = cipher.encrypt(plaintext)  # Шифрування тексту
        return ciphertext

    def decrypt(self, ciphertext):
        """Розшифрування тексту в режимі Counter (CTR)."""
        cipher = AES.new(self.key, AES.MODE_CTR, nonce=self.nonce)  # Ініціалізація AES шифру в
режимі CTR
        plaintext = cipher.decrypt(ciphertext)  # Розшифрування тексту
        return plaintext


# Створюємо екземпляр системи блочного шифрування
ctr_cipher = CTRCipher()


def encrypt_text():
    plaintext = entry_plaintext.get().encode()
    ciphertext = ctr_cipher.encrypt(plaintext)
    entry_ciphertext.delete(0, tk.END)
    entry_ciphertext.insert(0, ciphertext.hex())


def decrypt_text():
    try:
        ciphertext = bytes.fromhex(entry_ciphertext.get())
        plaintext = ctr_cipher.decrypt(ciphertext)
        entry_plaintext.delete(0, tk.END)
        entry_plaintext.insert(0, plaintext.decode())
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")


# Створення GUI за допомогою Tkinter
root = tk.Tk()
root.title("CTR Encryption System")

# Поле введення для відкритого тексту
tk.Label(root, text="Plaintext:").grid(row=0, column=0, padx=10, pady=10)
entry_plaintext = tk.Entry(root, width=50)
```

```
entry_plaintext.grid(row=0, column=1, padx=10, pady=10)

# Поле введення для зашифрованого тексту
tk.Label(root, text="Ciphertext:").grid(row=1, column=0, padx=10, pady=10)
entry_ciphertext = tk.Entry(root, width=50)
entry_ciphertext.grid(row=1, column=1, padx=10, pady=10)

# Кнопки для шифрування та розшифрування
btn_encrypt = tk.Button(root, text="Encrypt", command=encrypt_text)
btn_encrypt.grid(row=2, column=0, padx=10, pady=10)

btn_decrypt = tk.Button(root, text="Decrypt", command=decrypt_text)
btn_decrypt.grid(row=2, column=1, padx=10, pady=10)

# Запуск основного циклу GUI
root.mainloop()
```

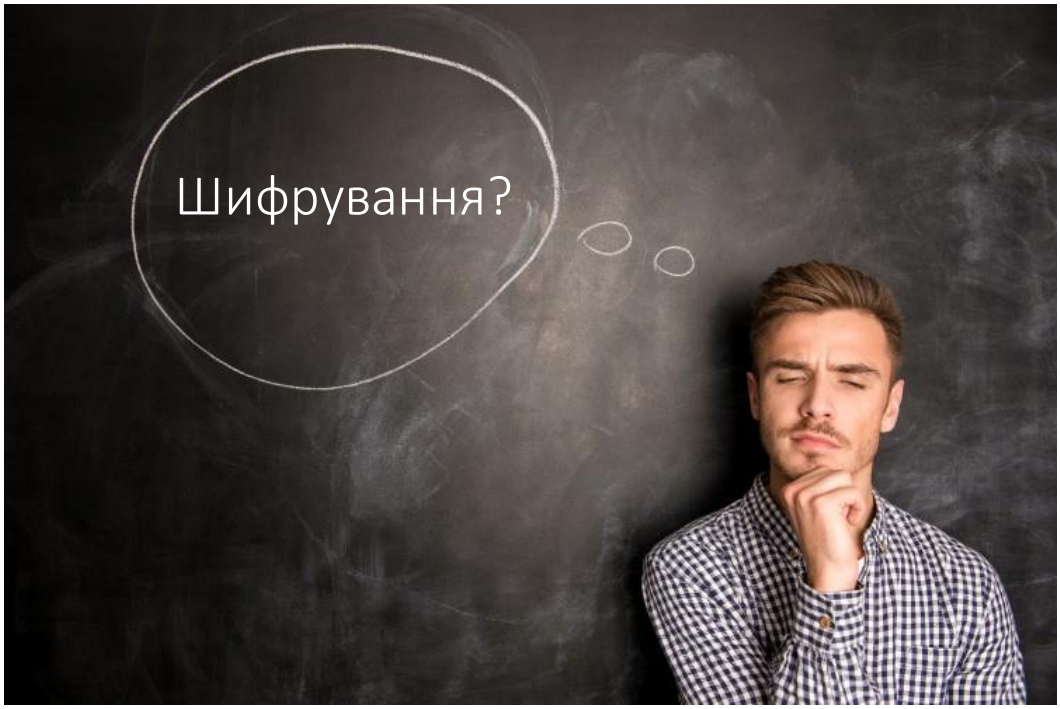
ДОДАТОК Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Дослідження блокових криптографічних систем шифрування інформації на основі мови Python

Досліджувач: Мурзін О. В

Керівник роботи к.т.н. Кільдашев В. Й.



Основні поняття шифрування

1.Відкритий текст (plaintext): Початкові дані, які потрібно зашифрувати.

2.Шифротекст (ciphertext): Дані після шифрування, які важко або неможливо прочитати без відповідного ключа.

3.Ключ: Спеціальна інформація, необхідна для шифрування і дешифрування даних.

4.Алгоритм шифрування : Метод або процедура, яка використовує ключ для перетворення відкритого тексту в шифротекст і навпаки.

Симетричне шифрування

•Використовує один і той самий ключ для шифрування і дешифрування даних.

•Приклади AES (Advanced Encryption Standard), DES (Data Encryption Standard), RC4.

•Плюси: Висока швидкість шифрування і дешифрування

•Мінуси: Проблеми з безпечною передачею ключами між сторонами

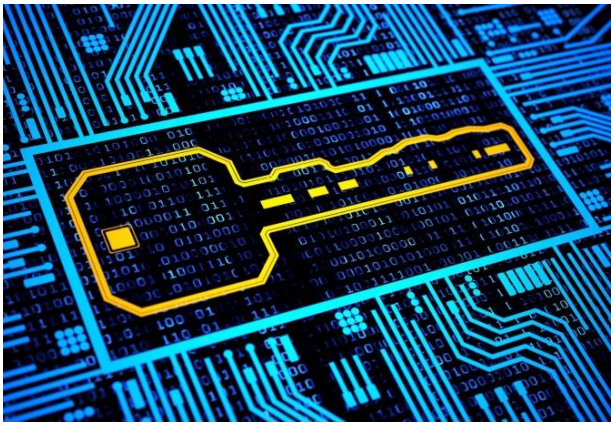
Асиметричне шифрування

•Використовує пару ключів: відкритий ключ (для шифрування) і закритий ключ (для дешифрування).

•Приклади RSA, ECC (Elliptic Curve Cryptography).

•Плюси: Підвищена безпека при передачі даних.

•Мінуси: Повільніше порівняно з симетричним шифруванням



Блочне шифрування

Основні принципи блочного шифрування

1. **Розмір блоку**: Дані розбиваються на блоки фіксованого розміру. Якщо розмір даних не кратний розміру блоку, використовується доповнення, щоб заповнити останній блок до потрібного розміру.
2. **Ключ**: Один і той самий ключ використовується для шифрування кожного блоку даних.
3. **Алгоритм шифрування** Блочні шифри використовують складні математичні операції для перетворення кожного блоку відкритого тексту в шифротекст

Режими роботи блочного шифрування

- Блочні шифри можуть працювати в різних режимах, кожен з яких має свої особливості і забезпечує різні рівні безпеки та продуктивності

Режими блочного шифрування

- Електронну книгу кодів (ECB);
- Ланцюгування шифроблоків (CBC);
- Зворотний зв'язок по шифротексту (CFB);
- Зворотний зв'язок по виходу (OFB);
- Лічильник (CTR)

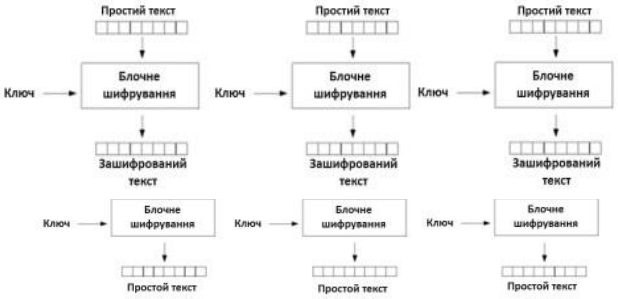
Electronic Codebook (ECB)

Як працює режим ECB

1.Розбиття на блоки : Дані розбиваються на блоки фіксованого розміру, наприклад, 128 біт для AES або 64 біт для DES.

2.Шифрування кожного блоку : Кожен блок відкритого тексту шифрується окремо за допомогою одного і того самого ключа, утворюючи відповідний блок шифротексту.

3.Збір зашифрованих блоків : Зашифровані блоки збираються разом, утворюючи кінцевий шифротекст .



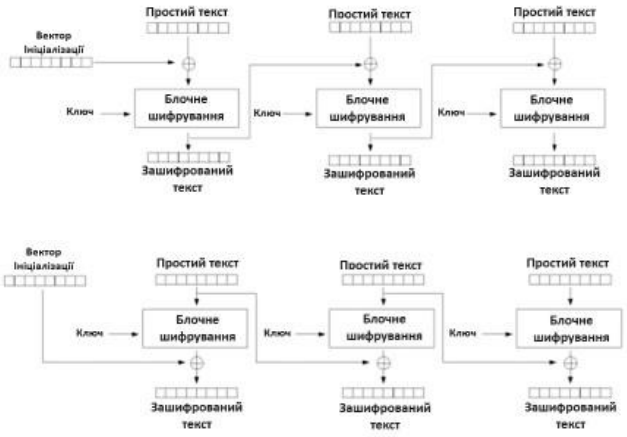
Cipher Block Chaining (CBC)

Як працює режим CBC

1.Розбиття на блоки : Дані розбиваються на блоки фіксованого розміру, наприклад, 128 біт для AES або 64 біт для DES.

2.Вектор ініціалізації (IV): Для шифрування першого блоку використовується вектор ініціалізації (IV). Це випадковий або псевдовипадковий блок даних тієї ж довжини, що й блоки відкритого тексту.

3.Шифрування кожного блоку : Кожен блок відкритого тексту перед шифруванням XOR-иться з попереднім блоком шифротексту.



Cipher Feedback (CFB)

Як працює режим CFB

Ініціалізація :

Використовується вектор ініціалізації (IV) тієї ж довжини, що і блок шифру.

Перший блок шифротексту формується шляхом шифруванняIV.

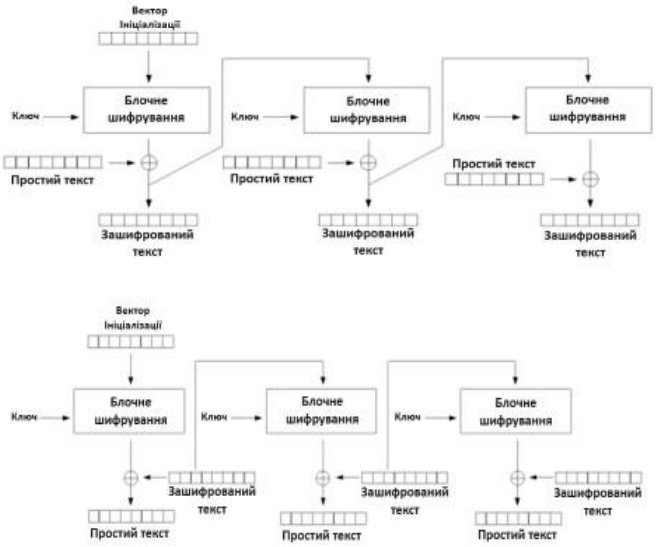
Шифрування першого блоку :

ШифруєтьсяIVдля отримання псевдовипадкового виходу.

Перший блок відкритого текстуXOR-ується з псевдовипадковим виходом для утворення першого блоку шифротексту.

Подальше шифрування :

Кожен наступний блок відкритого тексту XOR-ується з виходом шифрування попереднього блоку шифротексту.



Output Feedback (OFB)

Як працює режим OFB

Ініціалізація :

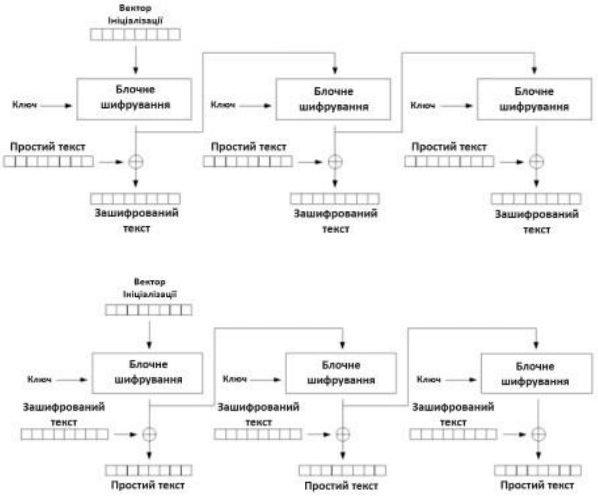
Вибирається випадковий вектор ініціалізації(IV).
Перший блок вихідного потоку генерується шляхом шифруванняIV.

Шифрування першого блоку :

ШифруєтьсяIVдля отримання першого блоку вихідного потоку.
Перший блок відкритого текстуXOR-ується з першим блоком вихідного потоку для утворення першого блоку шифротексту.

Подальше шифрування:

Кожен наступний блок вихідного потоку генерується шляхом шифрування попереднього блоку вихідного потоку.
Кожен блок відкритого текстуXOR-ується з відповідним блоком вихідного потоку.



Counter (CTR)

Як працює режим CTR

Ініціалізація:

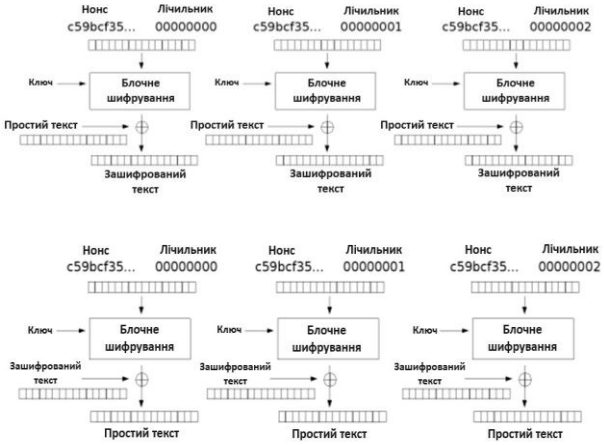
Вибирається унікальне початкове значення лічильника (nonce).Значення лічильника інкрементується для кожного блоку.

Генерація псевдовипадкових чисел:

Для кожного блоку шифрується значення лічильника для отримання псевдовипадкового виходу.

Шифрування:

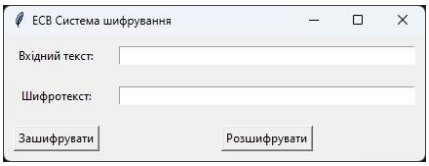
Кожен блок відкритого тексту XOR-ується з відповідним псевдовипадковим числом для утворення блоку шифротексту.

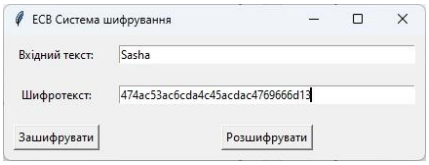


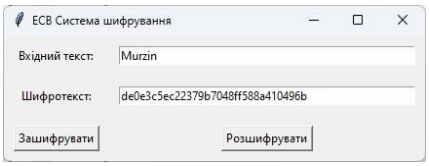
Таблиця зрівняння режимів блочного шифрування

Характеристика	ECB (Electronic Codebook)	CBC (Cipher Block Chaining)	CFB (Cipher Feedback)	OFB (Output Feedback)	CTR (Counter)
Принципроботи	Кожен блок шифрується незалежно	Кожен блок XOR-ується з попереднім шифротекстом перед шифруванням	Шифротекст попереднього блоку XOR-иться з наступним блоком відкритого тексту перед шифруванням	Ключовий потік генерується шифруванням вектора ініціалізації або зашифрованого попереднього блоку	Ключовий потік генерується шифруванням лічильника
Параллельнаяобработка	Так	Ні	Ні	Так	Так
Обробкапомилок	Помилка впливає лише на один блок	Помилка поширюється на наступний блок	Помилка поширюється на наступний блок	Помилка не поширюється	Помилка не поширюється
Патерни в шифротексті	Може мати патерни за однакових блоків дани	Не має	Не має	Не має	Не має
Складністьреалізації	Проста	Помірна	Помірна	Проста	Проста
НеобхідністьIV	Ні	Так	Так	Так	Так (nonce + лічильник)
Ініціалізація	Не потребує	Потрібен випадковий IV	Потрібен випадковий IV	Потрібен випадковий IV	Потрібен випадковий nonc
Переваги	Простота та швидка реалізація	Висока безпека	Потокове шифрування, підходить для потоків дани	помилки не поширюються	Потокове шифрування, паралельна обробка, висока ефективність
Характеристика	ECB (Electronic Codebook)	CBC (Cipher Block Chaining)	CFB (Cipher Feedback)	OFB (Output Feedback)	CTR (Counter)
Недоліки	Низька безпека через можливі патерни.	Складніше реалізація помилки поширюються на наступний блок	Складніше реалізація помилки поширюються на наступний блок	Можливі атаки на основі потоків, складність управлін IV	Потрібен унікальний лічильник для кожного блоку, складність управління nonce

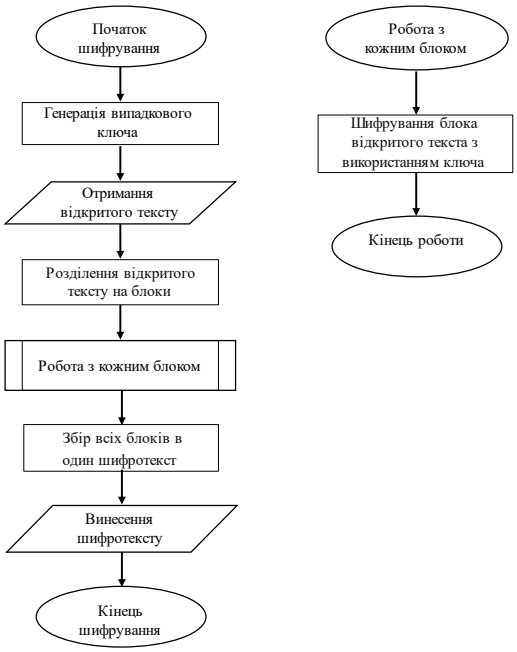
Реалізація ECB на мові програмування Python







Блок-схема алгоритма режима шифрування ECB реалізованого на Python



ВСП «Одеський технічний фаховий коледж ОНАХТ»

ВІДГУК

Керівника про кваліфікаційної роботи бакалавра студента

Мурзіна Олександра Володимировича

(прізвище, ім'я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Тема кваліфікаційної роботи бакалавра

Дослідження блокових криптографічних систем шифрування інформації на основі мови Python

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) Обсяг і якість виконання роботи (графічного матеріалу і розрахунково-пояснювальної записки)

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми дипломного проекту, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у дипломному проекті

Графічний матеріал виконано якісно, у достатньому обсязі. Графічний матеріал наочно демонструє результати роботи.

б) Самостійність роботи над проектом (роботою)

Студент самостійно обрав напрям та тематику дипломного проекту. Провів аналіз існуючих рішень і зробив необхідні висновки для реалізації проекту. Виявив навички самостійно опрацьовувати новий матеріал та виконувати пошук необхідної літератури та інших джерел інформації

в) Теоретична підготовка дипломника _____

відповідає вимогам, що надаються до бакалавра зі спеціальності _____

«Комп'ютерна Інженерія» _____

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва _____

У кваліфікаційній роботі досліджується блокові криптографічні системи шифрування даних. Розглянуто види блочних режимів шифрування та їх порівняння. _____

Оцінка розрахункової частини _____ добре

Оцінка графічної частини _____ добре

Загальна оцінка _____ добре

Прізвище, ім`я, по батькові Кільдішев Віталій Йосипович

Місто роботи і посада керівника проекту к.т.н., доцент кафедри кібербезпеки та технічного захисту інформації ДУІТЗ

Підпис _____

« 10 » 06 2024р.

ВСП «Одеський технічний фаховий коледж ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

Комп'ютерних систем відділення

Мурзіна Олександра

(прізвище, ім`я та по батькові)

Спеціальність 123 «Комп'ютерна інженерія»

Керівник роботи

Кільдішев Віталій Йосипович

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи бакалавра:

Дослідження блокових криптографічних систем шифрування інформації на основі мови Python

Обсяг розрахунково-пояснювальної записки 87 сторінок

Обсяг графічної частини проекту 14 аркушів

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) Висновок про ступінь відповідальності виконаного кваліфікаційної роботи бакалавра завданню

Робота відповідає технічному завданню до дипломного проекту. Виконана у відповідності з вимогами.

б) Характеристика виконання кожного розділу проекту ступеню використання дипломником останніх досягнень науки та техніки, передових методів на виробництві

При виконанні дипломного проекту студент продемонстрував уміння використовувати останні досягнення науки та техніки, уміння працювати з літературою. Так, студент грамотно дослідив та проаналізував методів та засобів блочного шифрування.

в) Оцінка якості виконання графічної частини проекту (роботи) і пояснювальної записки

Графічна частина відповідає вимогам, виконана якісно та відображає основні елементи проектування системи. Розглянуто роль і значення захисту інформації у сфері кібербезпеки, з акцентом на важливість шифрування даних.

г) Перелік позитивних якостей кваліфікаційної роботи бакалавра

Тема випускної роботи є актуальною, виконана у достатньому обсязі, відповідно до поставленого завдання.

д) Основні недоліки кваліфікаційної роботи бакалавра

1. Пояснювальна записка недостатньо структурована, етапи розробки недостатньо обґрунтовані.
2. Блок-схеми алгоритмів не відображують суть роботи блокових криптосистем
3. Наявні помилки оформлення пояснювальної записки

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім`я по батькові Селіванова Алла Віталіївна

Місце роботи і посада рецензента Одеський національний технологічний університет, декан факультету комп'ютерної інженерії, програмування та інформаційних систем

МІНІСТЕРСТВО ОСВІТИ І НАУКИ ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ, ПРОГРАМУВАННЯ ТА КІБЕРЗАХИСТУ

20.06.2024р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016351447

Дата перевірки:
12.06.2024 10:50:43 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 10:59:02 EEST

ID користувача:
100011688

Назва документа: 2БКС-28_Олександр_Мурзин

Кількість сторінок: 61 Кількість слів: 7990 Кількість символів: 57924 Розмір файлу: 649.07 KB ID файлу: 1016154992

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.93%
Схожість

Найбільша схожість: 0.78% з Інтернет-джерелом (https://keisuke-shikaku.hatenablog.com/?page=1688079600)

3.93% Джерела з Інтернету	311	Сторінка 63

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	4

Підозріле форматування	17 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Мурзін Олександр Володимирович
здобувач освіти гр. 2БКС-28, та

Кільдішев Віталій Йосипович,
керівник випускної кваліфікаційної роботи,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Дослідження блокових криптографічних систем шифрування інформації на основі мови програмування Python» (автор роботи – Мурзін О. В, керівник роботи – Кільдішев В.Й.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Мурзін О. В./

Керівник _____ / Кільдішев В.Й./

«13» червня 2024 р.