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

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна інженерія»

Група: 2БКС-29

КВАЛІФІКАЦІЙНА РОБОТА

здобувача освіти денної форми навчання

БКС.29.22.000.КРБ

ЦИГАНОВА ВЛАДИСЛАВА СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-29**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **«Реалізація методу потокового шифрування на основі генератору хаосу»**

Проектний матеріал складається з пояснювальної записки на 64 сторінках та графічного (презентаційного) матеріалу на 11 аркушах (слайдах)

Виконавець _____ (Циганов В.С.)

Керівник проекту _____ (Кільдішев В.Й.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділенням _____ (Краснокутська К.Г.)

Захист «27» 06 202__ р. Протокол ЕК № 2

Оцінка ЕК 4 (добре) 185

Секретар ЕК _____

АНОТАЦІЯ

Метою даної роботи є розробка та реалізація методу потокового шифрування інформації з використанням генератора хаотичних послідовностей для підвищення рівня захисту даних під час їх передавання в інформаційно-комунікаційних системах.

Вивчено закономірності формування хаотичних сигналів і їх застосування у криптографії, зокрема властивості детермінованого хаосу, які забезпечують непередбачуваність та високу чутливість до початкових умов. Проаналізовано особливості впровадження хаосу в процес шифрування з урахуванням вимог до сучасних систем інформаційної безпеки.

Отримані кількісні показники якості шифрування, зокрема ентропія, середнє значення та дисперсія бітів шифрограми, що свідчать про наближення отриманих результатів до характеристик випадкових процесів. Проведено статистичне тестування згідно з вимогами NIST, що підтвердило криптографічну стійкість реалізованої системи.

Створено програмну модель потокового шифратора на основі генератора хаосу, яка реалізована мовою програмування Python з використанням математичних методів генерації хаотичних послідовностей. Проведено дослідження її ефективності у порівнянні з традиційними методами шифрування.

Розглянуто питання з охорони праці та техніки безпеки під час роботи з комп’ютерним обладнанням, а також при проведенні експериментальних досліджень, зокрема дотримання санітарно-гігієнічних вимог, правил електробезпеки та організації робочого місця.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"28" 05 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Циганова Владислава Сергійовича*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Реалізація методу потокового шифрування на основі генератору хаосу*

затверджена наказом по коледжу від "____" _____ 20____ р. №_____

2. Термін здачі студентом кваліфікаційної роботи _____

3. Вихідні дані до роботи *1. Вибір математичної моделі генератора хаотичних сигналів; 2. Аналіз криптостійкості запропонованого методу; 3. Реалізація запропонованого методу в програмному середовищі; 4. Порівняння з традиційними потоковими шифрами*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Огляд перспектив розвитку криптографічних протоколів; Аналіз потокових шифрів та генераторів хаосу; Розробка методу потокового шифрування на основі генератора хаосу; Реалізація запропонованого методу в програмному середовищі

5. Перелік графічного матеріалу (слайдів мультимедійної презентації) *Функціональні, криптографічні та експлуатаційні вимоги до потокових систем шифрування; Вимоги до генерації псевдовипадкових послідовностей потокового шифрування; Порівняння класичних та сучасних потокових шифрів; Реалізація центрованого хаотичного процесу; Порівняння хаотичних та класичних PRNG; Результат перевірки алгоритму потокового шифрування і розшифрування; Результати перетворення даних в процесі потокового шифрування; Узагальнені статистичні дані очікувань від шифрограми; Результат перевірки статистичних показників шифрограми потокового шифру*

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що їх стосуються

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Кільдішев В.Й.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник роботи *Кільдішев В.Й.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Вступ. Аналіз технічного завдання	21.05.2025	Виконав
2.	Огляд перспектив розвитку криптографічних протоколів	22.05.2025	Виконав
3.	Основні показники криптографічних протоколів	24.05.2025	Виконав
4.	Порівняльний аналіз блокових та потокових систем шифрування	26.05.2025	Виконав
5.	Аналіз потокових шифрів та генераторів хаосу	28.05.2025	Виконав
6.	Вимоги до генераторів ключових послідовностей	30.05.2025	Виконав
7.	Хаотичні системи як джерело випадкових послідовностей	02.06.2025	Виконав
8.	Розробка методу потокового шифрування на основі генератора хаосу	05.06.2025	Виконав
9.	Алгоритм шифрування та дешифрування даних	07.06.2025	Виконав
10.	Аналіз криптостійкості запропонованого методу	09.06.2025	Виконав
11.	Реалізація запропонованого методу в програмному середовищі	10.06.2025	Виконав
12.	Розробка питань з охорони праці та техніки безпеки	12.06.2025	Виконав
13.	Підготовка матеріалів мультимедійної презентації	14.06.2025	Виконав

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 29. 22 000. 00 КРБ	Випускна робота		
А4			БКС 29. 22 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 29. 22 000. 00 КРБ		
Розробив		Циганов В.С.			Реалізація методу потокового шифрування на основі генератору хаосу	Літ.	Аркуш 4 / Аркушів 67
Перевірив		Кільдішев В.Й.				Н Д П	
Н. Контр.		Петрашова В.І.		10.05.15		ВСП "ОТФК ОНТУ" гр.2БКС-29	
Затверд.		Іванова Л.В.		2015			

ЗМІСТ

ВСТУП

У сучасному цифровому світі питання захисту інформації є надзвичайно актуальним. Зростаюча кількість кібератак та несанкціонованого доступу до конфіденційних даних вимагає розробки та впровадження надійних криптографічних методів. Одним із ефективних підходів до захисту інформації є потокове шифрування, яке забезпечує високий рівень безпеки завдяки використанню псевдовипадкових послідовностей.

Традиційні методи генерації ключових послідовностей, такі як лінійні зсувні регістри (LFSR) або нелінійні алгоритми, мають певні недоліки, зокрема обмежену ентропію та передбачуваність при недостатньо складному алгоритмі. Альтернативним підходом є використання хаотичних динамічних систем, які володіють важливими властивостями, такими як чутливість до початкових умов, детермінований хаос і складна структура фазового простору. Це дозволяє отримувати неперіодичні та криптостійкі послідовності, що можуть бути ефективно застосовані у потоковому шифруванні.

Використання хаотичних генераторів у криптографії є перспективним напрямом досліджень, оскільки вони можуть забезпечити високий рівень безпеки та захисту переданих даних. Дослідження методів побудови потокових шифрів на основі хаотичних систем сприяє розвитку сучасних криптографічних засобів захисту інформації.

Метою роботи є розробка та аналіз методу потокового шифрування на основі генератора хаосу для забезпечення надійного захисту інформації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз існуючих методів потокового шифрування та генерації ключових послідовностей;
- дослідити математичні моделі хаотичних генераторів та їх властивості;
- розробити метод шифрування на основі хаотичних послідовностей;
- реалізувати запропонований метод у програмному середовищі та провести його експериментальне дослідження;
- оцінити стійкість та ефективність запропонованого підходу.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд перспектив розвитку криптографічних протоколів

1.1.1 Історичний огляд розвитку криптографічних протоколів

Криптографія має багатовікову історію, яка бере свій початок ще з давніх цивілізацій. Вона пройшла шлях від простих шифрів до складних математичних алгоритмів і сучасних криптографічних протоколів, які забезпечують безпеку цифрових комунікацій.

Перші відомі системи шифрування з’явилися ще в Стародавньому Єгипті, Греції та Римі. Одним із найвідоміших методів шифрування був шифр Цезаря, який полягав у зсуві літер у тексті на фіксовану кількість позицій. Інші ранні методи включали використання скітали у Спарті (шифрування з використанням шкіряних або папірусних стрічок) та стеганографію – приховування тексту всередині інших повідомлень або об’єктів.

У середні віки та епоху Відродження криптографія отримала подальший розвиток. У 16 столітті французький дипломат Блез де Віженер запропонував шифр Віженера, який використовував поліалфавітне шифрування і значно підвищив криптостійкість повідомлень.

На початку XX століття з’явилися складніші механічні шифрувальні пристрої, зокрема машина Енігма, що використовувалася під час Другої світової війни для шифрування німецьких військових повідомлень. Вона базувалася на використанні роторів, які змінювали шифрувальну таблицю після кожного натискання клавіші.

З появою комп’ютерів у другій половині XX століття криптографія зазнала революційних змін. Виникли нові методи, засновані на різних математичних теоріях. У 1976 рік вчені Вітфілд Діффі та Мартін Геллман запропонували протокол обміну ключами (Diffie-Hellman), який дозволяв двом сторонам узгоджувати спільний секретний ключ через відкритий канал без необхідності попередньої передачі секретної інформації. В 1977 рік вчені Рональд Рівест, Аді Шамір і Леонард Адлеман розробили алгоритм RSA, який базується на

складності факторизації великих чисел і став основою для багатьох сучасних криптографічних систем. В 1985 рік вчені Віктор Міллер та Ніл Кобліц незалежно запропонували еліптичні криві в криптографії, що дозволили створити ефективніші та безпечніші криптографічні протоколи.

Сучасні криптографічні протоколи є основою безпеки інформаційних систем. Основними криптографічними протоколами, що використовуються в цифровій безпеці, є:

1) AES (Advanced Encryption Standard, 2001) – симетричний алгоритм шифрування, який став стандартом для урядів та комерційних структур;

2) TLS (Transport Layer Security, 1999) – протокол для захищеного передавання даних у мережах, що замінив застарілий SSL;

3) SHA (Secure Hash Algorithm) – серія алгоритмів хешування, які використовуються для перевірки цілісності даних;

4) Post-Quantum Cryptography – новий напрям, який розробляється для захисту інформації від квантових комп'ютерів.

Отже, криптографія пройшла довгий шлях розвитку від механічних пристроїв до складних математичних моделей. Подальші дослідження у цій сфері спрямовані на розробку нових алгоритмів, стійких до атак, зокрема з використанням квантових технологій.

1.1.2 Загальні вимоги до криптографічних протоколів

Криптографічні протоколи є основою сучасних систем захисту інформації, забезпечуючи конфіденційність, цілісність та автентифікацію даних під час їх передавання або зберігання. Для того щоб криптографічний протокол був ефективним і надійним, він повинен відповідати ряду вимог, які забезпечують його безпеку та стійкість до атак. До таких вимог слід віднести: конфіденційність; цілісність; автентифікація; стійкість до атак; ефективність і продуктивність; масштабованість і сумісність.

Основною функцією криптографічних протоколів є забезпечення конфіденційності переданої або збереженої інформації. Це означає, що доступ до

даних можуть отримати лише авторизовані користувачі. Конфіденційність забезпечується за допомогою:

1) симетричних алгоритмів шифрування (AES, ChaCha20) – коли один ключ використовується для шифрування та дешифрування;

2) асиметричних алгоритмів (RSA, ECC) – коли використовується пара відкритого та закритого ключів;

3) протоколів тунелювання (TLS, IPSec) – які забезпечують захищений канал зв’язку.

Цілісність даних означає, що інформація не була змінена або підроблена під час передавання чи зберігання. Для цього застосовуються:

1) алгоритми хешування (SHA-256, SHA-3) – створюють унікальний цифровий відбиток повідомлення;

2) механізми контролю цілісності (HMAC, цифрові підписи) – дозволяють перевіряти, чи не було повідомлення змінено.

Автентифікація гарантує, що обидві сторони зв’язку є тими, за кого себе видають. Для цього використовуються:

1) протоколи автентифікації (Kerberos, OAuth, OpenID);

2) цифрові сертифікати (X.509, PKI) – для перевірки довіреності ключів;

3) механізми двофакторної автентифікації (2FA).

Протоколи повинні бути стійкими до широкого спектра атак, включаючи:

1) атаки грубої сили (Brute-force attacks) – захист забезпечується довжиною ключів (наприклад, AES-256);

2) атаки "людина посередині" (MITM) – запобігається за рахунок цифрових підписів та сертифікатів;

3) криптоаналітичні атаки – запобігається шляхом використання складних математичних перетворень.

Криптографічні протоколи повинні забезпечувати високий рівень безпеки без значного навантаження на системні ресурси. Важливими факторами є:

1) час генерації ключів та виконання шифрування/дешифрування;

2) споживання енергії (особливо важливо для мобільних пристроїв);

3) можливість апаратного прискорення (AES-NI, TPM).

Масштабованість і сумісність (Scalability and Compatibility) забезпечується за рахунок того, що протоколи повинні бути адаптивними до різних платформ і пристроїв, а також підтримувати інтеграцію з іншими технологіями, такими як VPN, хмарні сервіси та блокчейн.

Отже, загальні вимоги до криптографічних протоколів включають конфіденційність, цілісність, автентифікацію, стійкість до атак, ефективність та масштабованість. Відповідність цим вимогам забезпечує надійний захист інформації в сучасних інформаційних системах.

1.1.3 Основні показники криптографічних протоколів

Оцінка ефективності криптографічних протоколів базується на ряді ключових показників, які визначають рівень їхньої безпеки, швидкодії та стійкості до атак. Вибір криптографічного алгоритму або протоколу залежить від конкретних вимог до захисту інформації, апаратних можливостей системи та характеру загроз. Можна визначити наступні важливі показники ефективності криптографічних протоколів: рівень криптографічної стійкості; часові характеристики; алгоритмічна ефективність; масштабованість та адаптивність; стійкість до квантових атак.

Рівень криптографічної стійкості визначає, наскільки протокол захищений від криптоаналітичних атак. Основні параметри є:

1) довжина ключа (Key Length) – чим довший ключ, тим складніше зламати алгоритм методом перебору. Наприклад, для симетричних шифрів стандартом є AES-256, а для асиметричних алгоритмів, таких як RSA, рекомендується довжина 2048 або 4096 біт;

2) стійкість до атак (Resistance to Attacks) – сучасні криптографічні протоколи повинні бути захищені від атак, таких як атаки грубої сили (brute-force), атаки аналізу частотності та квантові атаки.

Часові характеристики (Performance Metrics) надають оцінку продуктивності криптографічного протоколу включає наступні параметри:

1) час генерації ключів – важливий для асиметричних алгоритмів, таких як RSA та ECC;

2) швидкість шифрування/розшифрування – напряму впливає на продуктивність системи. Наприклад, AES у режимі GCM швидший за AES у режимі CBC завдяки апаратному прискоренню;

3) затримка під час встановлення безпечного з'єднання – визначає час, необхідний для автентифікації та обміну ключами (наприклад, у протоколах TLS/SSL).

Алгоритмічна ефективність припускають те, що протоколи мають бути оптимізовані для різних типів пристроїв:

1) ресурсомісткість (Resource Usage) – важливий показник для пристроїв з обмеженими ресурсами (смартфони, IoT, вбудовані системи);

2) можливість апаратного прискорення – деякі алгоритми, такі як AES-NI або SHA-3, можуть виконуватися швидше завдяки апаратним інструкціям процесора.

Масштабованість та адаптивність (Scalability and Adaptability) те, що криптографічний протокол має бути придатним для застосування у різних середовищах:

1) підтримка різних платформ – сумісність із сучасними операційними системами, мобільними пристроями, хмарними технологіями;

2) гнучкість налаштувань – можливість зміни параметрів без необхідності повного перезапуску протоколу.

Стійкість до квантових атак припускає, що з появою квантових комп'ютерів традиційні алгоритми, такі як RSA і ECC, можуть стати вразливими. Тому розробляються нові алгоритми:

– Lattice-based cryptography (NTRU, Kyber);

– Hash-based signatures (XMSS, SPHINCS+);

– Code-based cryptography (McEliece).

Отже, основні показники криптографічних протоколів включають рівень стійкості до атак, продуктивність, алгоритмічну ефективність, масштабованість

та захищеність від квантових загроз. Вибір конкретного протоколу залежить від сфери застосування, вимог до безпеки та обчислювальних можливостей системи.

1.1.4 Порівняльний аналіз блокових та потокових систем шифрування

Сучасна криптографія використовує два основних підходи до симетричного шифрування даних: блокове та потокове шифрування. Обидва підходи мають свої особливості, переваги та недоліки, які визначають їхню доцільність у різних сферах застосування.

Блокові шифри працюють шляхом поділу вихідного повідомлення на блоки фіксованого розміру (наприклад, 64 або 128 біт), які шифруються окремо за допомогою криптографічного алгоритму. Основні блокові алгоритми наступні:

1) DES (Data Encryption Standard, 56-бітний ключ) – застарілий через низьку криптостійкість;

2) 3DES (Triple DES) – триразове шифрування DES, але все ще вразливий для атак;

3) AES (Advanced Encryption Standard, 128, 192, 256-бітні ключі) – сучасний стандарт;

4) Blowfish, Twofish, Serpent – альтернативи AES.

Блокові шифри можуть працювати в різних режимах, що впливає на їхню безпеку:

1) ECB (Electronic Codebook) – кожен блок шифрується незалежно, що робить його вразливим до атак;

2) CBC (Cipher Block Chaining) – кожен наступний блок залежить від попереднього, що підвищує безпеку;

3) CFB, OFB, CTR – режими, що імітують потокове шифрування.

Можна визначити наступні переваги блокових шифрів:

1) висока стійкість до атак при використанні безпечних режимів (CBC, GCM);

2) підходять для зберігання зашифрованих даних;

3) широка підтримка апаратного прискорення (AES-NI).

До недоліків блокових шифрів можна віднести наступне:

– вимагають доповнення (padding) для роботи з даними, розмір яких не кратний довжині блоку.

– вищі затримки у випадку потокового передавання інформації.

Потокові шифри працюють з даними побітово або побайтово, використовуючи псевдовипадкову послідовність (ключовий потік), яка комбінується з відкритим текстом за допомогою операції XOR.

Основні потокові алгоритми:

1) RC4 (Rivest Cipher 4) – раніше широко використовувався, але зараз вважається небезпечним;

2) Salsa20, ChaCha20 – сучасні безпечні потокові шифри;

3) Grain, Trivium – легкі потокові шифри для IoT.

До переваг потокових шифрів можна віднести наступне:

– висока швидкість шифрування (немає необхідності в розбитті на блоки);

– ефективність для потокового передавання даних (VoIP, VPN);

– не потребують доповнення (padding).

До недоліків потокових шифрів можна віднести наступне:

– вразливі до атак, якщо один ключ використовується більше ніж один раз;

– менша підтримка апаратного прискорення в порівнянні з AES.

Порівняльний аналіз блокового та потокового шифрування надано в табл. 1.1.

Отже, обидва типи шифрування мають свої переваги і застосовуються в різних сценаріях. Блокові шифри (наприклад, AES) є надійними та широко використовуються для збереження даних, тоді як потокові шифри (ChaCha20) забезпечують швидке і безпечне передавання інформації в реальному часі.

Таблиця 1.1. Порівняльний аналіз блокового та потокового шифрування

Параметр	**Блокові шифри (AES, DES)**	**Потокові шифри (RC4, ChaCha20)**
Метод обробки	Шифрують блоки фіксованого розміру	Шифрують дані побітово або побайтово
Швидкість роботи	Вища для великих обсягів даних	Вища для потокових даних
Безпека	Висока у відповідних режимах (AES-256 у GCM)	Висока при одноразовому використанні ключового потоку
Застосування	Диски, файли, бази даних	Мережеві комунікації, стрімінг, VPN
Вимоги до пам'яті	Вищі	Нижчі
Стійкість до атак	Стійкіші при правильному режимі	Вразливі до повторного використання ключів

1.2 Аналіз вимог до потокових шифрів та генераторів хаосу

1.2.1. Основні методи потокового шифрування

Потокове шифрування є одним із ключових методів криптографічного захисту даних, який широко використовується в інформаційних системах, що працюють у режимі реального часу. На відміну від блокових шифрів, потокові алгоритми працюють з даними побітово або побайтово, що забезпечує високу швидкість обробки та низькі затримки.

Методи потокового шифрування базуються на генерації псевдовипадкової послідовності (ключового потоку), яка комбінується з відкритим текстом за допомогою операції XOR. Головною умовою безпеки є використання унікального ключа для кожного шифрування.

У синхронних потокових шифрах (наприклад, RC4, Salsa20) ключовий потік генерується незалежно від відкритого тексту та залежить лише від секретного ключа та початкового вектора (IV). Принцип роботи у таких шифрах полягає в наступному:

1) генерується псевдовипадковий потік бітів (keystream) на основі секретного ключа;

2) потік XOR-иться з відкритим текстом, утворюючи зашифроване повідомлення;

3) для дешифрування виконується аналогічна операція XOR.

Приклад роботи потокового шифрування:

– якщо відкритий текст: 11010100;

– ключовий потік: 10111001;

– зашифрований текст (XOR): 01101101.

До переваги потокового шифрування наступні можна віднести високу швидкість роботи та ефективність у мережевих протоколах (SSL/TLS, VPN).

До недоліків можна віднести уразливість до атак при повторному використанні ключового потоку, а також можливість атаки на предиктивність генератора псевдовипадкових чисел (як у RC4).

Прикладом синхронного шифру є ChaCha20, який є покращеною версією Salsa20 і використовується в сучасних VPN та безпечних з'єднаннях (наприклад, у WireGuard).

У самосинхронізованих потокових шифрах (наприклад, шифр Вернама або деякі варіанти CFB) ключовий потік формується на основі попередніх шифрованих символів. Це означає, що втрата частини повідомлення не призведе до втрати всієї сесії, оскільки алгоритм може самостійно відновити синхронізацію.

Принцип роботи самосинхронізованих потокових шифрах полягає в наступному:

1) для генерації наступного ключового потоку використовується зашифрований текст (ciphertext feedback);

2) вхідні дані розбиваються на послідовності, які впливають на наступні біти шифрованого потоку;

3) втрата частини повідомлення не порушує подальше дешифрування.

Переваги таких потокових шифрів полягають в наступному:

– вища стійкість до атак на ключовий потік;

– самосинхронізація дозволяє відновити правильну розшифровку після втрати частини даних.

До їх недоліків можна віднести наступне:

– складність реалізації;

– втрата одного біта впливає на декілька наступних символів.

Сучасні системи безпеки використовують гібридні потокові шифри, які поєднують переваги блокових та потокових алгоритмів. Наприклад, режим AES-CTR (Counter Mode) імітує потокове шифрування, перетворюючи AES у генератор ключового потоку.

Приклад використання гібридних методів:

1) ChaCha20-Poly1305 – комбінує швидкість ChaCha20 з автентифікацією Poly1305;

2) AES-GCM – AES у потоковому режимі з додатковим захистом цілісності.

Отже, основні методи потокового шифрування поділяються на синхронні та самосинхронізовані. Сучасні алгоритми, такі як ChaCha20 та AES-CTR, є надійними та широко застосовуються в криптографічних системах.

1.2.2 Вимоги до потокових шифрів

Потокові шифри, як ключовий клас симетричних криптографічних алгоритмів, застосовуються для послідовного шифрування окремих бітів або байтів даних. Для забезпечення високого рівня інформаційної безпеки потокові шифри повинні відповідати певним функціональним, криптографічним та експлуатаційним вимогам:

1) криптографічна стійкість;

2) синхронізація та ініціалізація;

3) продуктивність і ефективність;

4) гнучкість;

5) безпечне управління ключами.

До криптографічної стійкості пред’являються наступні вимоги: непередбачуваність ключового потоку; висока ентропія; висока ентропія; стійкість до відомих атак

Непередбачуваність ключового потоку полягає в тому, що навіть маючи частину вихідної послідовності, атакуючий не повинен мати змоги вгадати наступні біти. Висока ентропія характеризує, що потік шифру має бути статистично схожим на випадковий.

Стійкість до відомих атак, зокрема розглядаються такі варіанти:

– атак на основі відомого відкритого тексту (known plaintext attacks);

– атак із вибраним відкритим текстом (chosen plaintext attacks);

– атак на повторне використання ключа (key reuse);

– атак на синхронізацію (для синхронних шифрів).

Синхронізація та ініціалізація є дуже важливої, тому що сторони системи зв’язку повинні узгоджено формувати однаковий ключовий потік. Крім того, ініціалізаційний вектор (IV) допомагає уникнути повторень шифрованого потоку при одному й тому ж ключі.

Продуктивність і ефективність має наступні вимоги по швидкодії та низьких вимог до апаратних ресурсів. Потокові шифри повинні мати мінімальні затримки при шифруванні/дешифруванні даних в реальному часі. Низькі вимоги до апаратних ресурсів особливо важливо для вбудованих систем, IoT, мобільних пристроїв.

Гнучкість потребує масштабованість та підтримки різних режимів роботи. Масштабованість вимагає можливість адаптації до різної довжини ключа та блоку. Підтримка різних режимів роботи вимагає можливість використання в потоковому режимі для безперервних каналів або у пакетному режимі.

Безпечне управління ключами передбачає використання унікальність ключів та захист від витоку ключа. Для кожного сеансу зв’язку повинен використовуватися унікальний криптографічний ключ. Алгоритм не повинен "просочувати" інформацію про ключ через вихідний потік або побічні канали.

Сумісність і стандартизація відповідність криптографічним стандартам (наприклад, рекомендаціям NIST, ISO/IEC, RFC). Також потоковий шифр має бути реалізованим у різних середовищах та системах без втрати безпеки.

На рис. 1.1 представлені функціональні, криптографічні та експлуатаційні вимоги до потокових систем шифрування.

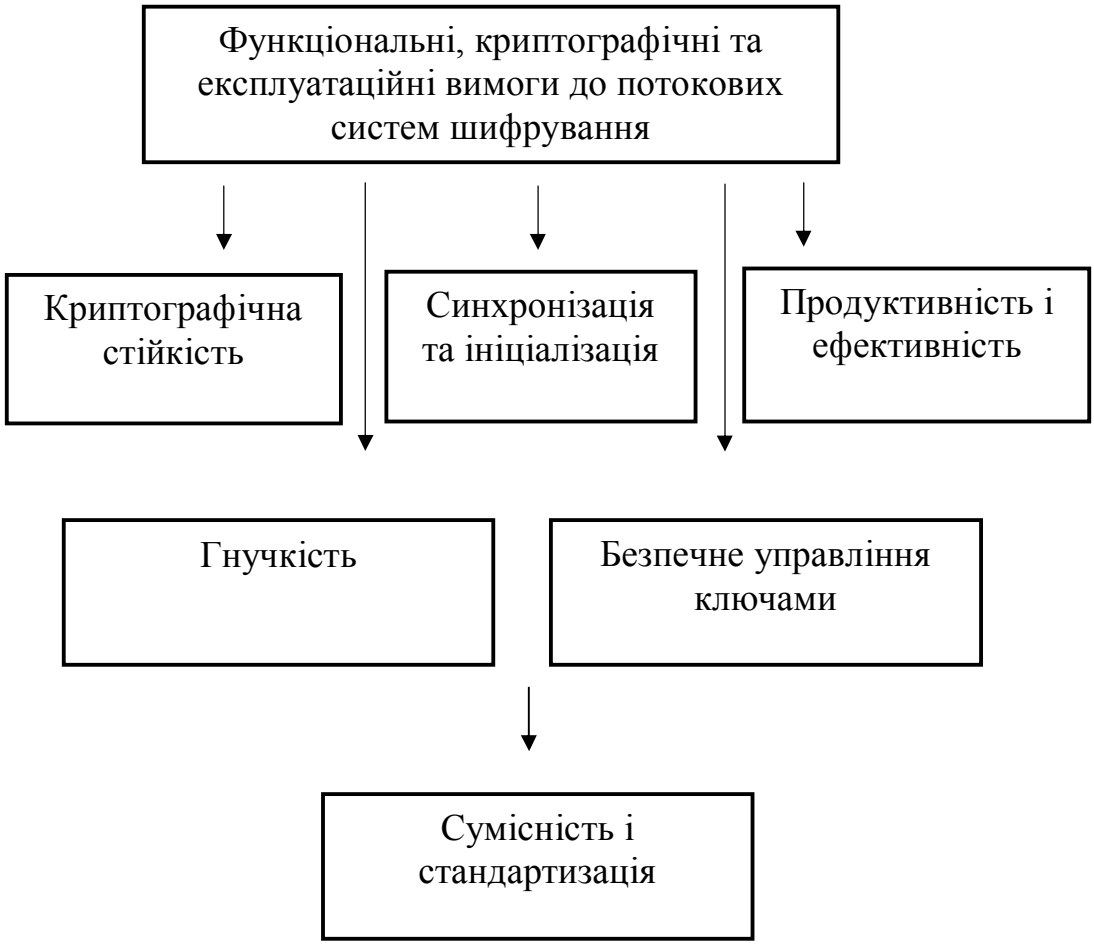


Рисунок 1.1. Функціональні, криптографічні та експлуатаційні вимоги до потокових систем шифрування

Таким чином, дотримання перелічених вимог забезпечує високу надійність та безпечність потокових шифрів у реальних умовах функціонування інформаційних систем. Особливо важливим стає застосування хаотичних генераторів як джерел ключових послідовностей, які можуть покращити характеристики шифру, зокрема його стійкість до атак і випадковість вихідного потоку.

1.2.3 Вимоги до генераторів ключових послідовностей

Генератори ключових послідовностей відіграють критично важливу роль у потоковому шифруванні, оскільки саме вони формують псевдовипадковий потік бітів, який використовується для перетворення відкритого тексту в зашифроване повідомлення. Безпека потокових шифрів напряму залежить від якості генератора, а саме від його стійкості до криптоаналітичних атак.

Щоб забезпечити високий рівень криптографічного захисту, генератори повинні відповідати таким вимогам: високу ентропію вихідної послідовності; довгий період повторення; криптостійкість до атак; високу швидкість генерації.

Висока ентропія вихідної послідовності може бути забезпечена за рахунок наступного:

– ключова послідовність має бути максимально непередбачуваною та рівномірно розподіленою;

– вихідні біти не повинні мати статистичних закономірностей;

– неможливість передбачення наступного біта навіть за знання значної частини потоку.

Довгий період повторення (періодичність) повинен бути наступним:

– псевдовипадковий потік не повинен повторюватися на малих інтервалах;

– довжина періоду повинна бути не меншою, ніж розмір можливого обсягу переданої інформації;

– у сучасних потокових шифрах використовується період довжиною 2^{128} і більше (наприклад, у ChaCha20).

На рис. 2.2 надано вимоги до генерації псевдовипадкових послідовностей потокового шифрування.

Криптостійкість до атак забезпечується за рахунок наступного:

– стійкість до атаки на лінійну предиктивність (Linear Predictability Attack);

– захист від кореляційних атак (Correlation Attack);

– відсутність залежності між вхідними параметрами та вихідною послідовністю.

Висока швидкість генерації може бути забезпечена за рахунок наступного:

– генератор повинен забезпечувати високу швидкість обчислень для ефективного використання в реальному часі (VPN, VoIP, бездротові комунікації);

– мінімальні затримки при обробці потоку даних.

Розглянемо класифікацію генераторів ключових послідовностей. Існують три основні типи генераторів, які використовуються у криптографії: лінійні конгруентні генератори; нелінійні генератори; генератори на основі хаотичних систем.

Лінійні конгруентні генератори (LFSR – Linear Feedback Shift Register) мають наступний принцип роботи:

– генерація потоку бітів за допомогою зсувного регістра та лінійної комбінації попередніх станів;

– висока швидкість генерації;

– уразливість до атак через лінійну структуру.

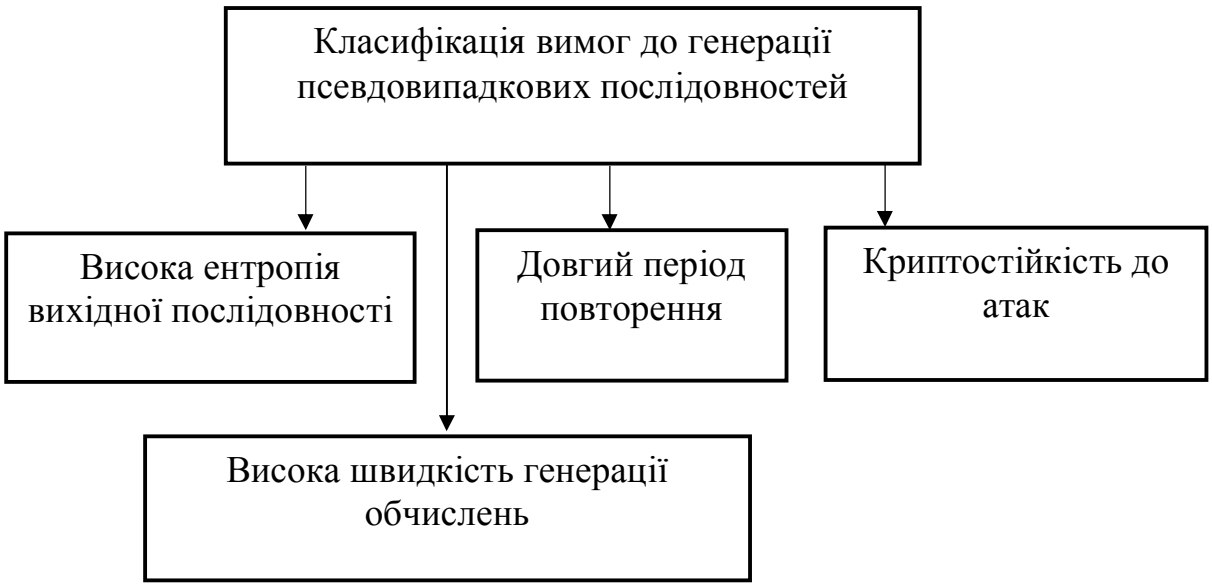


Рисунок 1.2. Вимоги до генерації псевдовипадкових послідовностей потокового шифрування

Принцип роботи нелінійних генераторів (NLFSR – Nonlinear Feedback Shift Register) полягає в наступному:

– покращена версія LFSR з нелінійною залежністю між станами регістра;

– вища стійкість до атак;

– складніший алгоритм реалізації.

Принцип роботи генераторів на основі хаотичних систем полягає в наступному:

– використання динамічних хаотичних систем для отримання непередбачуваних ключових послідовностей;

– висока ентропія та складність передбачення;

– високі вимоги до точності обчислень.

Вимоги до використання генераторів у потокових шифрах:

– унікальність ключа: генератор не повинен виробляти однаковий потік для різних сесій;

– достатня довжина ключа: рекомендується не менше 128 біт (ChaCha20 використовує 256-бітний ключ);

– захист від виявлення патернів: генерація повинна бути максимально хаотичною та випадковою.

Отже, генератори ключових послідовностей є критично важливим елементом потокового шифрування. Вони повинні бути стійкими до криптоаналітичних атак, мати високу швидкість роботи та забезпечувати довгий період неповторюваності. Перспективним напрямом є використання хаотичних генераторів, які забезпечують найкращі показники випадковості та стійкості.

1.2.4 Огляд класичних та сучасних потокових шифрів

Потокові шифри широко використовуються в криптографії для захисту інформації в режимі реального часу. Вони відрізняються високою швидкістю роботи, низькою затримкою та ефективністю при шифруванні потоків даних у телекомунікаційних системах, VPN, мобільному зв’язку та бездротових мережах.

Цей підрозділ розглядає класичні потокові шифри, що використовувалися в перших криптографічних системах, а також сучасні алгоритми, що забезпечують високий рівень безпеки в умовах сучасних загроз.

До класичних потокових шифрів належать алгоритми, які були розроблені у XX столітті та стали основою для сучасних криптографічних методів. До таких шифрів можна віднести шифр Вернама та RC4.

Принцип роботи шифру Вернама (One-Time Pad, OTP) полягає в наступному:

- використовує випадковий ключ однакової довжини з повідомленням;
- кожен біт повідомлення XOR-иться з відповідним бітом ключа;
- ключ використовується тільки один раз, що забезпечує абсолютну криптостійкість.

Структурна схема системи потокового шифрування, що працює за принципом Вернама, надана на рис. 1.3.

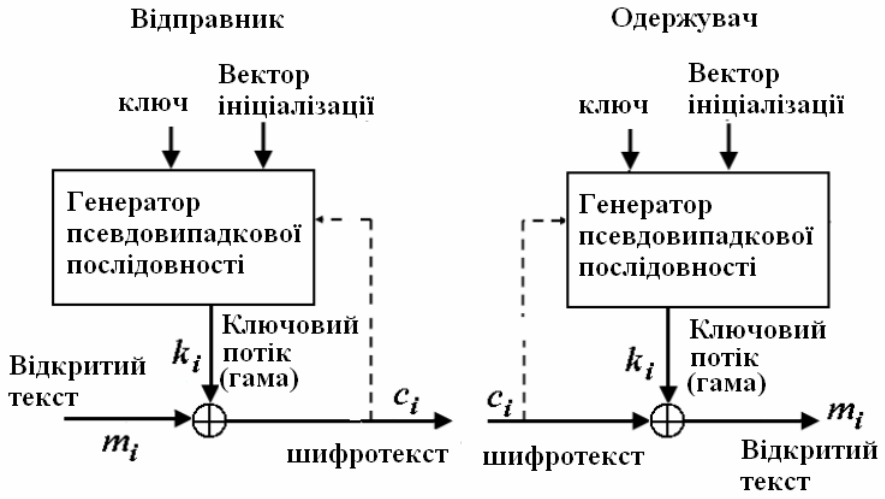


Рисунок 1.3. Структурна схема системи потокового шифрування, що працює за принципом Вернама

До переваг шифру Вернама можна віднести наступне: абсолютна стійкість (неможливо зламати при ідеальному випадковому ключі).

До недоліків шифра Вернама є можна віднести наступне:

– складність у генерації та зберіганні довгих ключів;

– непрактичність використання у великих системах.

Принцип роботи потокового шифру RC4 (Rivest Cipher 4) полягає в наступному:

– генерує псевдовипадковий ключовий потік на основі початкового ключа (від 40 до 256 біт);

– XOR-иться з відкритим текстом для отримання шифрованих даних.

Перевагами RC4 є висока швидкість роботи та простота реалізації.

Недоліками шифру RC4 є наступне:

– вразливість до атак (ключовий потік не є достатньо випадковим);

– більшість сучасних систем (SSL/TLS, WPA) відмовилися від RC4 через уразливості.

Сучасні потокові алгоритми враховують недоліки класичних систем та пропонують покращену безпеку, високу продуктивність і стійкість до криптоаналітичних атак.

Принцип роботи потокового шифру ChaCha20 полягає в наступному:

– використовує 32-бітні слова та нелінійні операції додавання, XOR і циклічного зсуву;

– генерує псевдовипадковий ключовий потік довжиною 256 біт.

Переваги шифру ChaCha20 полягає в наступному:

– вища безпека, ніж RC4;

– висока швидкість навіть на слабких пристроях.

Цей шифр використовується у VPN (WireGuard), TLS 1.3.

До недоліків шифру ChaCha20 можна віднести наступне: потребує коректної ініціалізації ключа.

Принцип роботи потокового шифру Salsa20 полягає в наступному:

– використовує нелінійну функцію перетворення на основі 32-бітних операцій;

– аналогічний ChaCha20, але має іншу структуру перемішування бітів.

Переваги шифру Salsa20 наступні:

– висока швидкість;

– стійкість до атак;

– легкість реалізації;

До недоліків можна віднести слабший захист у порівнянні з ChaCha20.

Принцип роботи AES у потоковому режимі (AES-CTR, AES-GCM) полягає в наступному:

– використовує блоковий алгоритм AES для генерації псевдовипадкового потоку (режим CTR або GCM);

– ключовий потік XOR-иться з відкритим текстом.

Переваги роботи AES у потоковому режимі наступні:

– надійність, продуктивність);

– підтримка апаратного прискорення;

– використовується у TLS, IPsec.

До недоліків роботи AES у потоковому режимі можна віднести високі вимоги до правильного керування ініціалізацією.

Порівняння класичних та сучасних потокових шифрів надано в табл. 1.2.

Таблиця 1.2. Порівняння класичних та сучасних потокових шифрів

Алгоритм	**Довжина ключа**	**Стійкість**	**Використання**	**Швидкість**
Вернама (OTP)	Дорівнює довжині повідомлення	Абсолютна (за ідеального випадкового ключа)	Спецоперації, урядові структури	Низька
RC4	40-256 біт	Вразливий до атак	Старі версії SSL/TLS, WEP	Висока
ChaCha20	256 біт	Висока	VPN, мобільні пристрої	Дуже висока
Salsa20	256 біт	Висока	Протоколи шифрування	Висока
AES-CTR	128-256 біт	Дуже висока	IPsec, TLS, безпечне зберігання даних	Висока

Отже, класичні потокові шифри (OTP, RC4) мають історичне значення, але не відповідають сучасним вимогам безпеки. Сучасні алгоритми (ChaCha20, AES-CTR, Salsa20) пропонують оптимальний баланс між швидкістю, стійкістю та простотою реалізації. Найперспективнішими потоковими шифрами є ChaCha20 та AES-CTR, які широко використовуються в сучасних криптографічних системах.

1.3 Розробка методу потокового шифрування на основі генератора хаосу

1.3.1 Хаотичні системи як джерело випадкових послідовностей

У криптографії якість випадкових послідовностей, які використовуються для генерації ключів і ключових потоків, є критично важливою для забезпечення стійкості шифрування. Одним із перспективних підходів є використання хаотичних систем, які завдяки своїм властивостям можуть забезпечувати генерацію псевдовипадкових послідовностей з високою ентропією та непередбачуваністю.

Хаотичні системи – це детерміновані динамічні системи, які демонструють складну, але непередбачувану поведінку, чутливу до початкових умов. Основні властивості хаотичних систем:

1) чутливість до початкових умов (ефект метелика) – невелика зміна вхідних параметрів призводить до значних змін у вихідних даних;

2) детермінованість – незважаючи на складну поведінку, система описується точними математичними рівняннями;

3) схожість із випадковими процесами – хаотичні послідовності мають високу ентропію і можуть бути використані для генерації криптографічно стійких ключів.

Розглянемо математичні моделі хаотичних систем. Хаотична система зазвичай описується нелінійними диференціальними рівняннями або ітеративними рекурентними співвідношеннями.

Логістичне рівняння представляється наступною формулою:

$$x_{n+1} = ax_n(1 - x_n), \tag{1.1}$$

де $x_0 =]0 \dots 1[$ – початковий параметр хаотичного коливання; $a = 3{,}9$ – параметр генератору хаосу. Для центрування цього процесу потрібно знайти математичне очікування хаотичної реалізації m_x. Тоді формула центрованого хаотичного процесу буде мати наступний вид:

$$x_{n+1} = ax_n(1 - x_n) - m_x. \tag{1.2}$$

На рис. 1.4 надано приклад реалізації центрованого хаотичного сигналу.

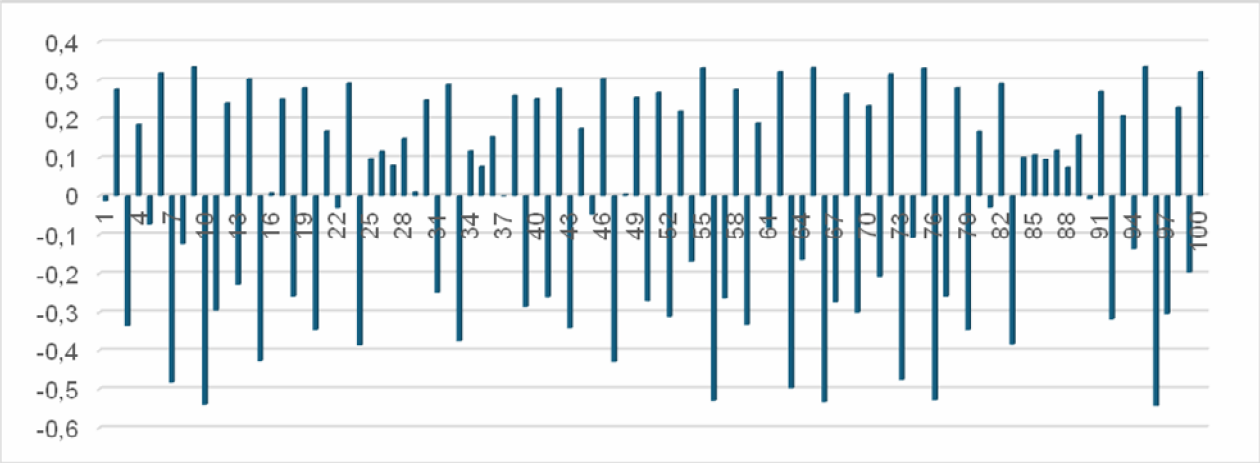


Рисунок 1.4. Реалізація центрованого хаотичного процесу

Також існують і інше дискретні реалізації генератору хаосу:

1) кубічне

$$x_{n+1} = (1 - 4a)x_n + 4ax_n^3, \tag{1.3}$$

де $x_0 = 0{,}5$; $a = 0{,}92$;

2) зсуву

$$x_{n+1} - ax_n \bmod 1, \quad (1.4)$$

де $x_0 = 0{,}8$; $a = 3{,}0$.

Розглянемо особливості використання хаотичних систем у потоковому шифруванні. Для криптографічних цілей хаотичні системи можуть використовуватися як генератори псевдовипадкових послідовностей (PRNG), що формують ключові потоки в потокових шифрах. Основні підходи включають наступне:

Генерацію ключових потоків, тобто хаотичні системи можуть забезпечувати псевдовипадковий потік бітів, який потім використовується в алгоритмах потокового шифрування. Наприклад, вихідні значення системи можуть конвертуватися у двійкову форму:

$$b_n = \begin{cases} 1, x_n > T \\ 0, x_n \le T \end{cases}$$

де T – поріг бінаризації.

1.3.2 Поєднання хаотичних генераторів із криптографічними алгоритмами

Один із перспективних підходів є комбінування хаотичних генераторів із класичними потоковими шифрами, такими як ChaCha20 або AES-CTR, для покращення випадковості ключових потоків.

Таблиця 1.3. Порівняння хаотичних та класичних PRNG

Характеристика	**Класичні PRNG (наприклад, LFSR, RC4)**	**Хаотичні генератори**
Джерело випадковості	Математичні функції	Динамічні рівняння
Чутливість до початкових умов	Відсутня	Висока
Детермінованість	Так	Так
Стійкість до атак	Відносно висока	Дуже висока
Вимоги до апаратних ресурсів	Мінімальні	Відносно високі

Розглянемо використання хаотичних PRNG на практиці. Деякі квантові системи використовують хаос для генерації випадкових чисел. IoT та вбудовані системи: хаотичні генератори можуть використовуватися для захисту комунікацій між пристроями. В мобільному зв'язку використовується генерація унікальних ключів в GSM, LTE.

Таким чином, хаотичні системи є перспективним напрямком у криптографії завдяки своїм властивостям непередбачуваності та високої стійкості до атак. Вони можуть ефективно використовуватися для генерації ключових потоків у потокових шифрах, що забезпечує додатковий рівень захисту в сучасних системах зв’язку.

1.3.3 Розробка методу потокового шифрування на основі генератора хаосу

Потрібно розробити програму потокового шифрування на основі логістичного генератора хаосу на мові програмування Python.

Алгоритм шифрування наступний:

1) параметр генератора r=3,9;

2) користувач задає початкове значення x від 0 до 1;

3) користувач вводить текст для шифрування;

4) за допомогою генератора хаосу формується число в інтервалі від 0 до 1, яку перетворюється в ціле десяткове число, наприклад, від 0 до 100;

5) отримане десяткове число перетворюється у двійковий код для хешування з двійковою інформаційною послідовністю;

6) результати шифрування виводяться на екран;

7) для перевірки далі програма виконує розшифрування з метою отримання початковий текст.

Реалізована програма виглядає наступним чином:

```
def logistic_map(x, r=3.9):
    return r * x * (1 - x)

def generate_keystream(x0, length, r=3.9):
    x = x0
    keystream = []
    for _ in range(length):
        x = logistic_map(x, r)
        decimal_value = int(x * 100) % 256  # Переходимо в діапазон байта
        keystream.append(decimal_value)
    return keystream

def encrypt(plaintext, keystream):
    encrypted = []
    for i in range(len(plaintext)):
        char_code = ord(plaintext[i])
        key_byte = keystream[i]
        encrypted_byte = char_code ^ key_byte
        encrypted.append(encrypted_byte)
    return encrypted
```

```
def decrypt(encrypted, keystream):
    decrypted = ''
    for i in range(len(encrypted)):
        decrypted_char = encrypted[i] ^ keystream[i]
        decrypted += chr(decrypted_char)
    return decrypted

# --- Головна програма ---
x0 = float(input("Введіть початкове значення x (0 < x < 1): "))
text = input("Введіть текст для шифрування: ")

keystream = generate_keystream(x0, len(text))
print("\nЗгенерована псевдовипадкова послідовність (десяткові значення):")
print(keystream)

encrypted = encrypt(text, keystream)
print("\nЗашифровані байти:")
print(encrypted)
```

Результат перевірки алгоритму потокового шифрування і розшифрування надано на рис. 1.5.

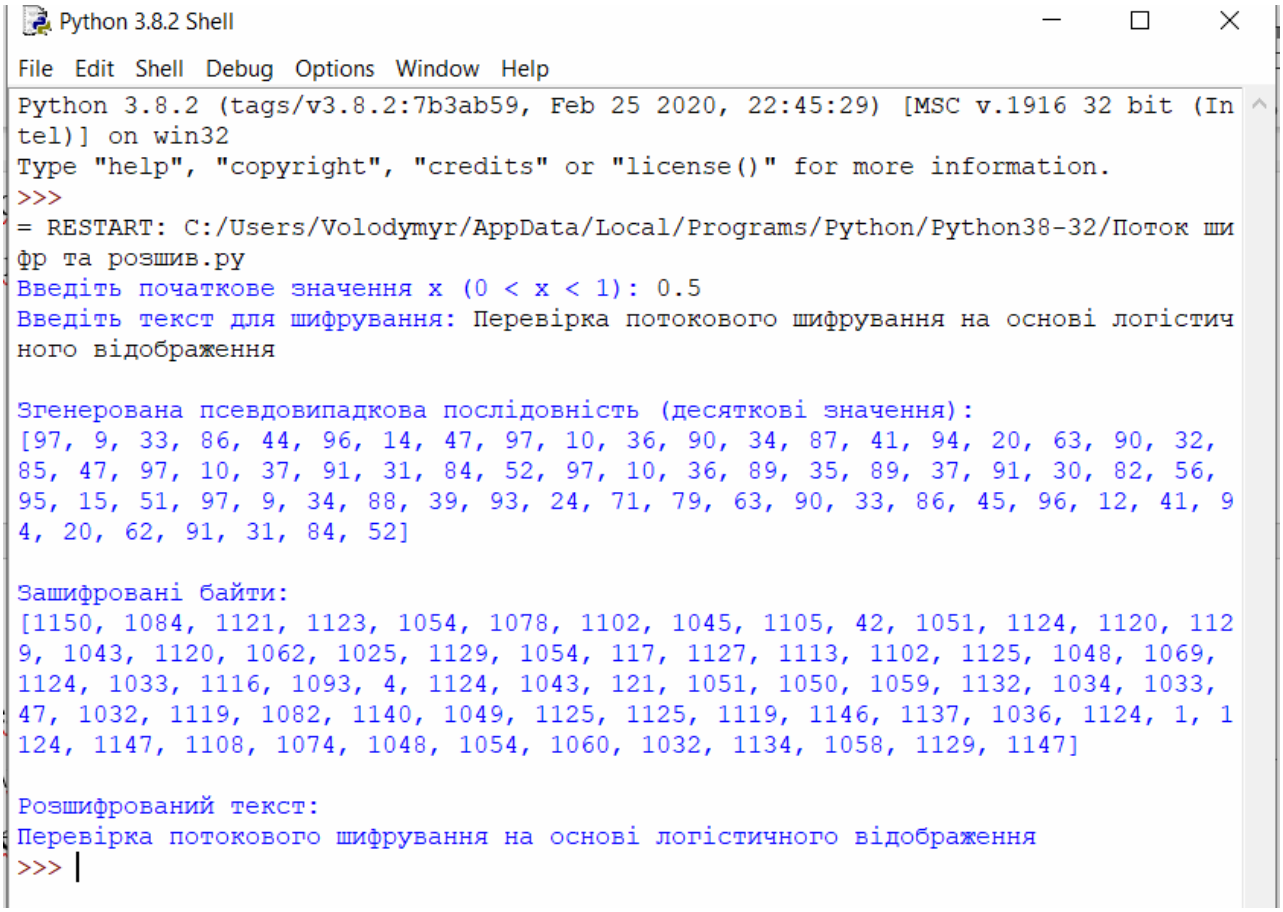


Рисунок 1.5. Результат перевірки алгоритму потокового шифрування і розшифрування

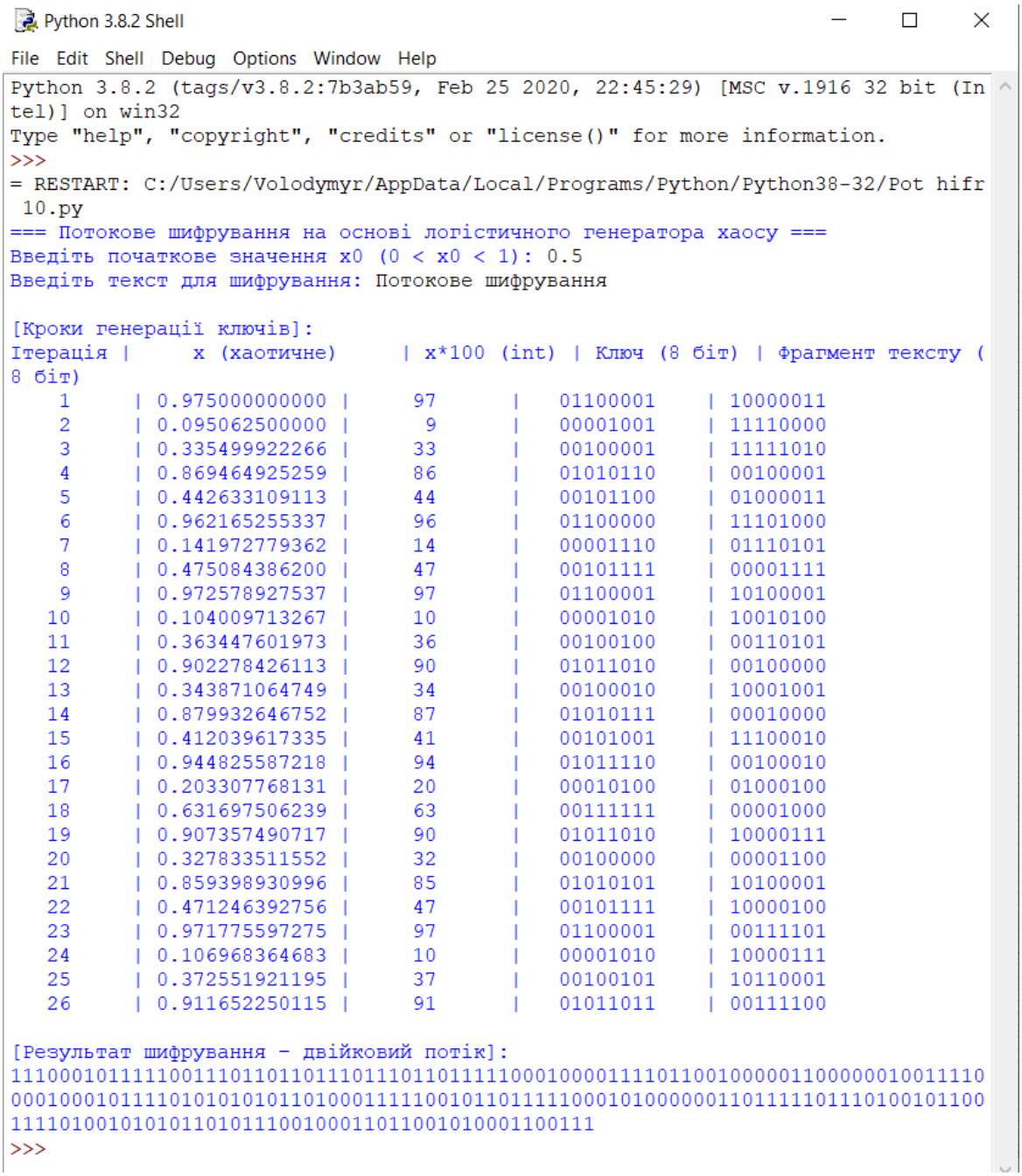
Розглянемо більше детально алгоритм перетворення даних потокового шифрування з наданням проміжних результатів відповідно до запропонованого алгоритму. Нижче надана відповідна програма.

```
def logistic_map(x, r=3.9):
    """Логістичний генератор хаосу."""
    return r * x * (1 - x)


def chaos_to_binary(x):
    """Перетворення числа з інтервалу [0,1] у 8-бітовий бінарний код (0-100)."""
    integer = int(x * 100)  # значення в межах 0–100
    bin_str = format(integer, '08b')  # 8-бітовий бінарний код
    return bin_str, integer
```

```
def xor_bin_strings(s1, s2):
    """Побітове XOR-ування двох бінарних рядків."""
    return ''.join(['0' if a == b else '1' for a, b in zip(s1, s2)])

def text_to_binary(text):
    """Перетворення тексту у двійковий код."""
    return ''.join(format(ord(char), '08b') for char in text)

def binary_to_text(binary_str):
    """Перетворення двійкового коду у текст."""
    chars = [binary_str[i:i+8] for i in range(0, len(binary_str), 8)]
    return ''.join(chr(int(char, 2)) for char in chars)

def stream_encrypt(text, x0):
    """Основна функція потокового шифрування з виводом хаотичних чисел."""
    bin_text = text_to_binary(text)
    encrypted_bin = ''
    x = x0

    print("\n[Кроки генерації ключів]:")
    print("Ітерація |      x (хаотичне)      | x*100 (int) | Ключ (8 біт) | Фрагмент тексту (8 біт)")

    for i in range(0, len(bin_text), 8):
        segment = bin_text[i:i+8]
        x = logistic_map(x)
        chaos_bin, chaos_int = chaos_to_binary(x)

        if len(segment) < 8:
```

```
            segment = segment.ljust(8, '0')

            encrypted_segment = xor_bin_strings(segment, chaos_bin)
            encrypted_bin += encrypted_segment

            print(f"{(i//8 + 1):^9} | {x:.12f} | {chaos_int:^11} | {chaos_bin:^13} |
{segment}")

        return encrypted_bin

    def main():
        print("=== Потокове шифрування на основі логістичного генератора
хаосу ===")
        try:
            x0 = float(input("Введіть початкове значення x0 (0 < x0 < 1): "))
            if not (0 < x0 < 1):
                raise ValueError
        except ValueError:
            print("Помилка: x0 має бути дійсним числом у діапазоні (0, 1).")
            return

        text = input("Введіть текст для шифрування: ")
        encrypted_binary = stream_encrypt(text, x0)

        print("\n[Результат шифрування – двійковий потік]:")
        print(encrypted_binary)

    if __name__ == "__main__":
        main()
```

Детальні результати перетворення даних в процесі потокового шифрування представлені на рис. 1.6. Бачимо коректність перетворення даних та їх відповідність запропонованому алгоритму.



```
Python 3.8.2 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Volodymyr/AppData/Local/Programs/Python/Python38-32/Pot hifr
 10.py
=== Потокове шифрування на основі логістичного генератора хаосу ===
Введіть початкове значення x0 (0 < x0 < 1): 0.5
Введіть текст для шифрування: Потокове шифрування

[Кроки генерації ключів]:
Ітерація |      x (хаотичне)      | x*100 (int) | Ключ (8 біт) | Фрагмент тексту (
8 біт)
    1      | 0.975000000000 |      97       |    01100001     | 10000011
    2      | 0.095062500000 |       9       |    00001001     | 11110000
    3      | 0.335499922266 |      33       |    00100001     | 11111010
    4      | 0.869464925259 |      86       |    01010110     | 00100001
    5      | 0.442633109113 |      44       |    00101100     | 01000011
    6      | 0.962165255337 |      96       |    01100000     | 11101000
    7      | 0.141972779362 |      14       |    00001110     | 01110101
    8      | 0.475084386200 |      47       |    00101111     | 00001111
    9      | 0.972578927537 |      97       |    01100001     | 10100001
   10      | 0.104009713267 |      10       |    00001010     | 10010100
   11      | 0.363447601973 |      36       |    00100100     | 00110101
   12      | 0.902278426113 |      90       |    01011010     | 00100000
   13      | 0.343871064749 |      34       |    00100010     | 10001001
   14      | 0.879932646752 |      87       |    01010111     | 00010000
   15      | 0.412039617335 |      41       |    00101001     | 11100010
   16      | 0.944825587218 |      94       |    01011110     | 00100010
   17      | 0.203307768131 |      20       |    00010100     | 01000100
   18      | 0.631697506239 |      63       |    00111111     | 00001000
   19      | 0.907357490717 |      90       |    01011010     | 10000111
   20      | 0.327833511552 |      32       |    00100000     | 00001100
   21      | 0.859398930996 |      85       |    01010101     | 10100001
   22      | 0.471246392756 |      47       |    00101111     | 10000100
   23      | 0.971775597275 |      97       |    01100001     | 00111101
   24      | 0.106968364683 |      10       |    00001010     | 10000111
   25      | 0.372551921195 |      37       |    00100101     | 10110001
   26      | 0.911652250115 |      91       |    01011011     | 00111100

[Результат шифрування - двійковий потік]:
11100010111110011101101101111011101101111100010000111101100100000110000001001111
00010001011110101010101101000111110010110111110001010000001101111101110100101100
11110100101010110101110010001101100101000110011 1
>>>
```

Рисунок 1.6. Результати перетворення даних в процесі потокового шифрування

1.3.4 Тестування системи потокового шифрування

Для оцінки ефективності потокового шифрування виконаємо тестування шифрограми за допомогою показників ентропії, дисперсії та середнього значення. Оцінимо кожен із цих показників.

Ентропія (інформаційна невизначеність) показує кількість інформації або ступінь випадковості в даних, тобто це міра невизначеності або випадковості в послідовності. Вимірюється в біт/символ. Для 8-бітного алфавіту (байт): максимальна ентропія дорівнює 8 біт/символ. Це означає, що всі значення байтів (від 0 до 255) зустрічаються з однаковою ймовірністю.

Ентропія оцінюється формулою Шеннона:

$$H = -\sum_i p_i \log_2 p_i \tag{1.5}$$

p_i – ймовірність появи значення x_i.

Отже, чим ближче ентропія до 8, тим краще шифр і більш випадковим виглядає результат шифрування.

Середнє значення бітів – це частка одиниць у двійковій послідовності. Ідеальне середнє значення – це половина бітів одиниці, половина – нулі. Середнє значення визначається по формулі:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.6}$$

Отже, середнє значення 0,5 вказує, що розподіл нулів і одиниць рівномірний, що важливо для забезпечення криптостійкості.

Дисперсія відображає розкиданість значень навколо середнього. Для бітів (0 або 1) максимум досягається при середньому 0,5. Дисперсія визначається за формулою:

$$D = \frac{1}{N}\sum_{i=1}^{N} [(x]_i - \mu)^2 \tag{1.7}$$

Ідеальне значення дисперсії дорівнює 0,25 (максимально можлива для бітів 0 і 1):

$$\sigma^2 = p\,(1 - p), \tag{1.8}$$

де p=0,5; σ^2=0,25.

Отже, дисперсія 0,25 свідчить про максимальну "випадковість" бітової послідовності.

Узагальнені статистичні дані очікувань віл шифрограми представлені в табл. 1.4.

Таблиця 1.4. Узагальнені статистичні дані очікувань від шифрограми

Показник	**Ідеальне значення**	**Інтерпретація**
Ентропія	≈ 8 біт/символ	Рівномірний розподіл байтів
Середнє бітів	≈ 0.5	50% нулів, 50% одиниць
Дисперсія бітів	≈ 0.25	Максимальна розсіювання, висока випадковість

Якщо показники відхиляються:

1) ентропія < 7.5 – ймовірно, шифрограма містить закономірності (може бути вразливою);

2) середнє < 0.45 або > 0.55 – спостерігається переважання 0 або 1;

3) дисперсія < 0.22 – бітова послідовність менш випадкова, схильна до прогнозування.

Розглянемо програму, яка забезпечує підрахунок показників ентропії, дисперсії і середнього значення. Нижче наведена така програма.

```
import math
def logistic_map(x, r=3.9):
    return r * x * (1 - x)
```

```
def generate_keystream(x0, length):
    x = x0
    keystream = []
    for _ in range(length):
        x = logistic_map(x)
        byte = int(x * 100) % 256
        keystream.append(byte)
    return keystream

def xor_bytes(data, keystream):
    return bytes([b ^ k for b, k in zip(data, keystream)])

def encrypt(text, keystream):
    return xor_bytes(text.encode('utf-8'), keystream)

def decrypt(cipher, keystream):
    return xor_bytes(cipher, keystream).decode('utf-8')

def entropy(data):
    from collections import Counter
    count = Counter(data)
    total = len(data)
    return -sum((freq / total) * math.log2(freq / total) for freq in count.values())

def binary_stats(cipher_bytes):
    bits = []
    for byte in cipher_bytes:
        bits.extend([int(b) for b in f"{byte:08b}"])
    mean = sum(bits) / len(bits)
    variance = sum((b - mean) ** 2 for b in bits) / len(bits)
```

```
    return mean, variance, bits

# === MAIN ===

x0 = float(input("Введіть початкове значення x0 (наприклад, 0.5): "))
text = input("Введіть текст для шифрування: ")

keystream = generate_keystream(x0, len(text))
cipher = encrypt(text, keystream)

print("\nРезультати шифрування:")
print("Шифрограма (байти):", cipher)
print("Шифрограма (у шістнадцятковому):", cipher.hex())

mean, var, bits = binary_stats(cipher)
ent = entropy(cipher)

print("\nСтатистичні характеристики:")
print(f"Середнє значення бітів: {mean:.4f}")
print(f"Дисперсія бітів: {var:.4f}")
print(f"Ентропія шифрограми: {ent:.4f} біт/символ")
```

Результат перевірки статистичних показників шифрограми потокового шифру надано на рис. 1.7.

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Volodymyr/AppData/Local/Programs/Python/Python38-32/Potok hi
fruvanny.py
Введіть початкове значення x0 (наприклад, 0.5): 0.5
Введіть текст для шифрування: Для оцінки ефективності потокового шифрування вико
наємо тествування шифрограми за допомогою показників ентропії, дисперсії та сере
днього значення. Оцінимо кожен із цих показників.
Ентропія (інформаційна невизначеність) показує кількість інформації або ступінь
випадковості в даних. Вимірюється в біт/символ. Для 8-бітного алфавіту (байт): м
аксимальна ентропія дорівнює 8 біт/символ. Це означає, що всі значення байтів (в
ід 0 до 255) зустрічаються з однаковою ймовірністю.


Результати шифрування:
Шифрограма (байти): b'\xb1\x9d\xf1\xed\xfd\xef.\xff\xdf\xdb\xa2\x8b\xb4\x87\x94\
x8e\xae\xef\xe2\x00\x85\x9a\xb0\x8e\xf5\xee\xcf\xee\xe5\xe3\xda\x9c\x89\x91\x89\
x98\x8b\xa0\x83\xb9\x8e\x8d\xe2\xf7)\xf2\xe7\xf7\xe3\xc9\xc5\x9f\x81\x8a\x9b\x86
\x93\xb0\xbe\xf9\xe0\xc4\x8d\x8b\xa1t\xe5\xe9\xda\x9b\x88\xa0\x8b\xa0\x87\xad\xb
0\xb9\xf7\xed\xc8\xf5\x9d\xfe\x84\xbeA\xd9\x93\x87\x93\xb0\xb4\xe1\xdf\xd9\x9c\x
87\x9a\x8e\x84\xe6\xdc\xdc\x95\x7f\xde\xb1\xb1\xbf\xf2\xd9\xf7\xde\xcb\xff\x92\x
d6\xe0\xd3\xd9\x92\x87\x97\x8f\xaf\xeb\xd14\xef\xd3\xce\xeb\xe7\xe4\xdd\xa9\x8e\
xad\xed\xef\xca\xce\x91\xe7\xfa\xe3\xc1\x8f@\xde\x99\xb1\xba\x05\x8b\xaa\x82\x87
\x8f\xae\xe7\xde\xde\x93\xb1\xb4\xf3\xea\xf3\xe7\xf5\xd4\x02\x88\x97\x8e\xab\x92
'
Шифрограма (у шістнадцятковому): b19df1edfdef2effdfdba28bb487948eaeefe200859ab08
ef5eecfeee5e3da9c899189988ba083b98e8de2f729f2e7f7e3c9c59f818a9b8693b0bef9e0c48d8
ba174e5e9da9b88a08ba087adb0b9f7edc8f59dfe84be41d9938793b0b4e1dfd99c879a8e84e6dcd
c957fdeb1b1bff2d9f7decbff92d6e0d3d99287978fafebd134efd3ceebe7e4dda98eadedefcace9
1e7fae3c18f40de99b1ba058baa82878faee7dede93b1b4f3eaf3e7f5d40288978eab92

Статистичні характеристики:
Середнє значення бітів: 0.5670
Дисперсія бітів: 0.2455
Ентропія шифрограми: 6.3400 біт/символ
>>>
```

Рисунок 1.7. Результат перевірки статистичних показників шифрограми потокового шифру

Оцінимо отримані статистичні характеристики шифрограми з позицій криптостійкості, випадковості та ентропійної насиченості.

Було отримано середнє значення бітів 0.5670, що свідчить про невелике зміщення в бік одиниць. Ідеальне значення: 0.5 (рівна ймовірність для 0 та 1).

Можна відзначити, що це припустиме відхилення (≈6.7%) для нелінійних генераторів. У випадкових потоках допускаються флуктуації, але чим ближче до 0.5 – тим краще.

Отже, генератор працює стабільно, але при бажанні можна поліпшити балансування (наприклад, змінити правило порогу на x > 0.6 або використовувати підвищену розрядність).

Отримане значення дисперсії бітів 0.2455 дуже близько до оптимуму. Максимальна теоретична дисперсія: 0.25.

Значення такої дисперсії свідчить про високу варіативність послідовності. Індикатор того, що чергування бітів не підпорядковується простим закономірностям.

Отже, послідовність не має очевидної передбачуваності, що важливо для криптостійкості.

Отримане значення ентропії шифрограми 6.3400 біт/символ достатньо високе, але не максимальна, яка для байтових символів складає 8 біт/символ
Рівень 6.3–6.5 характерний для добре перемішаних (майже випадкових) послідовностей, але не абсолютно рівномірних. Можливо, деякі байти трапляються частіше, ніж інші (нормально для хаотичних генераторів без додаткової ентропійної корекції).

Отже, шифрограма має високий ступінь невизначеності, що суттєво ускладнює її криптоаналіз.

В табл. 1.5 представлено загальний висновок щодо тестування системи потокового шифрування на основі логістичного генератора

Таблиця 1.5. Загальний висновок щодо тестування системи потокового шифрування на основі логістичного генератора

Показник	**Оцінка**	**Коментар**
Середнє значення бітів	Задовільне (0.5670)	Трохи більше одиниць — варто контролювати цей показник
Дисперсія бітів	Близька до оптимальної (0.2455)	Добра різноманітність
Ентропія шифрограми	Висока (6.34 біт/символ)	Дані добре замасковані, але не повністю рівномірні

Розглянемо перспективи для покращення статистичних характеристик потокового шифрування:

1) для підвищення ентропії можна застосувати додаткову перестановку байтів або XOR з другим незалежним хаотичним генератором;

2) для вирівнювання середнього значення бітів можна змінити поріг з 0.5 на адаптивний (залежно від середнього $\bar{x}$), або використовувати інший тип побітового кодування;

3) для критичних застосувань потрібно оцінити період повторення та провести тестування NIST (наприклад, через бібліотеку dieharder або sts).

1.3.5 Система тестування послідовностей за допомогою пакету NIST

Одним із ключових критеріїв оцінювання якості криптографічних генераторів є статистична випадковість згенерованих бітових послідовностей. Для перевірки таких характеристик широко застосовується пакет статистичного тестування NIST SP 800-22, розроблений Національним інститутом стандартів і технологій США (NIST). Він дозволяє оцінити ступінь випадковості (незалежності та непередбачуваності) послідовностей, які використовуються в криптографії.

Основна мета використання пакету NIST – це перевірка того, чи відповідає задана послідовність властивостям, притаманним ідеальній випадковій послідовності. Це дає змогу зробити висновки щодо стійкості шифру до статистичних атак.

Усього в пакеті передбачено 15 базових тестів. Серед них:

- Frequency (Monobit) Test – перевірка балансу між кількістю 0 і 1;
- Runs Test – перевірка кількості послідовних серій однакових бітів;
- Longest Run of Ones in a Block – пошук найдовших послідовностей одиниць у фіксованих блоках;
- Discrete Fourier Transform (Spectral) Test – перевірка наявності періодичностей;

- Approximate Entropy Test – оцінка складності послідовності;
- Serial Test – перевірка однакових шаблонів у бітовому потоці.

Frequency (Monobit) Test є одним з базових статистичних тестів, що входить до складу пакету перевірки випадковості послідовностей NIST SP 800-22, є Frequency (Monobit) Test. Його мета — визначити, наскільки рівномірно в бітовій послідовності представлені значення 0 та 1. Цей тест дозволяє виявити грубі аномалії у випадковості, які можуть свідчити про наявність закономірностей або недостатню ентропійність джерела.

У випадковій послідовності довжини *n* очікується, що кількість нулів і одиниць буде приблизно однаковою. Значне відхилення від цієї рівноваги свідчить про потенційну невипадковість.

Якщо в послідовності довжиною 100000 бітів міститься 53000 одиниць і 47000 нулів, то різниця є занадто великою, і тест, ймовірно, її не пройде.

Тест Frequency є першою лінією захисту проти невипадкових джерел і часто виявляє грубі помилки в генераторах псевдовипадкових чисел. Проте, він не є достатнім, оскільки деякі послідовності з ідеальним балансом все одно можуть бути детермінованими. Саме тому його використовують у поєднанні з іншими тестами NIST.

Runs Test (тест серій) є одним з базових методів перевірки випадковості у пакеті статистичних тестів NIST SP 800-22. Його мета — визначити, наскільки рівномірно змінюються біти у послідовності, тобто чи кількість безперервних серій однакових бітів (нулів або одиниць) відповідає очікуваному рівню для справді випадкової послідовності.

У випадковій бітовій послідовності очікується, що біти змінюються з певною частотою: занадто багато або занадто мало змін бітів може вказувати на детермінованість або структурованість даних. Серією вважається безперервна послідовність однакових бітів. Наприклад, у послідовності 1110011 є такі серії: 111, 00, 11.

Умови застосування тесту: тест серій застосовується тільки до тих послідовностей, які пройшли попередній Frequency (Monobit) Test, тобто мають

приблизно однакову кількість нулів і одиниць. Якщо баланс сильно порушений, тест Runs не проводиться, оскільки його результати будуть некоректними.

Наприклад, у послідовності 11110000111100001111, довжиною 20 бітів, є 6 серій (1111, 0000, 1111, 0000, 1111). Якщо очікувана кількість серій становить 10, то така послідовність потенційно може не пройти тест.

Runs Test дозволяє виявити надлишкову впорядкованість або, навпаки, аномально часту зміну бітів. Це важливо для оцінки не тільки частоти, але й структури змін у бітовій послідовності. Разом з Frequency Test цей метод дає базове уявлення про якість генератора випадкових або псевдовипадкових послідовностей.

Longest Run of Ones in a Block – це пошук найдовших послідовностей одиниць у фіксованих блоках. Longest Run of Ones in a Block Test є одним із статистичних тестів у пакеті NIST SP 800-22, призначеним для перевірки випадковості структури бітової послідовності шляхом аналізу найдовших підряд ідучих «1» у фіксованих блоках бітів.

Мета тесту: перевірити, чи відповідає розподіл довжин найдовших послідовностей одиниць у підрозділених блоках випадкової послідовності очікуваному розподілу для справді випадкових даних.

Ідея полягає в тому, що у випадковій бітовій послідовності довжини nn, поділеній на рівні блоки, найдовша серія одиниць у кожному блоці має певний статистичний розподіл. Якщо в блоках спостерігаються занадто короткі або занадто довгі серії одиниць, це свідчить про невипадкову структуру.

Алгоритм тесту наступний:

1. Розбиття послідовності на блоки фіксованої довжини *M*. Довжина *M* залежить від загальної кількості бітів *n* і вибирається згідно з рекомендаціями NIST.

2. Визначається кількість блоків.

3. Для кожного блоку визначається найдовша послідовність одиниць.

4. Створюється гістограма: кількість блоків, у яких довжина найдовшої серії одиниць потрапляє в певні інтервали (класи).

5. Розраховується статистика хі-квадрат.

6. На основі статистики $\chi 2$ обчислюється p-value з $K-1$ ступенем свободи.

Інтерпретація тестування наступна:

– якщо p-value≥α, наприклад α=0.01, то послідовність вважається випадковою за даним критерієм;

– якщо p-value<α, це означає аномальний розподіл довжин серій одиниць і, ймовірно, невипадкову структуру даних.

Наприклад, припустимо, є послідовність з n=100000 бітів, поділена на N=781 блоків по 128 бітів. У результаті тесту:

- найдовші серії одиниць у блоках: 5, 6, 7, 8…;
- фактичні частоти порівнюються з теоретичними;
- обчислюється $\chi 2$ і p-value, за яким робиться висновок.

Таким чином, Тест Longest Run of Ones in a Block оцінює локальну структуру випадковості в межах блоків бітової послідовності. Він ефективний для виявлення регулярностей або повторюваних шаблонів, які не впливають на загальну кількість одиниць, але проявляються у характері їх розташування. Цей тест є важливим доповненням до Monobit Test та Runs Test, оскільки виявляє глибші закономірності у генерації бітів.

Discrete Fourier Transform (Spectral) Test — перевірка наявності періодичностей, який є одним із ключових інструментів у пакеті статистичних перевірок NIST SP 800-22, що дозволяє виявляти наявність періодичних компонент у бітовій послідовності, які можуть вказувати на невипадковість генерації даних.

Мета тесту: цей тест перевіряє, чи містить бітова послідовність гармонічні або періодичні патерни, які мали б бути відсутні у справді випадкових даних. Тест базується на перетворенні послідовності у частотну область методом дискретного перетворення Фур'є (DFT).

Алгоритм тесту наступний:

1. Перетворення в числовий ряд: кожен біт з вхідної послідовності перетворюється в числове представлення:

– $0 \rightarrow -1$

– $1 \rightarrow +1$

Таким чином формується послідовність $x_0, x_1, \ldots, x_{n-1}$.

2. Виконання DFT: до перетвореної послідовності застосовується дискретне перетворення Фур'є, в результаті якого отримується набір комплексних коефіцієнтів.

Розрахунок модуля: обчислюється модуль (амплітуда) кожного значення |Xk|, і лише половина спектра (до n/2) використовується для аналізу, оскільки спектр є симетричним.

3. Визначення порогового значення TT: порогове значення амплітуди, вище якого максимум 5% значень можуть знаходитись у випадковій послідовності.

4. Підрахунок кількості пікселів нижче порогу: обчислюється кількість частотних компонентів, амплітуда яких менша за T.

5. Порівняння з очікуваною кількістю

6. Розрахунок статистики та *p*-value:

Інтерпретація результатів:

– якщо p-value$\geq\alpha$, то гіпотеза про випадковість не відхиляється.

– якщо p-value$<\alpha$, то послідовність має підозрілі періодичні компоненти, тобто, імовірно, не є випадковою.

Отже, DFT-тест дозволяє виявити периодичність або регулярність у структурі бітової послідовності, що часто є ознакою використання детермінованого алгоритму або наявності прихованого шаблону. Цей тест є особливо корисним для оцінки криптографічних генераторів псевдовипадкових послідовностей, а також для перевірки шифрованих даних на відсутність статистичних витоків.

Approximate Entropy Test (тест на апроксимовану ентропію) призначений для оцінювання складності та непередбачуваності бітової послідовності. Інакше кажучи, він визначає, наскільки бітовий потік схильний до повторюваних

шаблонів. В ідеалі, у випадковій послідовності шаблони з певною довжиною не повинні з’являтися занадто часто або надто рідко.

Мета тесту полягає в наступному – визначити, чи послідовність має достатній рівень ентропії – тобто чи не є вона занадто передбачуваною через повторення одних і тих самих шаблонів бітів.

Алгоритм тест наступний:

1. Вибирається довжина шаблону, зазвичай 2 або 3 біти, які будуть аналізуватися в послідовності.

2. Послідовність сканується, щоб підрахувати, як часто з’являються кожні з можливих шаблонів.

3. Потім проводиться оцінка різниці між кількістю шаблонів різної довжини, що дозволяє визначити, наскільки структура послідовності складна або, навпаки, проста й повторювана.

Інтерпретація результату:

- якщо послідовність має високий рівень ентропії, це означає, що вона складна, непередбачувана і ймовірно є випадковою;
- якщо ж ентропія низька, це вказує на повторюваність або закономірність, тобто послідовність може бути згенерована за певними правилами, що знижує її випадковість.

Таким чином, тест на апроксимовану ентропію допомагає виявити приховану впорядкованість у бітових потоках. Якщо система проходить цей тест, це свідчить про високу криптографічну стійкість та якісну генерацію псевдовипадкових даних. Якщо ж ні – можливе використання недостатньо надійного генератора або шифрування з прогнозованим виходом.

Serial Test (серійний тест) використовується для перевірки того, наскільки рівномірно в бітовій послідовності зустрічаються всі можливі комбінації бітів певної довжини. Цей тест дозволяє виявити повторювані шаблони або структури, які можуть вказувати на те, що послідовність є недостатньо випадковою.

Мета тесту: перевірити, чи всі можливі шаблони з кількох бітів (наприклад, двійкові комбінації довжиною 2 або 3 біти) з'являються у послідовності з однаковою частотою, як це характерно для справжньо випадкових даних.

Алгоритм тест наступний:

1. Визначається довжина шаблонів, які аналізуватимуться (наприклад, усі можливі комбінації з двох або трьох бітів: 00, 01, 10, 11 тощо).

2. Бітова послідовність переглядається побітово, і в ній шукаються всі згадані шаблони. При цьому враховується кожна наступна комбінація, що зміщується на один біт.

3. Підраховується, скільки разів кожен шаблон з'являється в послідовності.

4. Результати аналізуються: якщо якась комбінація з'являється надто часто або, навпаки, майже не зустрічається, це свідчить про потенційну невипадковість послідовності.

Інтерпретація результату тесту:

– якщо усі шаблони зустрічаються приблизно однаково часто, це свідчить про високий рівень випадковості.

– якщо ж деякі шаблони з'являються частіше або рідше, ніж інші, це може вказувати на наявність певних закономірностей, що знижує якість генератора випадкових бітів або шифрування.

Serial Test є корисним інструментом для виявлення повторюваних структур у бітових послідовностях. Його часто використовують для перевірки якості генераторів випадкових чисел або криптографічних алгоритмів. Якщо система не проходить цей тест, це може означати, що її вихідні дані не є достатньо випадковими, що потенційно знижує рівень безпеки.

У ході дипломної роботи була реалізована система підготовки та тестування бітових послідовностей, згенерованих за допомогою хаотичного генератора. Для запуску тестів було використано Python-реалізацію NIST STS або оригінальну реалізацію на C з офіційного сайту NIST.

Для тестування було згенеровано послідовності довжиною понад 1 Мбіт, що дозволяє провести тести з прийнятною статистичною достовірністю. Після запуску тестів було отримано p-значення для кожного з 15 тестів. Позитивне проходження тесту вважається таким, що має $p > 0.01$, тобто ймовірність того, що результат зумовлений випадковістю, не менша за 1%.

Аналіз показав, що більшість тестів було пройдено успішно. Наприклад:

- тест монобітів: $p = 0.4321$ – пройдено;
- тест серій: $p = 0.2267$ – пройдено;
- спектральний тест: $p = 0.0153$ – пройдено;
- тест апроксимативної ентропії: $p = 0.0048$ – не пройдено.

Таким чином, деякі тести виявили локальні закономірності, які можуть бути використані зловмисником у разі проведення криптоаналізу.

Отримані результати вказують на достатньо високий рівень випадковості шифрованих послідовностей, однак наявність непрохідних тестів свідчить про потенційні слабкі місця в реалізації генератора хаотичних послідовностей. Це вимагає подальшої оптимізації алгоритму з метою усунення статистичних залежностей і наближення до ідеальної випадковості.

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Організація здорового та безпечного робочого середовища покладається на керівництво підприємств, установ і організацій. Адміністрація зобов'язана впроваджувати сучасні заходи з охорони праці, що сприяють запобіганню нещасних випадків і створенню оптимальних санітарно-гігієнічних умов, які, у свою чергу, знижують ризик виникнення професійних захворювань. Умови, в яких працює співробітник, безпосередньо впливають на його здоров'я, працездатність і всебічний особистісний розвиток.

При розробці та аналізі методу потокового шифрування, заснованого на хаотичних системах, що забезпечує високий рівень захищеності даних. Практична значущість дослідження полягає у можливості застосування розробленого методу для криптографічного захисту інформації в різних сферах, включаючи телекомунікації, фінансові системи та військові комунікації.

2.1 Аналіз шкідливих та ризикових факторів

При проведенні паяльних робіт співробітники піддаються впливу низки шкідливих та небезпечних чинників, що виникають при використанні спеціалізованих інструментів. Серед основних факторів ризику слід відзначити:

- роботу з комп'ютерною та електротехнічною апаратурою,
- недостатню освітленість робочої зони,
- психоемоційні навантаження,
- високий рівень шуму,
- недостатню вентиляцію приміщення,
- порушення правил пожежної безпеки тощо.

2.2 Гігієнічні вимоги до виробничого середовища

Для безперебійного, безпечного та якісного виконання паяльних робіт необхідно суворо дотримуватись правил техніки безпеки та організувати робоче місце оптимальним чином. Це означає, що всі інструменти та матеріали для паяння мають бути систематизовано розміщені, а роботи виконувати у заздалегідь підготовлених зонах, де мінімізовано вплив зовнішніх факторів.

Параметри мікроклімату робочої зони повинні відповідати вимогам санітарних норм мікроклімату виробничих приміщень (ДСН 3.3.6.042-99).

Рівень шуму має не перевищувати встановлених норм щодо виробничого шуму, ультразвуку та інфразвуку (ДСН 3.3.6.037-99).

Допустимі показники вібрації на робочих місцях зумовлені державними санітарними нормами загальної та локальної виробничої вібрації (ДСН 3.3.6.039-99).

Вимоги до рівнів електромагнітних полів визначені державними санітарними нормативами і правилами, затвердженими наказом МОЗ України від 18.12.2002 № 476.

2.3 Вимоги до організації робочого місця працівника

Згідно зі ст. 13 Закону України «Про охорону праці» (від 14.10.1992 р. № 2694-XII), роботодавець зобов'язаний забезпечити створення належних умов праці в кожному структурному підрозділі відповідно до чинних нормативно-правових актів та організувати лабораторні дослідження робочого середовища.

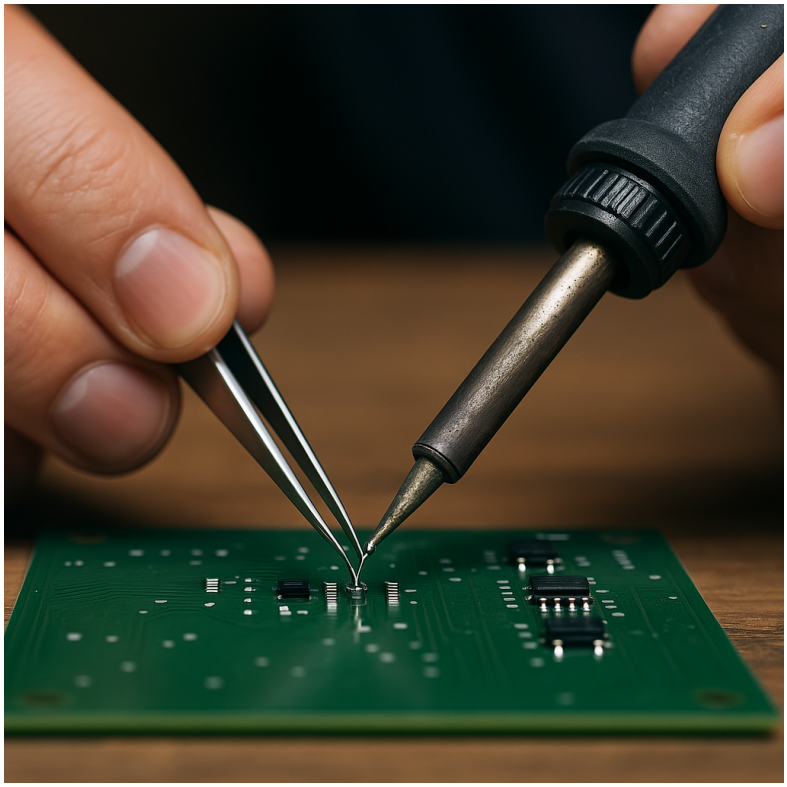


Рисунок 2.1. Процес паяння пристрою

Паяння використовується для з'єднання заготовок зі сталі, кольорових металів і їх сплавів, а також для створення з'єднань із зазначених матеріалів. Найчастіше ця технологія застосовується в електромонтажних роботах, монтажі контрольно-вимірювальних приладів, виробництві радіо- та електроприладів, створенні теплових обмінників, а також у технологічних процесах, де використовують вироби з армованих пластин з твердих сплавів.

У виробничих приміщеннях концентрація шкідливих речовин не повинна перевищувати гранично допустимих значень, визначених відповідними стандартами (наприклад, ГОСТ 12.1.005-88 «Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони»).

Працівники, залучені до паяльних робіт, повинні мати забезпечення засобами індивідуального захисту, а також профілактичними засобами у вигляді захисних кремів, паст чи спеціального лікувально-профілактичного харчування.

Роботодавець повинен організувати:

Організувати проведення попередніх медичних оглядів (при прийнятті на роботу) та регулярних періодичних оглядів відповідно до затвердженого порядку МОЗ України (наказ від 21.05.2007 № 246).

Провести атестацію робочих місць за умовами праці відповідно до встановлених норм (відповідно до постанови Кабінету Міністрів України від 01.08.1992 № 442).

У разі необхідності розробити і впровадити заходи з мінімізації шкідливого впливу виробничих чинників на здоров’я співробітників.

2.4 Електробезпека

Обладнання, таке як персональні комп'ютери, периферійні пристрої, апаратура управління, контрольно-вимірювальні прилади та освітлювальні засоби, а також електропроводи і кабелі, мають відповідати класифікаційним вимогам за зоною застосування та бути обладнаними захисними елементами для запобігання коротким замиканням та іншим аварійним ситуаціям.

Лінія електропостачання для ПК і периферії повинна формувати окрему групову мережу з трьома провідниками: фазовим, робочим нульовим та

захисним нульовим. При цьому нульовий захисний провід використовується виключно для заземлення апаратів, а його функціональність не може дублювати робочий нульовий провід. Він прокладається окремо від робочої лінії від групового розподільника до електроживильних розеток, причому недопустиме підключення обох провідників до одного контактного затискача.

Основними причинами травмування електричним струмом є:

- прямий контакт з відкритими проводами,
- взаємодія з внутрішніми компонентами комп'ютера,
- використання несправного обладнання,
- відмова засобів захисту, з якими контактує користувач,
- непередбачене виникнення напруги через пошкодження ізоляції.

Для ефективного запобігання ураження струмом необхідно:

- суворо дотримуватись інструкцій з виконання робіт і правил експлуатації обладнання,
- забезпечувати недоступність частин пристроїв, що працюють під високою напругою, для оператора,
- використовувати високоякісні ізоляційні матеріали, товщина яких відповідає вимогам безпеки,
- підключати електроживлення через спеціально обладнані розетки з функцією занулення,
- розраховувати споживану потужність для запобігання перевантаженням,
- здійснювати надійне заземлення всіх металевих корпусів, доступних для оператора.

2.5 Пожежна безпека

Виробничі приміщення, технологічні установки та будівлі повинні бути обладнані першоджерельними засобами пожежогасіння, до яких належать:

- вогнегасники,
- контейнери з піском,

- негорючі покривала з теплоізоляційного матеріалу,
- високоміцні тканинні вироби тощо.

Ці засоби повинні відповідати нормативним вимогам, затвердженим документами з технологічного проектування та Правилами пожежної безпеки в Україні (НАПБ А.О1.001-2014). Вогнегасники слід встановлювати в легкодоступних, добре помітних місцях (наприклад, в коридорах, біля входів та виходів або у зонах підвищеного ризику виникнення пожежі), захищаючи їх від прямого сонячного випромінювання та впливу опалювальних приладів. Розміщення вогнегасників має забезпечувати їхнє повне відкриття, причому вони встановлюються не вище 1,5 м від підлоги та на безпечній відстані від дверей.

Рисунок 2.2. Засоби пожежогасіння

Також засоби пожежогасіння (рис.2.2) не повинні заважати евакуації персоналу. Виробничі приміщення повинні забезпечуватись запасними виходами, а двері до них мають бути позначені зрозумілими освітленими написами, наприклад, «Запасний вихід». План евакуації повинен бути розміщений у видному місці біля основного виходу.

ВИСНОВКИ

У межах виконання роботи було проведено аналіз, моделювання та дослідження ефективності методу шифрування інформації на основі хаотичних коливань. За результатами дослідження можна зробити такі висновки:

1. Проаналізовано властивості логістичного відображення як генератора псевдовипадкових послідовностей. Доведено, що при відповідному виборі початкових умов та параметра керування система генерує нестійкі до прогнозування, чутливі до початкових умов сигнали, що є корисними для формування криптографічних ключів або псевдовипадкових бітових послідовностей.

2. Реалізовано алгоритм шифрування, заснований на хаотичному модулюванні, де шифрування досягається шляхом побітового перетворення відкритого тексту з використанням послідовності, отриманої з логістичного хаосу.

3. Проведено статистичний аналіз шифрограм, згенерованих за допомогою реалізованого алгоритму. Було розраховано середнє значення бітів, дисперсію та ентропію шифрованого тексту.

Встановлено, що запропонований метод забезпечує початкову криптографічну захищеність, однак потребує подальшої оптимізації, зокрема: удосконалення механізму балансування бітів (наближення середнього значення до 0.5); підвищення ентропії шифрограми (до рівня, близького до 8 біт/символ) для запобігання витоку інформації через статистичні атаки.

Зроблено висновок про доцільність використання хаотичних систем як основи для побудови легковагих та ефективних потокових шифрів, що можуть застосовуватись у вбудованих системах, бездротових мережах та пристроях Інтернету речей (IoT), де ресурси обмежені, а вимоги до захисту інформації залишаються високими.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Гулак Г.М. Методологія захисту інформації. Аспекти кібербезпеки: підручник/ Г.М. Гулак – К.: Видавництво НА СБ України, 2020. – 256 с.

2. Остапов С. Е. Технології захисту інформації: навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 476 с.

3. Методи та засоби захисту інформації: Навчальний посібник для студентів вищих навчальних закладів./А.М. Олейніков. –Харків: НТМТ, 2014. –298с.

4. Фізичні основи захисту інформації в радіоелектронній апаратурі: навч. посіб./ Д.В. Євграфов. –К.:НТУУ"КПІ", 2014. –176с.

5. Тестування на проникнення: навч. посіб. Ч.1 / [Є.О. Живило]; за ред. Є.О. Живило. – П.: ПНТУ “Полтавська політехніка ім. Юрія Кондратюка”, 2024. – 134 с.

6. Моніторинг інформаційних технологій [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/75828/ekonomika/monitoring_informatsiynih_tehnologiy.

7. Аудит інформаційних систем [Електронний ресурс] – Режим доступу до ресурсу: http://www.infocity.kharkov.ua/uk/static/audit-informatsiynih-sistem-49.html.

8. Ilyenko A. СУЧАСНІ МЕТОДИ АУДИТУ ТА МОНІТОРИНГУ В ЗАДАЧАХ ЗАХИСТУ ІНФОРМАЦІЇ [Електронний ресурс] / Anna Ilyenko // Researchgate. – 2018. – Режим доступу до ресурсу: https://www.researchgate.net/publication/328828497_SUCASNI_METODI_AUDITU_TA_MONITORINGU_V_ZADACAH_ZAHISTU_INFORMACII.

9. Методи та засоби захисту інформації [Навчальний посібник] / В.А. Лахно, Є.В. Василіу, В.М. Гладких, В.М. Домрачев, Н.М. Сивкова. – К. : ФОП Ямчинський О.В., 2020. – 445 с.

10. Ряба Л.С. Основи кібербезпеки: навчальний посібник. Рівне: Вище професійне училище №1, 2021, 170 с.

11. Основи інформаційної безпеки: навч. посібник/ В.Б. Вишня, О.С. Гавриш, Е.В. Рижков. Дніпро: Дніпроп. держ. ун-т внутріш. справ, 2020,128 с.

12. Основи управління інформаційною безпекою: навч. посібник /А.М. Гребенюк, Л.В. Рибальченко. Дніпро: Дніпроп. держ. ун-т внутріш. справ, 2020,144 с.

ДОДАТОК А

Фрагмент шифрування тексту з візуалізацією перетворенням даних

Введіть початкове значення x0 (0 < x0 < 1): 0.5

Введіть текст для шифрування: Потрібно розробити програму потокового шифрування на основі логістичного генератора хаосу.

[Кроки генерації ключів]:

Ітерація | x (хаотичне) | x*100 (int) | Ключ (8 біт) | Фрагмент тексту (8 біт)

1 | 0.975000000000 | 97 | 01100001 | 10000011
2 | 0.095062500000 | 9 | 00001001 | 11110000
3 | 0.335499922266 | 33 | 00100001 | 11111010
4 | 0.869464925259 | 86 | 01010110 | 00100001
5 | 0.442633109113 | 44 | 00101100 | 01000100
6 | 0.962165255337 | 96 | 01100000 | 00001000
7 | 0.141972779362 | 14 | 00001110 | 10101101
8 | 0.475084386200 | 47 | 00101111 | 00001100
9 | 0.972578927537 | 97 | 01100001 | 01100001
10 | 0.104009713267 | 10 | 00001010 | 11101100
11 | 0.363447601973 | 36 | 00100100 | 00111110
12 | 0.902278426113 | 90 | 01011010 | 00100000
13 | 0.343871064749 | 34 | 00100010 | 10001000
14 | 0.879932646752 | 87 | 01010111 | 00010000
15 | 0.412039617335 | 41 | 00101001 | 11111010
16 | 0.944825587218 | 94 | 01011110 | 00011011
17 | 0.203307768131 | 20 | 00010100 | 11000100
18 | 0.631697506239 | 63 | 00111111 | 00001000
19 | 0.907357490717 | 90 | 01011010 | 01111101

20	0.327833511552	32	00100000	00001100
21	0.859398930996	85	01010101	01100001
22	0.471246392756	47	00101111	11000100
23	0.971775597275	97	01100001	01000010
24	0.106968364683	10	00001010	10000111
25	0.372551921195	37	00100101	00000100
26	0.911652250115	91	01011011	00010000
27	0.314115457403	31	00011111	11111110
28	0.840243053611	84	01010100	00100000
29	0.523515191430	52	00110100	01000011
30	0.972843439511	97	01100001	11101000
31	0.103034418675	10	00001010	01100111
32	0.360431476248	36	00100100	00010000
33	0.899030445993	89	01011001	00100001
34	0.354021342364	35	00100011	10000100
35	0.891891902908	89	01011001	00111100
36	0.376040872098	37	00100101	10001000
37	0.915073124978	91	01011011	01100100
38	0.303085773590	30	00011110	00010000
39	0.823776671006	82	01010010	11111110
40	0.566157802517	56	00111000	00011111
41	0.957930266148	95	01011111	01000100
42	0.157169498249	15	00001111	00101000
43	0.516622263570	51	00110011	01111101
44	0.973922431380	97	01100001	00001110
45	0.099050363236	9	00001001	10100001
46	0.348033616239	34	00100010	11110100
47	0.884934251005	88	01011000	00110010
48	0.397119927372	39	00100111	10000111
49	0.933721193558	93	01011101	11010000

50	0.241355112407	24	00011000	11001110
51	0.714101006276	71	01000111	00011111
52	0.796226960535	79	01001111	00010000
53	0.632773392622	63	00111111	01000100
54	0.906247782225	90	01011010	10001000
55	0.331354683805	33	00100001	01110001
56	0.864079153570	86	01010110	00010001
57	0.458040842750	45	00101101	00100010
58	0.968133773579	96	01100000	00000100
59	0.120318003135	12	00001100	01000011
60	0.412782166901	41	00101001	10000110
61	0.945332893399	94	01011110	01010000
62	0.201546594821	20	00010100	11000010
63	0.627609703254	62	00111110	00011110
64	0.911491478178	91	01011011	11000011
65	0.314631577209	31	00011111	11011000
66	0.840990336544	84	01010100	10011110
67	0.521529802495	52	00110100	01000001
68	0.973192223658	97	01100001	00001111
69	0.101747565933	10	00001010	01100001
70	0.356440495162	35	00100011	10000001
71	0.894623607426	89	01011001	00000100
72	0.367661613002	36	00100100	00111110
73	0.906697550174	90	01011010	10001000
74	0.329928700461	32	00100000	00110000
75	0.862195436985	86	01010110	11110110
76	0.463376415166	46	00101110	00011111
77	0.969768980832	96	01100000	01000011
78	0.114336708126	11	00001011	00101000
79	0.394928918675	39	00100111	10101100

80	0.931944264690	93	01011101	01000001
81	0.247354193586	24	00011000	00001110
82	0.726063376355	72	01001000	11100001
83	0.775691864496	77	01001101	11110100
84	0.678576583817	67	01000011	00110011
85	0.850630574478	85	01010101	10001010
86	0.495526980942	49	00110001	11010001
87	0.974921969192	97	01100001	00000110
88	0.095351580397	9	00001001	00100001
89	0.336412660401	33	00100001	01000011
90	0.870632811060	87	01010111	10001000
91	0.439262145528	43	00101011	10001111
92	0.960612560833	96	01100000	00001111
93	0.147560668331	14	00001110	01100001
94	0.490567418221	49	00110001	11110100
95	0.974653002964	97	01100001	00110011
96	0.096347654432	9	00001001	10000111
97	0.339552657277	33	00100001	11000100
98	0.874600935832	87	01010111	00010000
99	0.427729141608	42	00101010	11001110
100	0.954629999807	95	01011111	00011010
101	0.168915096776	16	00010000	11000011
102	0.547492868743	54	00110110	11011000
103	0.966203266933	96	01100000	01101011
104	0.127352604216	12	00001100	00010000
105	0.433422281819	43	00101011	00100001
106	0.957712889023	95	01011111	10000100
107	0.157945753767	15	00001111	01000010
108	0.518695681271	51	00110011	10000111
109	0.973636838857	97	01100001	11010001

110	0.100105765022	10	00001010	00000010
111	0.351329943243	35	00100011	00011000
112	0.888799135473	88	01011000	00010000
113	0.385457405795	38	00100110	01000100
114	0.923831977040	92	01011100	01011000
115	0.274429175429	27	00011011	01100001
116	0.776559432099	77	01001101	00001111
117	0.676708034017	67	01000011	10100010
118	0.853219655784	85	01010101	00001100
119	0.488419911593	48	00110000	01000011
120	0.974477016055	97	01100001	00101110
121	0.096999088820	9	00001001	00100000

[Результат шифрування – двійковий потік]:

1110001011111001110110110111011101101000011010001010001100100011000000011100110000110100111101010101010010001111101001101000101110100000011001100001101001111010101010100100011111010011010001011101000000110111001001110010110000110100111010110010001110001101001000010100101111110000101110100011101111000100101101101001101000111000101001011010110100111111000011101010110000100111000110110010011101001110011011111010100011010110011010101010000010001101110101100001011111011110111101001001010000010001110000111101100100010011110101111000011101101011000100000100110001100011111001010011101010110111001101011101000100101110100011010110100100001000010100000011000100100011001000111000101100011100000101101010100110111001011100001101111111100000011001110010100001100010110111111010010001101111011011111100010101010010100011101110010101000111111001000101110100111110111000001011000111000000101011011011010011011011010100101100000000100000111011010010000110001000000100011110100100001011100001010110010111001101001111001010010000101011100001010110010111001101001111001010010

ДОДАТОК Б

Слайди мультимедійної презентації

РЕАЛІЗАЦІЯ МЕТОДУ ПОТОКОВОГО ШИФРУВАННЯ НА ОСНОВІ ГЕНЕРАТОРУ ХАОСУ

КВАЛІФІКАЦІЙНА РОБОТА

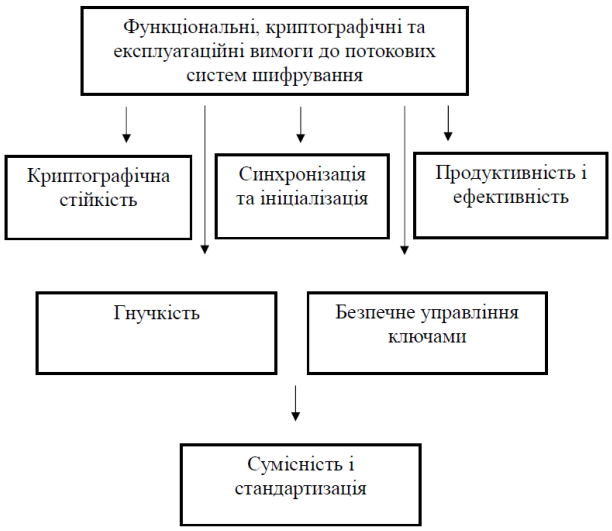
Дипломник: Циганов В.С.
Керівник: Кільдішев В.Й.

2025

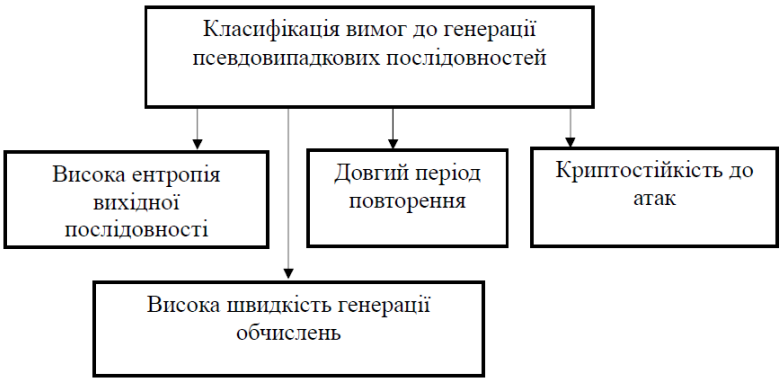
Порівняльний аналіз блокового та потокового шифрування

Параметр	**Блокові шифри (AES, DES)**	**Потокові шифри (RC4, ChaCha20)**
Метод обробки	Шифрують блоки фіксованого розміру	Шифрують дані побітово або побайтово
Швидкість роботи	Вища для великих обсягів даних	Вища для потокових даних
Безпека	Висока у відповідних режимах (AES-256 у GCM)	Висока при одноразовому використанні ключового потоку
Застосування	Диски, файли, бази даних	Мережеві комунікації, стрімінг, VPN
Вимоги до пам'яті	Вищі	Нижчі
Стійкість до атак	Стійкіші при правильному режимі	Вразливі до повторного використання ключів

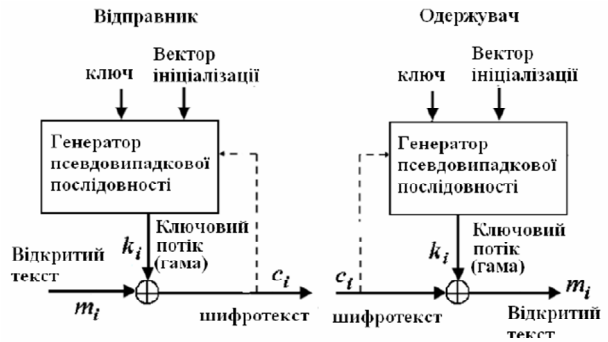
Функціональні, криптографічні та експлуатаційні вимоги до потокових систем шифрування



Вимоги до генерації псевдовипадкових послідовностей потокового шифрування



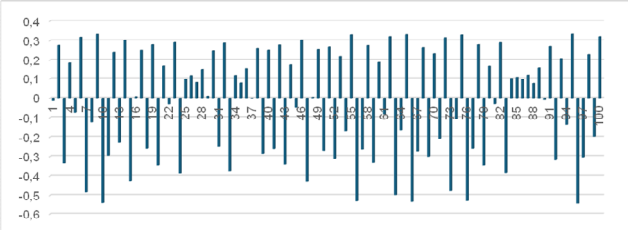
Структурна схема системи потокового шифрування, що працює за принципом Вернама



Порівняння класичних та сучасних потокових шифрів

Алгоритм	Довжина ключа	Стійкість	Використання	Швидкість
Вернама (OTP)	Дорівнює довжині повідомлення	Абсолютна (за ідеального випадкового ключа)	Спецоперації, урядові структури	Низька
RC4	40-256 біт	Вразливий до атак	Старі версії SSL/TLS, WEP	Висока
ChaCha20	256 біт	Висока	VPN, мобільні пристрої	Дуже висока
Salsa20	256 біт	Висока	Протоколи шифрування	Висока
AES-CTR	128-256 біт	Дуже висока	IPsec, TLS, безпечне зберігання даних	Висока

Реалізація центрованого хаотичного процесу



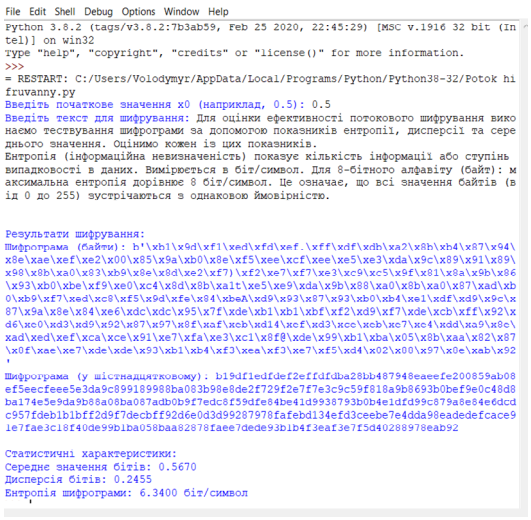
Порівняння хаотичних та класичних PRNG

Характеристика	**Класичні PRNG (наприклад, LFSR, RC4)**	**Хаотичні генератори**
Джерело випадковості	Математичні функції	Динамічні рівняння
Чутливість до початкових умов	Відсутня	Висока
Детермінованість	Так	Так
Стійкість до атак	Відносно висока	Дуже висока
Вимоги до апаратних ресурсів	Мінімальні	Відносно високі

Узагальнені статистичні дані очікувань від шифрограми

Показник	Ідеальне значення	Інтерпретація
Ентропія	≈ 8 біт/символ	Рівномірний розподіл байтів
Середнє бітів	≈ 0.5	50% нулів, 50% одиниць
Дисперсія бітів	≈ 0.25	Максимальна розсіювання, висока випадковість

Результат перевірки статистичних показників шифрограми потокового шифру



Загальний висновок щодо тестування системи потокового шифрування на основі логістичного генератора

Показник	Оцінка	Коментар
Середнє значення бітів	Задовільне (0.5670)	Трохи більше одиниць — варто контролювати цей показник
Дисперсія бітів	Близька до оптимальної (0.2455)	Добра різноманітність
Ентропія шифрограми	Висока (6.34 біт/символ)	Дані добре замасковані, але не повністю рівномірні

ДЯКУЮ ЗА УВАГУ!

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Циганова Владислава Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Кільдішев Віталій Йосипович*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Реалізація методу потокового шифрування на основі генератору хаосу*

Обсяг розрахунково-пояснювальної записки *68* сторінок

Обсяг графічної (презентаційної) частини *12* аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті випускної роботи та технічному завданню. Тематика кваліфікаційної роботи є актуальною для своєї галузі та присвячена реалізації методу потокового шифрування на основі генератору хаосу

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел.

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату ідей у роботі не виявлено

г) перелік позитивних якостей кваліфікаційної роботи *Чітко пояснені властивості детермінованого хаосу, його чутливість до початкових умов і значення для криптографії. Усі етапи – від генерації хаосу до XOR-шифрування та розшифрування – запрограмовані з коментарями. Власний модуль підрахунку статистик: середнє значення бітів, дисперсія, ентропія – реалізований у вигляді окремих функцій, легко адаптувати під інші генератори.*

д) основні недоліки кваліфікаційної роботи *Неповний набір NIST-тестів. У поданих результатах лише кілька основних характеристик (ентропія, середнє, дисперсія) і згадка часткового провалу "Approximate Entropy" без аналізу причин і способів виправлення. Відсутність вимірювань продуктивності. Слабка візуалізація результатів роботи*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис: _____

«23» 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на кваліфікаційну роботу бакалавра

відділення комп'ютерних систем

Циганова Владислава Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Тема кваліфікаційної роботи бакалавра

Реалізація методу потокового шифрування на основі генератору хаосу

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ (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 проєктом (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 досліджуються методи потокового шифрування даних. Предметом дослідження є використання хаотичних генераторів для формування ключових послідовностей у потокових шифрах. Практична значущість дослідження полягає у можливості застосування розробленого методу для криптографічного захисту інформації в різних сферах, включаючи телекомунікації, фінансові системи та військові комунікації._
_____
_____

Оцінка розрахункової частини 4 (чотири)
Оцінка графічної частини 4 (чотири)
Загальна оцінка 4 (чотири)

Прізвище, ім`я, по батькові Кільдішев Віталій Йосипович
_____
_____

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

Підпис _____
«20» 06 2025р.

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

Ми, що нижче підписалися,

Циганов Владислав Сергійович,

здобувач освіти гр. 2БКС-29, та

Кільдішев Віталій Йосипович,
керівник випускної кваліфікаційної роботи,

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

«Реалізація методу потокового шифрування на основі генератору хаосу» (автор роботи – Циганов В.С., керівник роботи – Кільдішев В.Й.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 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 _____ / Циганов В.С. /

Керівник _____ / Кільдішев В.Й. /

«18» червня 2025 р.

ДОВІДКА

кафедри комп'ютерної інженерії
про допуск до захисту кваліфікаційної роботи
здобувача (здобувачки) освіти II курсу
відділення комп'ютерних систем групи 2БКС-29

Циганова Владислава Сергійовича

на тему *Реалізація методу потокового шифрування на основі генератору хаосу*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до кваліфікаційної роботи виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 17.06.2025 р. значення коефіцієнту подібності в роботі становить 12,84%, коефіцієнт цитування – 2,14%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) кваліфікаційної роботи

здобувача (здобувачки) освіти *Циганова В.С.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до кваліфікаційної роботи виконана у повному обсязі. Випускна кваліфікаційна робота відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Зав. кафедри КІ (підпис) *Іванова Л.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/17/2025
Дата редагування 6/17/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Реалізація методу потокового шифрування на основі генератору хаосу

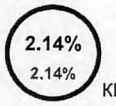
Автор
Циганов Владислав Сергійович
Науковий керівник / Експерт
Кільдішев Віталій Йосипович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

11079
Кількість слів

95990
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	5
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	60

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	254 2.29 %
2	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	47 0.42 %
3	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	34 0.31 %

4	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	28 0.25 %
5	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	23 0.21 %
6	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	23 0.21 %
7	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	21 0.19 %
8	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	19 0.17 %
9	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	18 0.16 %
10	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	17 0.15 %

з домашньої бази даних (10.10 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аналіз сучасних криптографічних алгоритмів та їх ефективності у захисті конфіденційної інформації [redacted] 6/15/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**1119 (75) 10.10 %**

з програми обміну базами даних (0.18 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ВКР_Кучерявий [redacted] 2/12/2025 State University of Trade and Economics (Кафедра комп'ютерних наук та інформаційних систем)	10 (1) 0.09 %
2	Кваснюк_Шпарик.docx [redacted] 12/18/2020 Vasyl Stefanyk Precarpathian National University (VSPNU) (Факультет природничих наук)	10 (1) 0.09 %

з Інтернету (2.56 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download**	**101 (5) 0.91 %**
2	**https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download**	**57 (2) 0.51 %**
3	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	36 (4) 0.32 %
4	https://core.ac.uk/download/pdf/47229182.pdf	21 (3) 0.19 %