Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: "Дослідження результативності атаки зсуву на блокові шифри"

Виконав: студент IV курсу, групи СБ-41

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Задорожний А. В.
підпис (прізвище та ініціали)

Керівник Загородна Н. В.
підпис (прізвище та ініціали)

Нормоконтроль Дроздова Т.В.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Задорожному Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження результативності атаки зсуву на блокові шифри

Керівник роботи Загородна Наталія Володимирівна, к.т.н., доцент, завідувач кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 02 » 05 2025 року № 4/7-361

2. Термін подання студентом завершеної роботи 20.06.2025

3. Вихідні дані до роботи Характеристики експериментального шифру, вимоги до програмної реалізації.

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Блокові шифри та їх значення для інформаційної безпеки. 1.1. Симетрична криптографія в інформаційній безпеці. 1.2. SPN та мережі Фейстеля. 1.3. Блокові шифри. 2. Аналіз атак на блокові шифри. 2.1. Атаки на блокові шифри. 2.1.1. Основні типи атак. 2.1.2. Статистичні методи криптоаналізу. 2.1.3. Алгебраїчні методи криптоаналізу. 2.1.4. Інші методи криптоаналізу. 2.2. Атака зсуву. 2.2.1. Вразливість до атак зсуву. 2.2.2. Алгоритм проведення атаки зсуву. 3. Практичне дослідження результативності атаки зсуву. 3.1. Обгрунтування вибору середовища розробки програмного забезпечення. 3.2. Методика проведення експерименту. 3.3. Програмна реалізація експерименту. 3.4. Результати дослідження. 4. Безпека життєдіяльності, основи охорони праці. 4.1. Вимоги ергономіки до організації робочого місця оператора ПК. 4.2. Поведінкові реакції населення у надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Актуальність теми. 3. Мета, об'єкт, предмет. 4. Атака зсуву. 5. Алгоритм атаки зсуву. 6. Методика дослідження. 7. Варіації шифру. 8. Методика атаки зсуву. 9. Методика зламу грубою силою. 10. Результати експерименту. 11. Кількість перевірених пар. 12. Порівняння результатів. 13. Висновки. 14. Напрямки подальших досліджень

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С.Ю., к.т.н., доц. кафедри МТ		

7. Дата видачі завдання 29.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01 – 02.02	Виконано
2.	Підбір джерел для огляду симетричної криптографії	03.02 – 07.02	Виконано
3.	Опрацювання джерел в галузі дослідження	08.02 – 18.02	Виконано
4.	Проведення аналізу атак на блокові шифри	19.02 – 23.02	Виконано
5.	Проведення аналізу алгоритму проведення атаки зсуву	24.02 – 03.03	Виконано
6.	Написання програмої реалізації алгоритму атаки	04.03 – 11.03	Виконано
7.	Створення методики проведення експерименту	12.03 – 24.03	Виконано
8.	Проведення експерименту та аналіз результатів	25.03 – 06.04	Виконано
9.	Оформлення розділу «Блокові шифри та їх значення для інформаційної безпеки»	07.04 – 17.04	Виконано
10.	Оформлення розділу «Аналіз атак на блокові шифри»	18.04 – 27.04	Виконано
11.	Оформлення розділу «Практичне дослідження результативності атаки зсуву»	28.04 – 17.05	Виконано
12.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	18.05 – 22.05	Виконано
13.	Оформлення кваліфікаційної роботи	23.05 – 01.06	Виконано
14.	Нормоконтроль	02.06 – 15.06	Виконано
15.	Перевірка на плагіат	16.06 – 20.06	Виконано
16.	Попередній захист кваліфікаційної роботи	23.06 – 24.06	Виконано
17.	Захист кваліфікаційної роботи	25.06.2025	Виконано

Студент _____ Задорожний А. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження результативності атаки зсуву на блокові шифри // Кваліфікаційна робота ОР «Бакалавр» // Задорожний Андрій Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБ-41 // Тернопіль, 2025 // С. 60, рис. – 12, табл. – 1, кресл. – - , додат. – 1.

Ключові слова: криптографія, блокові шифри, атака зсуву, шифрування.

У даній кваліфікаційній роботі досліджується результативність атаки зсуву, криптоаналітичного методу, що експлуатує ітераційну природу та симетричні властивості раундових функцій деяких блокових шифрів.

Під час проведення дослідження було виконано теоретичний аналіз блокових шифрів, різних типів криптоаналітичних атак та детально розглянуто алгоритм проведення атаки зсуву. Також було здійснено експериментальні дослідження з використанням програмної реалізації атаки зсуву на моделі блокового шифру з різною кількістю раундів. Оцінка ефективності атаки зсуву проводилась за метриками часу зламу та кількості необхідних перевірених пар відкритого-закритого тексту. Отримані результати були порівняні з результатами зламу грубою силою.

Результати експерименту підтверджують, що ефективність атаки зсуву зростає зі збільшенням кількості раундів шифрування, тоді як ефективність атаки грубої сили знижується.

Висновки та подальші рекомендації щодо перспективних досліджень, що представлені у роботі, будуть сприяти підвищенню рівня безпеки інформаційних систем шляхом врахування специфічних вразливостей блокових шифрів.

ABSTRACT

Study of the Effectiveness of Shift Attacks on Block Ciphers // Thesis of educational level "Bachelor"// Zadorozhnyi Andrii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБ-41 // Ternopil, 2025 // P. 60, fig. - 12, tab. - 1, drawing - , added. – 1.

Keywords: cryptography, block ciphers, slide attack, encryption.

This qualification work examines the effectiveness of the slide attack, a cryptanalytic method that exploits the iterative nature and symmetric properties of round functions of some block ciphers.

In the course of the research, a theoretical analysis of block ciphers and various types of cryptanalytic attacks was carried out, and the algorithm for conducting a slide attack was examined in detail. Experimental studies were also carried out using software implementation of the slide attack on block cipher models with different numbers of rounds. The effectiveness of the slide attack was evaluated based on the metrics of the time required to break the cipher and the number of necessary verified pairs of plaintext and ciphertext. The results obtained were compared with the results of brute force attacks.

The results of the experiment confirm that the effectiveness of the slide attack increases with the number of encryption rounds, while the effectiveness of the brute force attack decreases.

The conclusions and recommendations presented in the paper will contribute to improving the security of information systems by taking into account the specific vulnerabilities of block ciphers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES	—	Advanced Encryption Standard
DES	—	Data Encryption Standard
3DES	—	Triple Data Encryption Standard
MDS	—	Maximum Distance Separable Matrix
SPN	—	Substitution–Permutation Network
IDEA	—	International Data Encryption Algorithm
S-Box	—	Substitution Box
P-Box	—	Permutation Box
KPA	—	Known-Plaintext Attack
CPA	—	Chosen-Plaintext Attack
CCA	—	Chosen-Ciphertext Attack
COA	—	Ciphertext-Only Attack
DCA	—	Differential Cryptanalysis Attack
LCA	—	Linear Cryptanalysis Attack
SCA	—	Side-Channel Attack

ВСТУП

У сучасному цифровому світі, де обмін даними відбувається зі швидкістю світла, питання їхньої конфіденційності та цілісності набуває першочергового значення. Блокові шифри, як основа більшості криптографічних систем, відіграють ключову роль у забезпеченні безпеки інформації, починаючи від захисту фінансових транзакцій і закінчуючи персональними даними. Однак, незважаючи на їхню широку поширеність та постійне вдосконалення, загрози з боку криптоаналітиків залишаються незмінними. Тому дослідження нових та вже відомих методів атак на блокові шифри є критично важливим для розуміння їхніх потенційних вразливостей та розробки більш надійних алгоритмів захисту.

Атака зсуву (slide attack) є одним із таких потужних криптоаналітичних інструментів, який експлуатує симетричні властивості ітераційних шифрів. Її особливість полягає в тому, що вона не залежить від кількості раундів шифрування, що робить її потенційно ефективною навіть проти шифрів з великою кількістю ітерацій, які вважаються стійкими до інших видів атак. Розуміння механізмів цієї атаки та її ефективності надає можливість не лише знаходити слабкі місця в існуючих криптографічних протоколах, але й формувати принципи проектування нових, більш стійких до подібних загроз шифрів [1].

Таким чином, дослідження результативності атаки зсуву на блокові шифри є надзвичайно актуальним завданням. Воно дозволяє не тільки поглибити знання в галузі криптоаналізу, а й надати практичні рекомендації щодо підвищення криптографічної стійкості систем. Отримані в ході проведення експерименту результати можуть бути використані для оцінки безпеки існуючих шифрів, розробки нових стандартів криптографічного захисту та підвищення загального рівня кібербезпеки, що є життєво важливим в умовах постійно зростаючих кіберзагроз.

Метою даної кваліфікаційної роботи є дослідження результативності атаки зсуву на блокові шифри шляхом теоретичного аналізу її механізмів та практичної демонстрації ефективності проти обраного експериментального шифру з різною

кількістю раундів. Кінцевою метою є порівняння ефективності цієї атаки з атакою грубої сили для оцінки її практичної цінності в криптоаналізі.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести всебічний огляд теоретичних основ блокових шифрів та їхніх архітектурних особливостей, що можуть впливати на криптографічну стійкість;

– виконати аналіз існуючих криптоаналітичних атак на блокові шифри, з особливим акцентом на їхні принципи дії та умови застосування;

– детально вивчити алгоритм атаки зсуву, включаючи її математичні основи, умови для успішного проведення та методи виявлення "зсунутих пар";

– розробити та реалізувати програмний модуль для проведення атаки зсуву на обраний блоковий шифр, забезпечивши його функціональність та точність;

– провести серію експериментів з атакою зсуву на блоковий шифр, варіюючи кількість раундів шифрування, та зафіксувати кількість необхідних циклів перебору ключів та необхідних пар відкритий-закритий текст;

– виконати порівняльний аналіз ефективності атаки зсуву з атакою грубої сили, оцінюючи їхні переваги та недоліки з точки зору обчислювальних ресурсів та часу;

– сформулювати висновки щодо практичної цінності атаки зсуву та надати рекомендації щодо підвищення стійкості блокових шифрів до подібних атак.

Об'єктом нашого дослідження є симетричні блокові шифри, архітектура яких складається з повторюваних раундових функцій, що є передумовою для успішного застосування атаки зсуву.

Предметом нашого дослідження є результативність атаки зсуву на блокові шифри. Це включає аналіз умов, за яких атака зсуву може бути застосована, її теоретичні основи та аспекти практичної реалізації. Предмет дослідження охоплює дослідження затрат часу необхідного для проведення атаки, а також його порівняння з показниками атаки грубої сили.

Практичне значення цієї роботи полягає у наданні конкретних даних про ефективність атаки зсуву, що має прямі наслідки для оцінки та підвищення криптографічної безпеки. Результати дослідження допоможуть оцінити реальну стійкість існуючих блокових шифрів, особливо тих, що використовують

ітераційні раундові функції, виявляючи їхні потенційні вразливості до цього типу атак. Це дозволить розробникам та аудиторам безпеки переоцінити ризики та, за потреби, посилити захист.

Крім того, отримані знання та практичні результати сприятимуть розробці нових, більш стійких до атак зсуву блокових шифрів. Розуміння експлуатованих механізмів дозволить інтегрувати контрзаходи на етапі проектування, підвищуючи загальний рівень кібербезпеки. Методика проведення експериментів та розроблений програмний код також можуть бути використані як навчальні матеріали для підготовки фахівців у галузі криптографії та кібербезпеки, поглиблюючи їхнє розуміння криптоаналітичних атак та способів захисту від них.

РОЗДІЛ 1 БЛОКОВІ ШИФРИ ТА ЇХ ЗНАЧЕННЯ ДЛЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

1.1 Симетрична криптографія в інформаційній безпеці

Через зростання обсягів обміну даними через Інтернет, забезпечення безпеки інформації від загроз має першочерговий пріоритет серед завдань кібербезпеки. У сучасному світі забезпечення конфіденційності даних є значним викликом і для наукових дослідників, і для безпосередніх фахівців у галузі кібербезпеки. Конфіденційність даних означає їхній захист від несанкціонованого доступу, викрадення чи компрометації. Цього можна досягти за допомогою криптографії, що включає процеси шифрування та розшифрування даних. Метою криптографії є захист критично важливих даних або документів, що зберігаються на жорсткому диску, або передаються через незахищений канал зв'язку. Шифрування даних – це спосіб забезпечення секретності повідомлень шляхом їхнього перетворення на прихований текст, тоді як зворотний процес отримання оригінального тексту з прихованого називається розшифруванням. Шифрування та розшифрування стають можливими за допомогою використання певних секретних ключів в алгоритмах шифрів. Кожен алгоритм шифрування спрямований на те, щоб максимально ускладнити процес розшифрування без використання ключа, застосованого при шифруванні.

Існує три основні типи криптографічних технік (див. рисунок 1.1):

– симетрична криптографія,

– асиметрична криптографія,

– хешування.

У техніці симетричного ключа як шифрування, так і розшифрування здійснюються на основі одного ключа, який називається приватним або секретним ключем [2]. Для обміну цим приватним ключем між відправником і одержувачем необхідний захищений канал.

Симетричні криптографічні алгоритми поділяються на два типи залежно від способу обробки вхідних даних: блокові шифри та потокові шифри. У системах,

заснованих на блокових шифрах, дані обробляються або шифруються фіксованими групами бітів, що називаються блоками. Натомість, у системах, заснованих на потокових шифрах, дані обробляються як безперервний потік бітів.

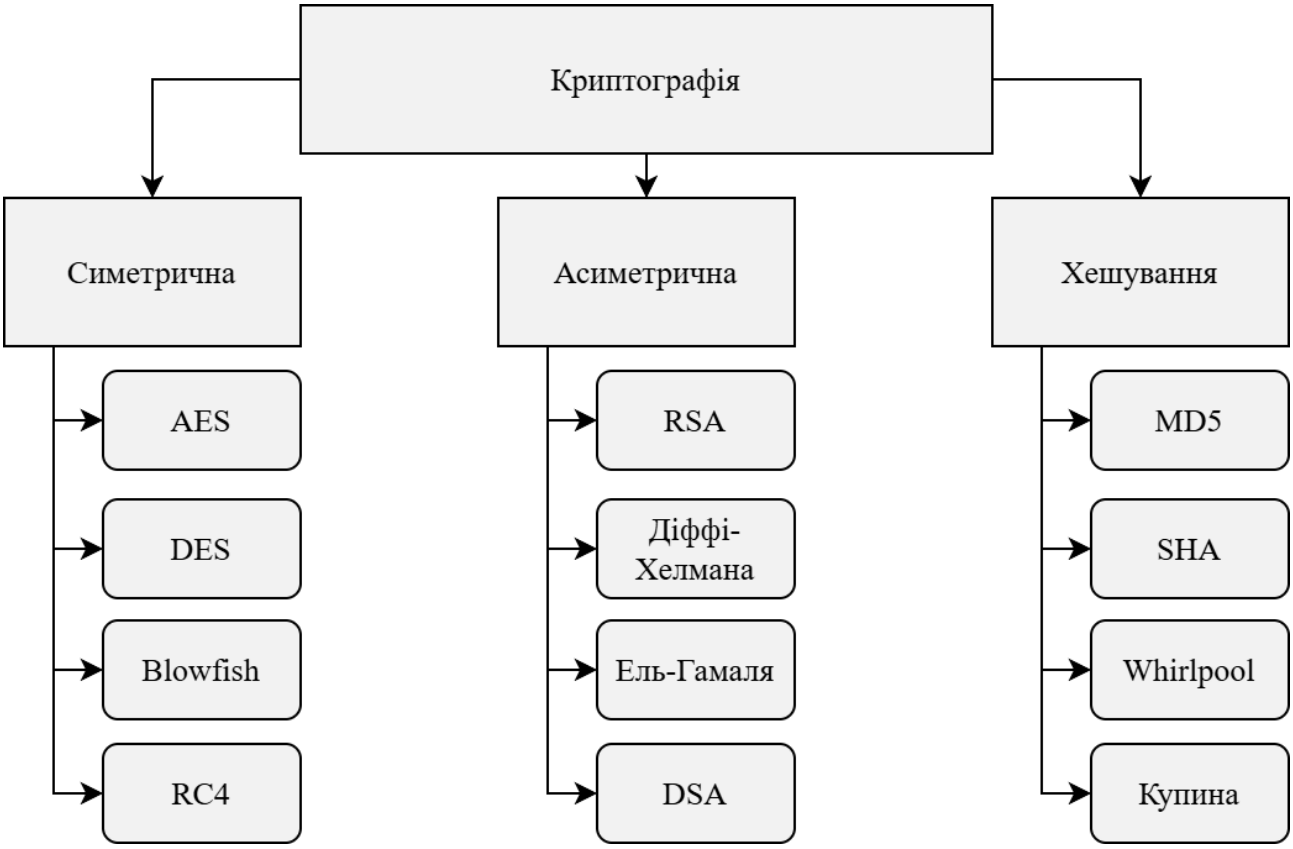

Рисунок 1.1 – Класифікація криптографічних алгоритмів

Більшість сучасних криптографічних алгоритмів ґрунтуються на статичній раундовій функції, яка застосовується багаторазово для багаторазового перемішування даних. Це забезпечує необхідну стійкість проти криптоаналізу. Для досягнення цієї стійкості, криптографічні алгоритми повинні відповідати двом фундаментальним принципам, сформульованим Клодом Шенноном: розсіювання (diffusion) та замішування (confusion) [3].

Розсіювання — це властивість, яка гарантує, що зміна одного біта відкритого тексту призводить до зміни багатьох бітів шифротексту. Це ускладнює виявлення зв'язків між відкритим текстом і шифротекстом, оскільки невеликі зміни на вході поширюються по всьому шифротексту, приховуючи початкові статистичні властивості відкритого тексту. Розсіювання зазвичай

досягається за допомогою операцій перестановки бітів або складних матричних перетворень [4].

Замішування спрямоване на приховування зв'язку між статистичними властивостями шифротексту та значенням ключа. Це досягається шляхом використання нелінійних операцій, таких як таблиці підстановок (S-boxes або S-блоки). S-блоки перетворюють фіксовану кількість вхідних бітів на фіксовану кількість вихідних бітів, забезпечуючи складні нелінійні взаємозв'язки. Кожен елемент S-блоку є унікальним відображенням, що ускладнює відновлення ключа. Приклад S-блоку, який приймає 4 біти на вході і перетворює їх на 4 інші біти на виході зображений на рисунку 1.2 у вигляді таблиці підстановки [5].

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**
00	63	7c	77	7b	f2	6b	6f	c5
10	ca	82	c9	7d	fa	59	47	f0
20	b7	fd	93	26	36	3f	f7	cc
30	04	c7	23	c3	18	96	05	9a
40	09	83	2c	1a	1b	6e	5a	a0
50	53	d1	00	ed	20	fc	b1	5b
60	d0	ef	aa	fb	43	4d	33	85
70	51	a3	40	8f	92	9d	38	f5

Рисунок 1.2 – Приклад 4-бітного S-блоку

Обидва принципи є взаємодоповнювальними та критично важливими для стійкості шифру. Сучасні криптографічні алгоритми, такі як AES (Advanced Encryption Standard), 3DES (Triple Data Encryption Standard) та BLOWFISH, широко використовуються для забезпечення конфіденційності даних [6]. Вони інтегрують ці принципи для досягнення високого рівня безпеки. Наприклад, у таких шифрах як AES, замішування реалізується за допомогою S-блоків, а розсіювання – через складні операції з бітами.

Зокрема, блокові шифри використовують різні типи операцій для забезпечення розсіювання та замішування. Для розсіювання часто

застосовуються такі методи, як бітові перестановки (P-boxes), які просто переставляють біти у визначеному порядку, та матриці дифузії, що поширюють зміну одного біта на кілька інших. Прикладом матриці дифузії є MDS (Maximum Distance Separable) матриця, яка є оптимальною для розсіювання [7].

Для замішування ключовими компонентами є S-блоки. Вони можуть бути реалізовані як глобальні (що застосовуються до всього блоку даних) або локальні (що застосовуються до менших частин блоку). Метою S-блоків є створення нелінійних перетворень, які приховують статистичні властивості відкритого тексту. Якість S-блоку оцінюється за його бієктивністю (кожен вхід має унікальний вихід), низькою лінійністю та низьким диференціальним рівнем.

Слід зазначити, що як блокові, так і потокові шифри та їхні режими шифрування повинні мати здатність забезпечувати сильний лавинний ефект (Avalanche Effect) [8]. Цей ефект означає, що мінімальна зміна у вхідних даних (один біт) призводить до значних змін у вихідних даних (багатьох бітах). Чим сильніший лавинний ефект, тим важче криптоаналітику виявити закономірності та відновити ключ.

Існуючі симетричні криптографічні примітиви, такі як DES (Data Encryption Standard), AES (Advanced Encryption Standard), SNOW 3G та BLOWFISH, широко застосовуються в сучасних системах зв'язку та безпеки [9]. AES, зокрема, є стандартом і широко використовується завдяки своїй надійності та ефективності. Ці примітиви зазвичай базуються на структурі підстановки-перестановки (SPN), де чергуються нелінійні S-блоки (для замішування) та лінійні P-блоки (для розсіювання), або на мережі Фейстеля (Feistel Network, FN) [10].

Наприклад, AES використовує SPN-структуру, де кожен раунд включає операції SubBytes (замішування за допомогою S-блоків), ShiftRows (зсув рядків для розсіювання), MixColumns (перемішування стовпців для подальшого розсіювання) та AddRoundKey (додавання раундового ключа). Ці операції ретельно розроблені для забезпечення високого рівня розсіювання та замішування, що робить AES дуже стійким до відомих атак.

Особливе значення має питання, чому більшість симетричних алгоритмів використовують ітераційну структуру. Відповідь полягає в тому, що

багаторазове повторення однієї і тієї ж раундової функції забезпечує високий рівень безпеки за відносно простих операцій [11]. Наприклад, дизайн AES дозволяє досягти чудового розсіювання, що робить його стійким до диференціального та лінійного криптоаналізу. Це пояснює ефективність таких алгоритмів, як AES, які є економічними з точки зору обчислень, але забезпечують надійний захист.

1.2 SPN та мережі Фейстеля

Блокові шифри, як ключовий елемент симетричної криптографії, базуються на ітераційному застосуванні певних трансформацій до блоків даних. Двома основними архітектурними підходами до побудови таких шифрів є мережі заміни-перестановки (SPN) та мережі Фейстеля. Ці структури забезпечують необхідні криптографічні властивості, такі як розсіювання та замішування, шляхом послідовного застосування простих, але ефективних операцій. Вибір архітектури суттєво впливає на стійкість шифру до різноманітних криптоаналітичних атак [12].

Мережа заміни-перестановки (SPN) є поширеним типом конструкції блокового шифру, який широко використовується в сучасній криптографії, включаючи такі стандарти як AES (Advanced Encryption Standard). Ця структура складається з кількох повторюваних раундів, кожен з яких включає чергування двох основних типів операцій:

– заміни (Substitution),

– перестановки (Permutation).

Операції заміни реалізуються за допомогою S-блоків (Substitution boxes). S-блоки є нелінійними компонентами, які приймають певну кількість вхідних бітів і перетворюють їх на таку ж (або іншу) кількість вихідних бітів. Їхня основна функція – забезпечити замішування (confusion), тобто приховати зв'язок між статистичними властивостями шифротексту та ключа. Нелінійність S-блоків робить їх критично важливими для опору шифру лінійному та диференціальному

криптоаналізу. Кожен S-блок, по суті, є таблицею підстановки, де кожному вхідному значенню відповідає унікальне вихідне [13].

Операції перестановки реалізуються за допомогою P-блоків (Permutation boxes) або інших лінійних перетворень. Їхнє завдання – забезпечити розсіювання (diffusion), тобто поширити вплив одного зміненого біта вхідних даних на якомога більшу кількість бітів вихідних даних. P-блоки зазвичай просто переставляють біти або групи бітів між виходами S-блоків. Це гарантує, що вихідні біти одного S-блоку стають входами для різних S-блоків у наступному раунді, тим самим розподіляючи інформацію.

У типовій SPN-структурі кожен раунд починається з операції додавання раундового ключа (AddRoundKey), яка зазвичай виконується за допомогою операції виключного АБО (XOR) між проміжним станом блоку даних та раундовим ключем. Цей крок забезпечує залежність шифротексту від секретного ключа [14]. Після додавання ключа дані проходять через шари заміни (S-блоки), а потім через шари перестановки (P-блоки або інші лінійні перетворення). Цей цикл повторюється певну кількість раундів (див. рисунок 1.3).

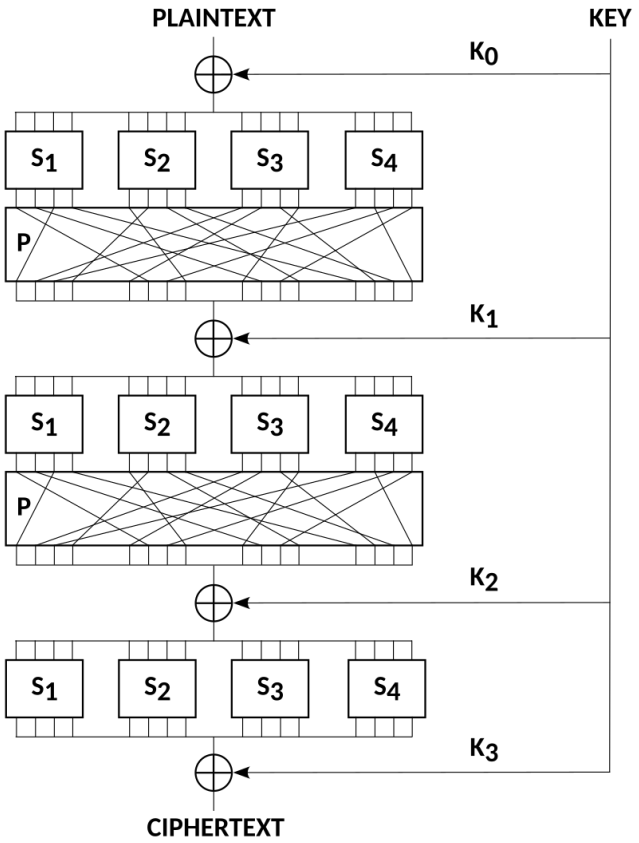

Рисунок 1.3 – Типова SPN-структура

Надійність SPN-шифрів значною мірою залежить від якості використовуваних S-блоків та ефективності шару розсіювання. Добре спроектовані S-блоки повинні мати високу нелінійність, низький диференціальний рівень та задовольняти критеріям лавинного ефекту. Ефективність розсіювання, у свою чергу, визначається тим, наскільки швидко зміна одного біта поширюється на весь блок.

Перевагою SPN-архітектури, порівняно з іншими конструкціями блокових шифрів, є її прямий та гнучкий характер. Це означає, що процес шифрування та розшифрування в SPN не вимагає ідентичного порядку операцій або функцій для обернення. Хоча для зручності та спрощення реалізації часто застосовується "обернена" SPN-структура, де кожен раунд розшифрування є оберненим до відповідного раунду шифрування, ця гнучкість дозволяє розробникам створювати складніші та, потенційно, більш стійкі схеми [15]. Можливість використовувати різні набори операцій для шифрування та розшифрування без необхідності дотримання суворої симетрії, як у мережах Фейстеля, відкриває простір для креативного дизайну [16].

Саме ця структурна гнучкість та ефективність дозволяють створювати на базі SPN дуже потужні та стійкі до криптоаналітичних атак шифри. Яскравим прикладом цього є Advanced Encryption Standard (AES), який широко визнаний як еталон безпеки в безлічі сучасних додатків – від захисту інтернет-трафіку до конфіденційного зберігання даних. Завдяки ретельно продуманому поєднанню нелінійних S-блоків та ефективних лінійних перетворень для розсіювання, AES демонструє виняткову стійкість. Тому, коли мова йде про проектування нових криптографічних алгоритмів, саме SPN-архітектура часто отримує перевагу, оскільки вона надає розробникам інструменти для побудови високонадійних та продуктивних шифрів.

Хоча SPN-структури є надзвичайно ефективними, іншою фундаментальною та не менш важливою архітектурою для побудови блокових шифрів є мережа Фейстеля. Ця конструкція, названа на честь її винахідника Хорста Фейстеля, вирізняється своєю симетричністю та простотою реалізації як шифрування, так і

розшифрування, що робить її особливо привабливою для багатьох криптографічних застосувань, включаючи такі відомі шифри, як DES.

Мережа Фейстеля – це архітектурний шаблон для побудови блокових шифрів, який забезпечує можливість легкої реалізації як шифрування, так і розшифрування за допомогою одних і тих же раундових функцій [17]. Блок відкритого тексту ділиться на дві рівні частини – ліву L_0 та праву R_0. Кожен раунд операції Фейстеля використовує ці дві частини таким чином, що розшифрування є простим оберненням операцій шифрування.

У типовому раунді мережі Фейстеля i, ліва частина L_i стає правою частиною наступного раунду R_{i+1}, а нова ліва частина L_{i+1} обчислюється як сума по модулю 2 (XOR) поточної правої частини R_i з результатом застосування раундової функції f до лівої частини L_i та раундового ключа K_i (див. рисунок 1.4). Тобто, згідно формул (1.1) та (1.2):

$$L_{i+1} = R_i \tag{1.1}$$

$$R_{i+1} = L_i \oplus f(R_i, K_i) \tag{1.2}$$

Ключова особливість мережі Фейстеля полягає у симетричності процесу шифрування та розшифрування. Щоб розшифрувати блок, потрібно просто застосувати ті ж самі раундові функції у зворотному порядку, використовуючи раундові ключі у зворотному порядку. Наприклад, якщо ми маємо L_{i+1} та R_{i+1}, ми можемо отримати L_i та R_i за формулами (1.3) та (1.4):

$$L_i = R_{i+1} \oplus f(L_{i+1}, K_i) \tag{1.3}$$

$$R_i = L_{i+1} \tag{1.4}$$

Ця властивість робить мережі Фейстеля дуже привабливими для апаратних реалізацій, де необхідно мінімізувати розмір схеми та складність, оскільки одні й ті самі логічні блоки можуть використовуватися як для шифрування, так і для розшифрування.

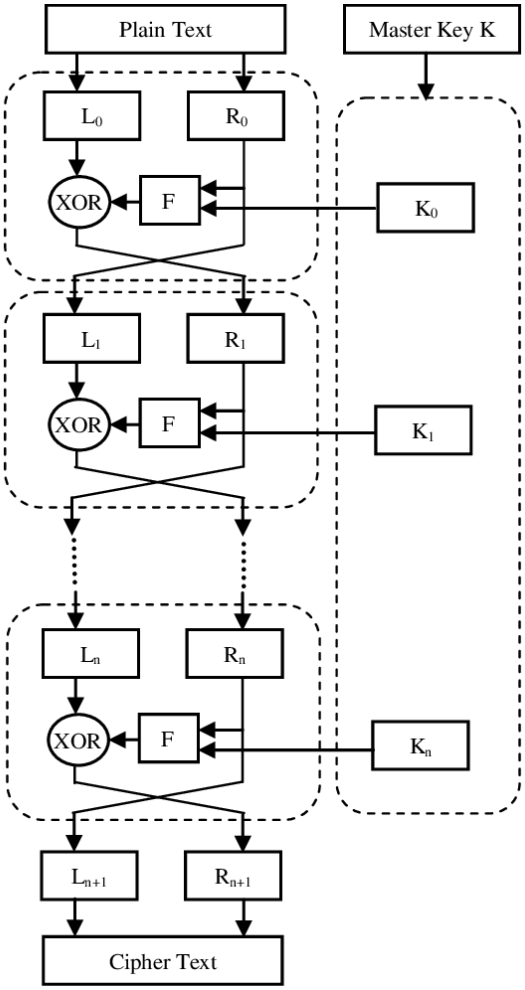

Рисунок 1.4 – Типова структура мережі Фейстеля

Сама раундова функція (f) є серцем мережі Фейстеля і відповідає за забезпечення замішування та розсіювання. Зазвичай вона містить нелінійні компоненти, такі як S-блоки, та лінійні перетворення для поширення бітових змін. Якість цієї функції безпосередньо впливає на криптографічну стійкість усього шифру. На відміну від SPN, де всі біти блоку проходять через S-блоки в кожному раунді, у мережі Фейстеля S-блоки знаходяться лише всередині функції f, яка обробляє лише половину блоку [18].

Однією з переваг мережі Фейстеля є те, що функція f не обов'язково повинна бути оберненою. Навіть якщо f не є бієктивною, загальна операція шифрування залишається оборотною завдяки структурі Фейстеля. Це надає більшу гнучкість при розробці раундової функції, дозволяючи використовувати простіші та швидші незворотні перетворення.

1.3 Блокові шифри

Блокові шифри є наріжним каменем сучасної симетричної криптографії, перетворюючи фіксовані блоки відкритого тексту на шифротекст за допомогою секретного ключа. Їхня ефективність і безпека базуються на принципах розсіювання та замішування, що досягається за допомогою ітераційних раундових функцій. Історично та архітектурно блокові шифри еволюціонували навколо двох основних структур, описаних раніше: мереж заміни-перестановки (SPN) та мереж Фейстеля [19].

Найвидатнішим представником SPN-архітектури є Advanced Encryption Standard (AES). Прийнятий як стандарт у 2001 році, AES став одним із найпоширеніших блокових шифрів у світі, замінивши DES. AES оперує з блоками повідомлень розміром 128 біт і підтримує різні довжини ключів: 128, 192 або 256 біт. Його раундова функція складається з чотирьох основних операцій:

– subBytes (заміни байтів через S-блок),

– shiftRows (зсув рядків),

– mixColumns (перемішування стовпців)

– addRoundKey (додавання раундового ключа).

Висока нелінійність S-блоків та ефективне розсіювання забезпечують AES винятковою стійкістю до відомих криптоаналітичних атак, що робить його еталоном сучасної криптографії.

В свою чергу, найбільш відомим блоковим шифром на основі мережі Фейстеля є Data Encryption Standard (DES). Розроблений IBM і прийнятий як федеральний стандарт у 1977 році, DES шифрує блоки даних розміром 64 біти за допомогою 56-бітового ключа (плюс 8 бітів парності). Він складається з 16 раундів мережі Фейстеля, де раундова функція включає розширення бітів, додавання ключа, S-блоки та P-блок. Хоча довжина ключа DES виявилася недостатньою для протистояння сучасним атакам грубої сили (що призвело до розробки 3DES, Triple DES), його архітектура стала основою для багатьох інших

криптографічних алгоритмів і відіграла значну роль у розвитку криптоаналізу [20].

Крім AES і DES, існує безліч інших блокових шифрів, що базуються на цих архітектурах. Наприклад, Blowfish, розроблений Брюсом Шнайером, є 64-бітовим блоковим шифром з ключем змінної довжини (від 32 до 448 біт), що також використовує мережу Фейстеля. Він відрізняється використанням великих, залежних від ключа S-блоків, які обчислюються перед початком шифрування [21]. Це робить його досить швидким, але з повільною ініціалізацією.

Інший приклад – IDEA (International Data Encryption Algorithm), 64-бітовий блоковий шифр з 128-бітовим ключем. Він не належить до чистої SPN або Фейстеля, але використовує комбінацію операцій XOR, додавання по модулю 2 та множення по модулю 2, що забезпечує високий рівень безпеки та стійкість до лінійного та диференціального криптоаналізу [22].

Нарешті, слід зазначити, що постійне вивчення та аналіз блокових шифрів, включаючи ті, що базуються на SPN та мережах Фейстеля, є критично важливим. Кожна нова атака, як-от атака зсуву, допомагає краще зрозуміти потенційні вразливості та розробити більш стійкі алгоритми, що є невід'ємною частиною еволюції криптографії у відповідь на зростаючі кіберзагрози.

РОЗДІЛ 2 АНАЛІЗ АТАК НА БЛОКОВІ ШИФРИ

2.1 Атаки на блокові шифри

2.1.1 Основні типи атак

Безпека блокових шифрів є постійною темою досліджень у криптографії, де криптоаналітики розробляють різноманітні методи для виявлення їхніх вразливостей. Ці атаки можна класифікувати за кількома критеріями, зокрема за обсягом інформації, доступної криптоаналітику, та за основними принципами їхньої роботи. Розуміння цих класифікацій є ключовим для оцінки стійкості шифру та розробки ефективних контрзаходів.

Однією з найпоширеніших класифікацій атак є поділ за типом доступу до інформації, яку має криптоаналітик. До них належать:

– атака з відомим відкритим текстом (Known-Plaintext Attack, KPA): Криптоаналітик має доступ до кількох пар "відкритий текст — відповідний шифротекст", але не може самостійно вибирати відкриті тексти. Його мета — відновити ключ або знайти алгоритм, який дозволяє розшифровувати нові шифротексти.

– атака з вибраним відкритим текстом (Chosen-Plaintext Attack, CPA): Криптоаналітик може вибирати відкриті тексти для шифрування та отримувати відповідні шифротексти. Це дає йому значно більше контролю та інформації, що робить цей тип атаки потужнішим за KPA.

– атака з вибраним шифротекстом (Chosen-Ciphertext Attack, CCA): Криптоаналітик може вибирати шифротексти для розшифрування та отримувати відповідні відкриті тексти. Це дозволяє аналізувати поведінку шифру при розшифруванні, що може виявити певні вразливості.

– атака лише на основі шифротексту (Ciphertext-Only Attack, COA): Найскладніший сценарій, де криптоаналітик має доступ лише до шифротекстів і не має жодної інформації про відповідні відкриті тексти. Цей тип атаки

спирається на статистичні властивості мови або наявність відомих шаблонів у зашифрованих даних.

Крім класифікації за доступом до інформації, атаки на блокові шифри також поділяються за методологією їхнього проведення. До основних категорій належать статистичні, алгебраїчні та інші спеціалізовані атаки.

Статистичні атаки експлуатують статистичні властивості шифру, що можуть зберігатися або проявлятися після шифрування. Найвідомішими прикладами є диференціальний криптоаналіз (Differential Cryptanalysis, DCA) та лінійний криптоаналіз (Linear Cryptanalysis, LCA). DCA аналізує відмінності між парами відкритих текстів та відповідними відмінностями у шифротекстах. Він шукає "диференціальні характеристики", які з високою ймовірністю вказують на певні ключові біти. LCA, натомість, шукає лінійні апроксимації нелінійної функції шифру, що дозволяє виводити залежності між бітами відкритого тексту, шифротексту та ключа [23].

Алгебраїчні атаки розглядають шифр як систему алгебраїчних рівнянь, де невідомими є біти ключа. Якщо шифр можна ефективно представити у вигляді системи рівнянь (наприклад, поліноміальних над скінченним полем), то, теоретично, можна спробувати розв'язати цю систему для отримання ключа. Хоча це часто є обчислювально складним завданням для добре спроектованих шифрів, розвиток обчислювальної техніки та нові алгоритми розв'язання систем рівнянь роблять цей напрямок постійно актуальним.

До інших спеціалізованих атак належить широкий спектр методів, кожен з яких використовує специфічні вразливості або властивості шифру. Наприклад, атака грубої сили (Brute-Force Attack) є найпрямішою, але часто обчислювально витратною. Вона полягає у послідовному переборі всіх можливих ключів доти, доки не буде знайдено правильний ключ. Її ефективність прямо залежить від довжини ключа.

Також існують такі атаки, як атака зсуву (Slide Attack), яка є предметом даного дослідження. Вона використовує той факт, що деякі ітераційні шифри мають ідентичні раундові функції, що дозволяє знаходити "зсунуті пари" (plaintexts P та P′ такі, що P′ = Enc(P, K) для певного ключа K). Атака зсуву не

залежить від кількості раундів, що робить її небезпечною для шифрів з великою кількістю ітерацій, які спроектовані без урахування цієї вразливості.

Окрім вищезгаданих, існують також атаки за часом (Timing Attacks), які аналізують час виконання криптографічних операцій; атаки за споживанням потужності (Power Analysis Attacks), що вивчають зміни в енергоспоживанні пристрою; та атаки з використанням побічних каналів (Side-Channel Attacks), які експлуатують будь-яку доступну інформацію, що "витікає" з фізичної реалізації криптосистеми [24]. Усі ці методи постійно вдосконалюються, що вимагає від розробників криптографічних алгоритмів постійної адаптації та підвищення стійкості своїх розробок.

2.1.2 Статистичні методи криптоаналізу

Статистичні методи криптоаналізу є одним із найбільш потужних та фундаментальних підходів до виявлення вразливостей у криптографічних алгоритмах, особливо в блокових шифрах. Вони ґрунтуються на аналізі статистичних властивостей шифротексту та його взаємозв'язків з відкритим текстом або ключем. Ці методи шукають не випадкові закономірності, які можуть виникнути внаслідок процесу шифрування, навіть якщо на перший погляд дані здаються випадковими.

Основна ідея статистичного криптоаналізу полягає у виявленні відхилень від ідеальної випадковості. Для стійкого шифру шифротекст повинен виглядати як послідовність випадкових бітів. Однак, якщо криптоаналітик може виявити будь-які статистичні кореляції між вхідними та вихідними даними або між шифротекстом і ключем, це може бути використано для відновлення частини або всього ключа [25].

Двома найвідомішими та найефективнішими статистичними атаками на блокові шифри є диференціальний криптоаналіз (Differential Cryptanalysis, DCA) та лінійний криптоаналіз (Linear Cryptanalysis, LCA). Обидва методи є атаками з відомим або вибраним відкритим текстом і були надзвичайно впливовими у проектуванні та оцінці безпеки сучасних шифрів.

Диференціальний криптоаналіз (DCA) зосереджується на аналізі відмінностей у парах відкритого тексту та відповідних відмінностей у шифротекстах. Він працює з поняттям "диференціалу" – XOR-суми двох відкритих текстів та відповідної XOR-суми їхніх шифротекстів. Криптоаналітик шукає "диференціальні характеристики", які з високою ймовірністю вказують на певні зміни в проміжних станах шифру. Наприклад, якщо відомо, що певний вхідний диференціал з високою ймовірністю дає певний вихідний диференціал після кількох раундів шифрування, це може бути використано для визначення бітів раундового ключа. DCA був успішно застосований до DES та багатьох інших шифрів [26].

Лінійний криптоаналіз (LCA), у свою чергу, шукає лінійні апроксимації нелінійної функції шифру. Він намагається знайти лінійне співвідношення (зазвичай XOR-суму бітів) між певними бітами відкритого тексту, шифротексту та ключа, яке виконується з ймовірністю, відмінною від 0.5 (тобто, відмінною від чистої випадковості). Якщо таке співвідношення знайдено з достатнім "зміщенням" від 0.5, криптоаналітик може використати це для статистичного виведення бітів ключа. LCA також був ефективно застосований до DES та є потужним інструментом для аналізу SPN-шифрів.

Обидва методи, DCA та LCA, вимагають значної кількості пар "відкритий текст – шифротекст" для проведення статистичного аналізу. Чим вища ймовірність диференціальної характеристики або зміщення лінійної апроксимації, тим менше даних потрібно для успішної атаки. Тому розробники шифрів прагнуть створити S-блоки та інші компоненти, які мають низьку диференціальну та лінійну ймовірності, щоб ускладнити проведення цих атак. Завдяки цим методам, сучасні шифри, такі як AES, були розроблені з урахуванням стійкості до DCA та LCA, що робить їх набагато надійнішими.

2.1.3 Алгебраїчні методи криптоаналізу

Алгебраїчні методи криптоаналізу представляють собою принципово інший підхід до виявлення вразливостей у шифрах порівняно зі статистичними. Вони

ґрунтуються на математичному моделюванні криптографічного алгоритму як системи алгебраїчних рівнянь. У цій парадигмі біти ключа та проміжні стани шифрування розглядаються як змінні, а операції шифрування (наприклад, XOR, додавання, множення, S-блоки) перетворюються на відповідні алгебраїчні рівняння. Метою атаки є розв'язання цієї системи рівнянь для отримання секретного ключа.

Основна ідея полягає у вираженні кожного кроку шифрування у вигляді системи поліноміальних рівнянь над певним скінченним полем (наприклад, GF(2), де операції виконуються за модулем 2, а біти є елементами поля). Кожен S-блок, що є нелінійним перетворенням, може бути точно представлений як набір таких поліноміальних рівнянь. Чим простіші алгебраїчні представлення внутрішніх компонентів шифру, тим легше побудувати таку систему.

Після побудови системи рівнянь, що описує шифрування, криптоаналітик намагається її розв'язати. Якщо кількість рівнянь перевищує кількість невідомих (бітів ключа), і система не є надто складною, теоретично можна знайти унікальний розв'язок. Однак, на практиці, системи, що виникають з реальних блокових шифрів, є високонелінійними та надрозмірними, що робить їх розв'язання обчислювально дуже складним, навіть для сучасних комп'ютерів.

Одним з найвідоміших методів розв'язання таких систем є алгоритм XL (eXtended Linearization) та його варіанти, такі як XSL (eXtended Sparse Linearization). Ці алгоритми намагаються лінеаризувати систему поліноміальних рівнянь шляхом додавання нових рівнянь, отриманих множенням існуючих рівнянь на змінні. Мета полягає в тому, щоб зрештою отримати достатньо лінійних рівнянь, які можна розв'язати за допомогою методів лінійної алгебри, таких як виключення Гауса [27].

Іншим підходом є використання атакування Графнерової бази (Gröbner Basis Attack). Графнерова база є спеціальним набором поліномів, який дозволяє спростити систему поліноміальних рівнянь та ефективніше її розв'язати. Хоча цей метод є дуже потужним у теорії, його обчислювальна складність є експоненційною від кількості змінних, що обмежує його застосування для шифрів з великим розміром ключа або блоку.

Стійкість шифру до алгебраїчних атак значною мірою залежить від алгебраїчної складності його внутрішніх компонентів, особливо S-блоків. Добре спроектовані S-блоки мають високий алгебраїчний ступінь та складну структуру, що ускладнює їхнє представлення простими поліномами. Також, використання складних перетворень у раундах, які створюють багаточленні залежності високого ступеня, робить алгебраїчні атаки менш практичними. Незважаючи на обчислювальні труднощі, алгебраїчні методи залишаються важливою областю досліджень, оскільки будь-які прориви в алгоритмах розв'язання поліноміальних систем можуть мати значний вплив на безпеку криптографічних алгоритмів.

2.1.4 Інші методи криптоаналізу

Окрім статистичних та алгебраїчних підходів, існує широкий спектр інших методів криптоаналізу, які експлуатують різні вразливості криптографічних систем. Ці методи можуть бути класифіковані як "атаки побічними каналами" або "атаки, що використовують специфічні властивості реалізації", і вони часто є не менш, а іноді й більш небезпечними, ніж класичні математичні атаки.

Однією з найпряміших, але водночас найменш витончених атак є атака грубої сили (Brute Force Attack). Цей метод полягає у послідовному переборі всіх можливих значень ключа доти, доки не буде знайдено правильний ключ, який дозволить розшифрувати шифротекст до зрозумілого відкритого тексту. Ефективність атаки грубої сили прямо залежить від довжини ключа: зі збільшенням довжини ключа на один біт, кількість можливих варіантів подвоюється, що швидко робить атаку обчислювально нездійсненною для ключів достатньої довжини (наприклад, 128 біт і більше).

Окрему групу становлять атаки побічними каналами (Side-Channel Attacks). Ці атаки не намагаються зламати криптографічний алгоритм безпосередньо, а замість цього експлуатують інформацію, яка "витікає" з фізичної реалізації криптосистеми. Такою інформацією можуть бути варіації у споживанні потужності, час виконання операцій, електромагнітне випромінювання, акустичні шуми або навіть температурні зміни. Навіть найменші залежності між

цими фізичними параметрами та секретним ключем можуть бути використані для його відновлення.

Яскравим прикладом побічної атаки є атака за часом (Timing Attack). Вона ґрунтується на аналізі точного часу, який криптографічний пристрій витрачає на виконання операцій шифрування або розшифрування. Якщо час виконання залежить від значення секретного ключа або від конкретних даних, що обробляються, криптоаналітик може, вимірюючи ці часові відмінності, отримати інформацію про ключ. Наприклад, операції множення або модульні операції можуть займати різний час залежно від значень операндів.

Ще однією потужною атакою побічними каналами є атака за споживанням потужності (Power Analysis Attack). Вона вивчає коливання електричного струму, що споживається криптографічним пристроєм під час виконання операцій. Ці коливання корелюють з активністю внутрішніх компонентів пристрою та обробкою бітів даних. Аналізуючи ці "сигнатури потужності", можна отримати інформацію про біти ключа. Розрізняють прості атаки за потужністю (Simple Power Analysis, SPA) та диференціальні атаки за потужністю (Differential Power Analysis, DPA), останні з яких є більш складними та вимагають статистичного аналізу багатьох вимірювань.

Також існують атаки з використанням несправностей (Fault Injection Attacks). Вони полягають у навмисному внесенні контрольованих несправностей у роботу криптографічного пристрою (наприклад, шляхом короткочасного зміни напруги живлення, частоти тактового сигналу, або впливу лазером). Метою є отримання некоректних, але передбачуваних виходів, які потім аналізуються для виявлення інформації про секретний ключ. Цей метод вимагає фізичного доступу до пристрою, але може бути надзвичайно ефективним.

Всі ці "інші" методи криптоаналізу підкреслюють важливість комплексного підходу до безпеки криптосистем. Недостатньо мати лише математично стійкий алгоритм; необхідно також звертати увагу на його імплементацію, щоб уникнути витоку інформації через побічні канали. Захист від таких атак вимагає застосування спеціальних заходів, таких як рандомізація операцій, маскування даних, балансування споживання потужності та фізичний захист пристроїв.

2.2 Атака зсуву

2.2.1 Вразливість до атак зсуву

Атака зсуву (Slide Attack) є потужним методом криптоаналізу, який експлуатує специфічну вразливість ітераційних блокових шифрів, де всі раунди шифрування використовують одну й ту саму раундову функцію, часто з циклічним або повторюваним ключем. Цей тип атаки був вперше представлений Девідом Вагнером та Алексом Бірюковим у 1999 році. Основна ідея полягає у пошуку "зсунутих пар" (slide pairs) – двох пар "відкритий текст - шифротекст" (P_0,C_0) та (P_1,C_1), де P_1 є результатом шифрування P_0 одним раундом шифрування, тобто $P_1 = \Pi(P_0) \oplus k$, а C_1 є результатом шифрування C_0 одним раундом шифрування також: $C_1 = \Pi(C_0) \oplus k$.

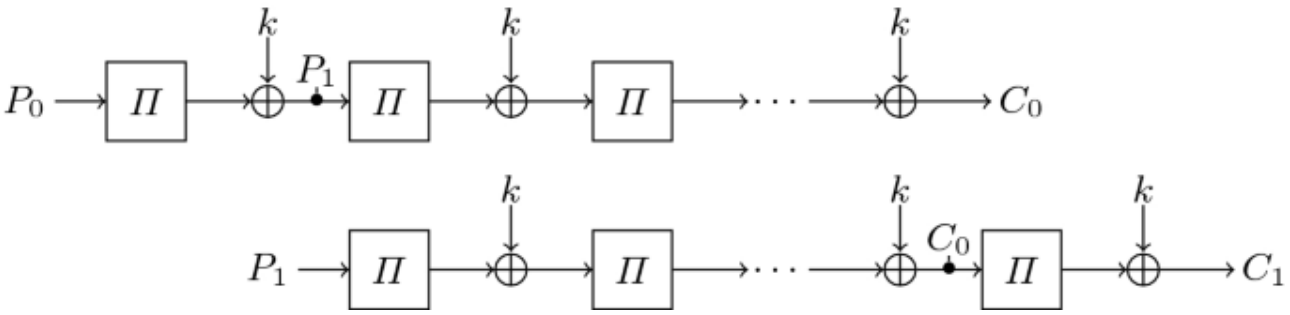

Рисунок 2.1 – Принцип атаки зсуву

Мета атаки зсуву полягає у використанні цих зсунутих пар для отримання інформації про ключ, зазвичай, шляхом атаки на один раунд шифрування, що значно знижує обчислювальну складність порівняно з атакою на повний багато раундовий шифр. Важливою особливістю атаки зсуву є те, що її ефективність не залежить від кількості раундів шифрування. Це відрізняє її від більшості інших криптоаналітичних методів, таких як диференціальний або лінійний криптоаналіз, ефективність яких зазвичай падає експоненційно зі збільшенням кількості раундів.

До атак зсуву вразливі, перш за все, ті блокові шифри, які мають ідентичні (або дуже схожі) раундові функції та використовують циклічні або повторювані

раундові ключі. Це означає, що функція, яка виконується в кожному раунді шифрування, має ту саму структуру і поводить себе однаково, незалежно від номеру раунду. Якщо ключ для кожного раунду є тим самим або повторюється з певним періодом, це створює умови для пошуку зсунутих пар.

Класичними прикладами вразливих шифрів є ті, які були спроектовані без усвідомлення загрози атаки зсуву. Багато ранніх блокових шифрів та деякі схеми, що використовують "чисту" ітерацію однієї й тієї ж функції, можуть бути вразливими. Це стосується як шифрів на основі мережі Фейстеля, так і шифрів на основі SPN, якщо їхні раундові функції не диференціюються між собою [28].

Конкретними прикладами вразливих шифрів можуть бути деякі теоретичні конструкції, а також спрощені версії реальних шифрів. Наприклад, якщо у DES (Data Encryption Standard) використовувати той самий раундовий ключ для всіх 16 раундів, він став би надзвичайно вразливим до атаки зсуву. Хоча DES використовує різні раундові ключі, які генеруються з майстер-ключа, існують модифікації або спрощені версії, де цей захист може бути відсутнім.

Для захисту від атак зсуву розробники шифрів застосовують різні стратегії. Найбільш ефективною є введення відмінностей між раундами, наприклад, за допомогою унікальних раундових констант або шляхом забезпечення того, що раундові ключі є унікальними та нециклічними, і генеруються складним чином з майстер-ключа. Це руйнує симетрію, необхідну для формування зсунутих пар, і, таким чином, унеможливлює проведення атаки зсуву [29]. Сучасні стандарти, такі як AES, активно використовують ці принципи, забезпечуючи унікальність кожного раунду та роблячи його стійким до цієї загрози.

2.2.2 Алгоритм проведення атаки зсуву

Атака зсуву є унікальним криптоаналітичним методом, що експлуатує ітеративну природу блокових шифрів. Її успіх великою мірою залежить насамперед від того, чи наявна симетрія у раундових функціях та ключовому розкладі шифру. Для розуміння механізму цієї атаки необхідно детально розглянути її алгоритм проведення.

Першим кроком у підготовці до атаки зсуву є аналіз архітектури цільового блокового шифру. Атака є ефективною проти шифрів, де всі раунди шифрування використовують ідентичну (або дуже схожу) раундову функцію, і генерація раундових ключів є циклічною або повторюваною. Якщо E_K позначає повне шифрування з ключем K, а E_k позначає один раунд шифрування з раундовим ключем k, то для вразливого шифру $E_K(P) = E_{k_N}(\dots E_{k_2}(E_{k_1}(P))\dots)$, де k_i є раундовими ключами, а функція E_k є ідентичною для всіх раундів.

Центральним елементом атаки зсуву є поняття "зсунутої пари". Зсунута пара складається з двох пар відкритий-закритий текст (P_0,C_0) та (P_1,C_1), таких, що $P_1 = E_K(P_0)$ і $C_1 = E_K(C_0)$, де E_K – це функція одного раунду шифрування з раундовим ключем k. Тобто, P_1 є результатом шифрування P_0 одним раундом, а C_1 – результатом шифрування C_0 також одним раундом. Завдяки ітеративній природі шифру, якщо $C_0 = E_K(P_0)$ і $C_1 = E_K(P_1)$, то P_1 — це відкритий текст, який, будучи зашифрованим повним шифром E_K, дасть C_1, але при цьому $E_K(P_1)$ також є $E_K(E_k(P_0))$.

Наступним кроком для проведення атаки зсуву буде пошук зсунутих пар. Для пошуку зсунутих пар криптоаналітику необхідний доступ до великої кількості пар відкритий-закритий текст. В більшості випадків це передбачає сценарій атаки з відомим відкритим текстом або з вибраним відкритим текстом, де можна отримати шифротексти для самостійно обраних відкритих текстів. Зібрані пари (P_i,C_i) формують вихідну множину даних.

Основна фаза атаки полягає у переборі всіх можливих пар з зібраної множини (P_i,C_i) та (P_j,C_j) і перевірці умови "зсунутості". Для кожної такої пари виконується гіпотетичне шифрування P_i одним раундом з деяким можливим раундовим ключем k (або частиною ключа), щоб перевірити, чи дорівнює результат P_j. Аналогічно, розшифровується C_j одним раундом, щоб перевірити, чи дорівнює результат C_i. Якщо ці умови виконуються для однієї й тієї ж раундової функції E_k, то знайдено зсунуту пару.

На наступному етапі алгоритму, коли зсунута пара (P_0,C_0) та (P_1,C_1) ідентифікована, це означає, що P_1 є результатом шифрування P_0 одним раундом, а C_1 є результатом шифрування C_0 одним раундом, використовуючи той самий

раундовий ключ k. Тобто, P′=Ek(P) та C′=Ek(C). Це дозволяє криптоаналітику обмежетися "атакою" на лише один раунд шифрування, що значно простіше, ніж атакувати повний шифр. Виходячи з відомих P_0, C_0, P_1, C_1, та знаючи структуру одного раунду, можна обчислити значення k.

Після успішного відновлення раундового ключа k з однієї зсунутої пари, його можна використати для виведення повного майстер-ключа K, якщо відома схема генерації раундових ключів. Це перетворює атаку зсуву з перебору повної довжини ключа на перебір значно меншої довжини раундового ключа, що суттєво зменшує обчислювальну складність.

Складність атаки зсуву оцінюється кількістю необхідних пар відкритого тексту та обчислювальними ресурсами для пошуку зсунутих пар. Оскільки атака не залежить від кількості раундів, її складність залишається відносно постійною, тоді як складність атаки грубої сили зростає експоненційно з кожним додатковим раундом. Це робить атаку зсуву особливо ефективною проти шифрів, які спроектовані з великою кількістю раундів для протидії іншим атакам, але мають у собі симетрію, що експлуатується атакою зсуву.

Для захисту від атаки зсуву розробники шифрів застосовують різні контрзаходи. Найбільш ефективним є введення "несиметричності" або "відмінностей" між раундами шифрування. Це може бути досягнуто шляхом використання унікальних раундових констант, які додаються в кожному раунді, або шляхом забезпечення того, що раундові ключі генеруються таким чином, що вони не повторюються та не створюють "зсунутих" відношень. Це руйнує властивість, необхідну для формування зсунутих пар, і, таким чином, унеможливлює проведення атаки зсуву.

Тому, наприклад, такі сучасні стандарти шифрування як AES (Advanced Encryption Standard), розроблені таким чином, щоб бути стійкими до атаки зсуву. Ця стійкість досягається завдяки свідомому впровадженню певних архітектурних рішень, які руйнують симетрію, що експлуатується атакою зсуву. Головний принцип захисту AES полягає у тому, що його раундові функції не є ідентичними у чистому вигляді, і, що ще важливіше, раундові ключі не повторюються і не утворюють циклічних залежностей.

Зокрема, AES ефективно протистоїть атаці зсуву завдяки використанню унікальних раундових ключів, які генеруються з майстер-ключа за допомогою складного ключового розкладу. Кожен раунд шифрування AES використовує свій власний, відмінний від інших раундів, підключ, який не буде циклічним і не буде мати жодної прямої залежності від інших раундових ключів. Це означає, що функція $E_k(P)$ (шифрування одного раунду) завжди використовує різний ключ k для кожного раунду. Такий підхід повністю руйнує передумову для атаки зсуву, а саме – ідентичність операції E_k для виявлення зсунутих пар. Навіть якщо б основні перетворення (SubBytes, ShiftRows, MixColumns) були однаковими, додавання унікального раундового ключа на кожному кроці робить кожен раунд унікальним і унеможливлює знаходження зсунутих пар, які необхідні для проведення атаки.

РОЗДІЛ 3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ АТАКИ ЗСУВУ

3.1 Обгрунтування вибору середовища розробки програмного забезпечення

Для того, щоб програмно реалізувати алгоритм атаки зсуву на блокові шифри, що є ключовою частиною практичного дослідження, необхідно було обрати правильне середовище розробки та мову програмування. Кінцевий вибір ґрунтувався на критеріях ефективності, гнучкості, доступності та зручності для розробки криптоаналітичних інструментів. Враховуючи ці фактори, було обрано мову програмування Python та інтегроване середовище розробки JetBrains PyCharm Community Edition.

Вибір Python як мови програмування обумовлений його широкими можливостями та високою гнучкістю, що є критично важливим для наукових досліджень та прототипування. Python відзначається читабельним синтаксисом, що значно прискорює процес розробки та налагодження коду, а також полегшує його розуміння іншими дослідниками. Крім того, наявність великої кількості вбудованих бібліотек та модулів для роботи з бітовими операціями, даними та математичними функціями дозволяє ефективно реалізовувати криптографічні примітиви та складні алгоритми атаки.

JetBrains PyCharm Community Edition було обрано як середовище розробки через його потужні функціональні можливості та орієнтованість саме на Python. PyCharm пропонує широкий спектр інструментів, що значно покращують продуктивність розробника. Серед них – інтелектуальне автодоповнення коду, розширені можливості рефакторингу, ефективний відладчик, що дозволяє покроково відстежувати виконання програми та аналізувати стан змінних, а також зручні інструменти для управління проектами та віртуальними середовищами (див. рис. 3.1).

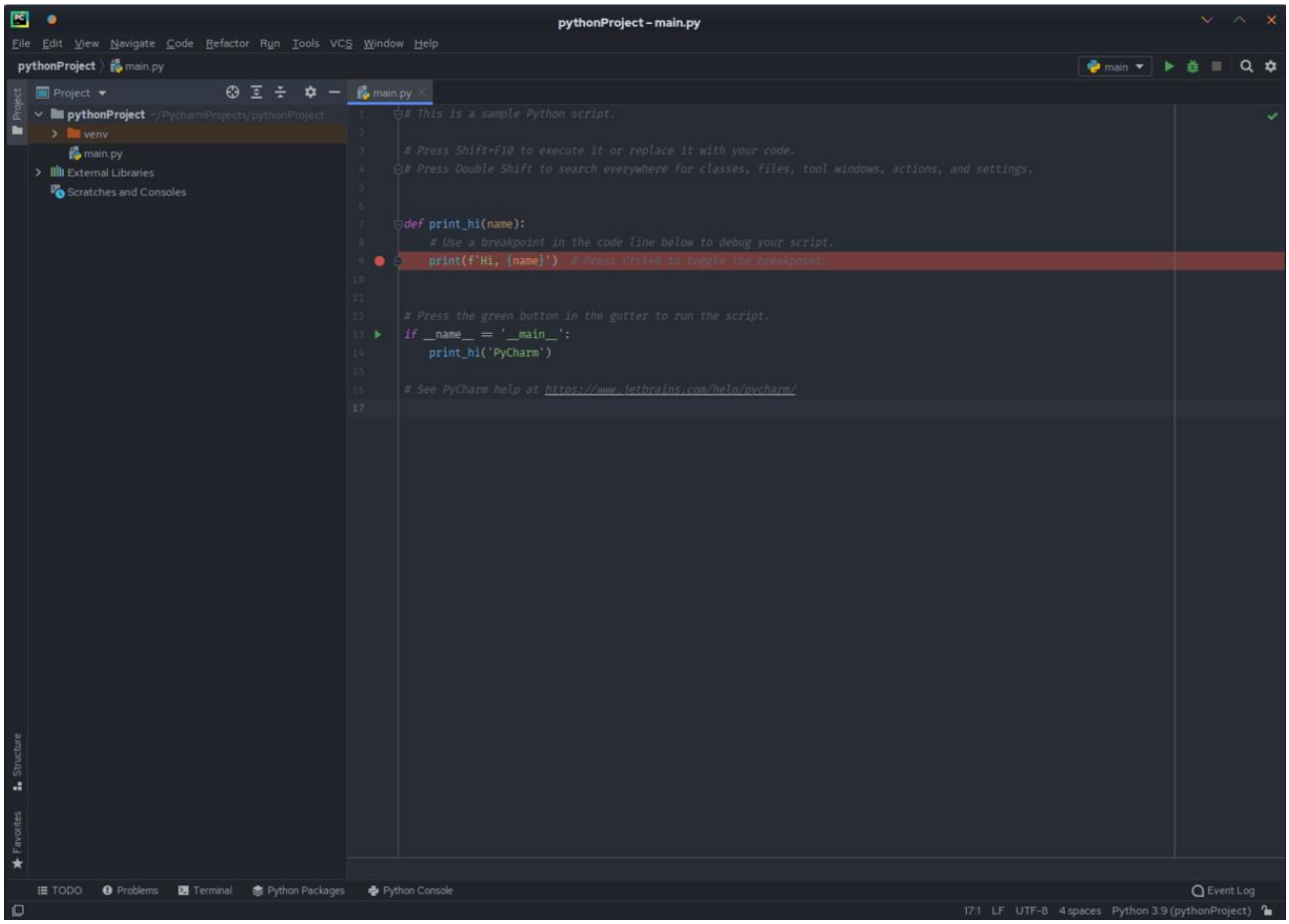

Рисунок 3.1 – Вікно програми JetBrains PyCharm

Особливо цінними для даної роботи є можливості налагодження (debugging) у PyCharm. Реалізація криптоаналітичних атак часто включає роботу з великими обсягами даних та складними логічними умовами, де найдрібніша помилка може призвести до некоректних результатів. Надійний відладчик PyCharm дозволяє ефективно виявляти та усувати такі помилки, забезпечуючи коректність і точність програмного забезпечення для атаки зсуву.

На додаток до цього, PyCharm Community Edition є вільно розповсюджуваною версією, що робить його доступним для широкого кола користувачів та дослідників без додаткових фінансових витрат. Це забезпечує можливість відтворення експериментів та верифікації отриманих результатів, що є важливим аспектом наукової роботи. Поєднання простоти та потужності Python з розширеними можливостями PyCharm створює ідеальне середовище для реалізації та дослідження складності криптоаналітичних атак.

3.2 Методика проведення експерименту

Методика проведення експерименту розроблена для емпіричного дослідження результативності атаки зсуву на блокові шифри та її порівняння з ефективністю атаки грубої сили. Метою є отримання кількісних даних, які дозволять об'єктивно оцінити переваги та недоліки кожного методу в залежності від ключових параметрів шифру, зокрема кількості раундів шифрування, які використовуються в блоковому шифрі.

Для досягнення цієї мети буде використано модель блокового шифру, що імітує ітераційну структуру з ідентичними раундовими функціями і циклічними ключами, які повторюються кожні два раунди, що в свою чергу робить його вразливим до атаки зсуву. Загальна схема алгоритму зображена на рисунку 3.2.

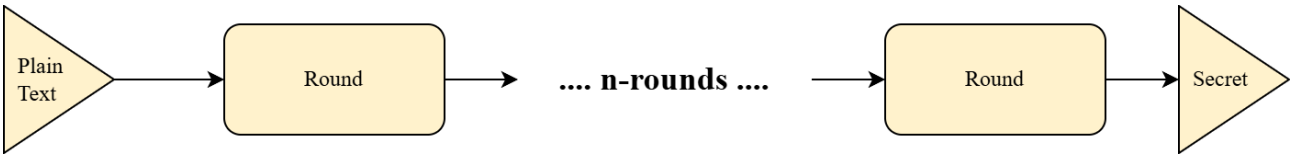

Рисунок 3.2 – Алгоритм ескпериментального шифру

Щоб вивчити вплив "глибини" шифрування на ефективність атак, буде використано 11 варіацій цього шифру, що відрізнятимуться лише кількістю повторюваних раундів операцій: 5, 10, 20, 40, 50, 60, 80, 100, 125, 150 та 200 раундів відповідно. Такий діапазон раундів дозволить спостерігати поведінку атак як на "коротких", так і на "глибоких" версіях шифру. Кожен раунд буде складати лише з операцій додавання ключа за модулем 2 і таблиці підстановки (див. рисунок 3.3).

Для кожної з варіацій шифру буде проведено 100 незалежних операцій шифрування, використовуючи різні випадково згенеровані секретні ключі. Це дозволить уникнути впливу специфіки одного ключа на загальну результативність атаки та забезпечити репрезентативність отриманих даних. Для кожного з цих ключів будуть згенеровані необхідні пари "відкритий текст - шифротекст", які слугуватимуть вхідними даними для подальшого проведення атак.

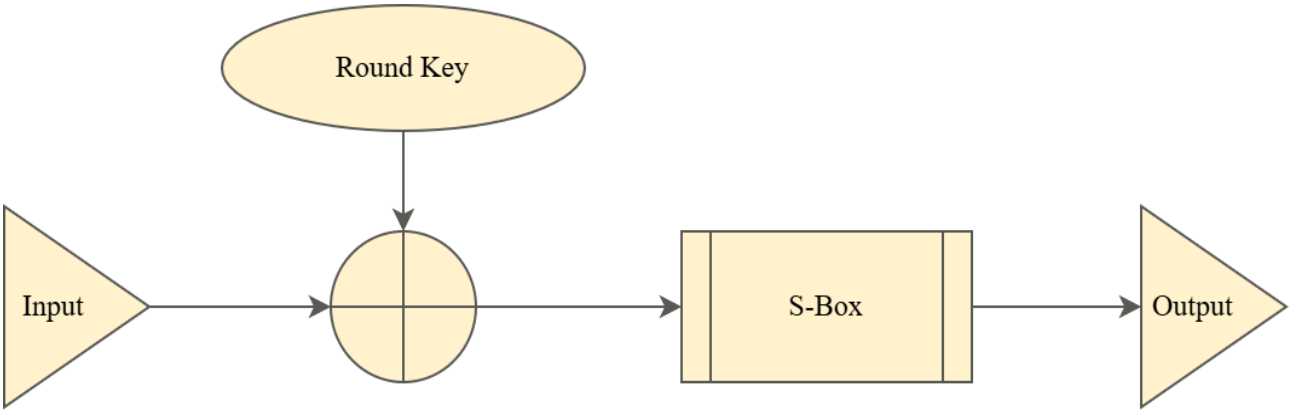

Рисунок 3.3 – Алгоритм одного раунду експериментального шифру

Наступним кроком для кожного з ключів, що використовуються в поточній варіації шифру, буде проведено атаку зсуву. Під час виконання атаки будуть фіксуватися дві ключові метрики: час, необхідний для успішного відновлення ключа, та кількість затребуваних (перевірених) пар "відкритого-закритого тексту", які були використані алгоритмом атаки зсуву для виявлення власне зсунутих пар. Ці параметри дозволять оцінити обчислювальну складність та ресурсну ефективність атаки.

Аналогічно, для кожного з тих самих ключів на кожній з одинадцяти варіацій шифру буде проведено злам грубою силою. Для цієї атаки основною метрикою буде час, витрачений на перебір всіх можливих ключів до моменту знаходження правильного. Час буде вимірюватися з такою ж точністю, що і час проведення атаки зсуву, щоб коректно відобразити експоненційну залежність складності атаки грубої сили від довжини ключа та кількості раундів. Визначення результативності атаки зсуву на блоковий шифр буде проведено за рахунок порівняння її результатів з результатами атаки грубої сили.

Таким чином, у результаті експерименту буде зібрано дані для 1100 окремих операцій зламу (11 версій шифру × 100 різних ключів). Кожна така операція включатиме час зламу атакою зсуву та кількість випробуваних пар, а також час зламу атакою грубої сили. Отримані дані будуть систематизовані та представлені у вигляді таблиць та графіків для наочного порівняння ефективності обох типів атак залежно від кількості раундів шифрування.

3.3 Програмна реалізація експерименту

Програмна реалізація алгоритму атаки зсуву є центральною частиною практичного дослідження, що дозволяє емпірично перевірити теоретичні припущення та оцінити ефективність атаки. Розроблена програма написана на мові Python і призначена для симуляції процесу шифрування блокового шифру, який відповідає умовам вразливості до атаки зсуву, а також для безпосереднього проведення криптоаналітичної атаки.

Основу програмної реалізації становить модель блокового шифру, що використовує ітеративну структуру з ідентичними раундовими функціями. Для демонстрації атаки було імплементовано спрощений блоковий шифр з параметрами, що дозволяють наочно продемонструвати концепцію "зсунутих пар". Зокрема, реалізовано функції для окремого раунду шифрування (з додаванням раундового ключа, застосуванням S-блоку), які продеменстровані в лістингу 3.1. Функція для повного багато раундового шифрування в свою чергу продемонстрована в лістингу 3.2. Ці функції слугують базою для генерації тестових даних та моделювання криптографічних операцій.

Лістинг 3.1 – Функції шифрування окремих раундів

```
def s_box(input_s):
    return __s_box[input_s]
def reverse_s_box(output_s):
    return __s_box.index(output_s)
def round_enc(input_r, key_r):
    return s_box(input_r ^ key_r)
```

Лістинг 3.2 – Функція повного багатораундового шифрування

```
def encrypt(input_p, key1, key2, rounds_count):
    result = input_p
    for round_N in range(rounds_count):
        round_key = key1 if round_N % 2 == 1 else key2
        result = round_enc(result, round_key)
    return result
```

Центральний модуль програми присвячений реалізації самого алгоритму атаки зсуву. Він включає функції для генерації достатньої кількості пар "відкритий текст - шифротекст". Ці пари генеруються шляхом шифрування випадкових відкритих текстів за допомогою повного, відомого нам, шифру. Далі, основний цикл алгоритму перебирає згенеровані пари, намагаючись знайти серед них "зсунуті пари". Для цього, для кожної потенційної пари (Pi,Ci) та (Pj ,Cj), виконується спроба перевірити умову Pj=Ek(Pi) та Cj=Ek(Ci), де Ek – це функція одного раунду шифрування (див. лістинг 3.4).

Лістинг 3.4 – Функція проведення атаки зсуву

```
    def slide_pair(key1, key2, rounds_count):
        plain1_pool, plain2_pool = list(range(0, 2**__bits)), list(range(0, 2**__bits))
        random.shuffle(plain1_pool)
        random.shuffle(plain2_pool)
        pairs_count = 0
        for plain1 in plain1_pool:
            secret1 = encrypt(plain1, key1, key2, rounds_count)
            for plain2 in plain2_pool:
                pairs_count += 1
                poss_keys, secret2 = [], encrypt(plain2, key1, key2, rounds_count)
                exp_key1_pool, exp_key2_pool = list(range(0, 2**__bits)), list(range(0, 2**__bits))
                for exp_key1 in exp_key1_pool:
                    for exp_key2 in exp_key2_pool:
                        if round_enc(round_enc(secret1, exp_key1), exp_key2) == secret2:
                            poss_keys.append([exp_key1, exp_key2])
                for exp_key1, exp_key2 in poss_keys:
                    if round_enc(round_enc(plain1, exp_key1), exp_key2) == plain2:
                        if exp_key1 == key1 and exp_key2 == key2:
                            return pairs_count
        return pairs_count
```

Для ідентифікації зсунутих пар програма використовує механізм, який перевіряє гіпотетичні раундові ключі та застосовує їх до відкритих текстів та шифротекстів з зібраної множини. Цей процес включає "частковий" або, в гіршому випадку, повний перебір для вузького пошуку потенційних зсунутих пар. Після виявлення такої пари, алгоритм переходить до етапу відновлення ключа. Цей етап передбачає використання властивостей зсунутої пари для ізоляції та визначення бітів раундового ключа, а потім – і повного майстер-ключа, якщо це дозволяє схема його генерації. Лістинг 3.5 демонструє реалізацію функції, яка використовується для проведення атаки грубої сили.

Лістинг 3.5 – Функція атаки зламу грубою силою

```
def brute_force(plain, secret, round_number):
    expec_key1_pool = list(range(0, 2 ** __bits))
    expec_key2_pool = list(range(0, 2 ** __bits))
    random.shuffle(expec_key1_pool)
    random.shuffle(expec_key2_pool)
    for exp_key1 in expec_key1_pool:
        for exp_key2 in expec_key2_pool:
            expec_secret = encrypt(plain, exp_key1, exp_key2,
round_number)
            if expec_secret == secret:
                return
```

Окрім реалізації самої атаки, програма включає функціонал для збору та аналізу статистичних даних, що характеризують її ефективність та загального проведення самого експерименту. Це дозволяє вимірювати час, необхідний для успішного проведення атаки, кількість необхідних пар відкритого тексту, а також показник успішності відновлення ключа, безпосередньо під час проведення атак зсуву та зламу грубою силою (див. лістинг 3.6). Отримані дані використовуються для порівняльного аналізу ефективності атаки зсуву з атакою грубої сили, що є ключовим аспектом експериментального дослідження.

Лістинг 3.6 – Програмна реалізація методики експерименту

```
    slide_time, brute_time = 0, 0
    pairs_count_total, pairs_count_numb = 0, 0
    round_number_pool = [5, 50, 100, 200]
    for round_number in round_number_pool:
        for i in range(100):
            key1, key2 = random.randint(0, (2**__bits) - 1), 
random.randint(0, (2**__bits) - 1)
            slide_start_time = time.time()
            pairs_count = slide_pair(key1, key2, round_number)
            pairs_count_total += pairs_count
            pairs_count_numb += 1
            slide_time += time.time() - slide_start_time

            plain = random.randint(0, (2**__bits) - 1)
            secret = encrypt(plain, key1, key2, round_number)
            start_time = time.time()
            brute_force(plain, secret, round_number)
            brute_time += time.time() - start_time
        print(f'{round_number} - {slide_time} - {pairs_count_total
/ pairs_count_numb} - {brute_time}')
```

Повний лістинг коду програми наведено в додатку А. На рисунку 3.4 зображено скриншот виконання програми.

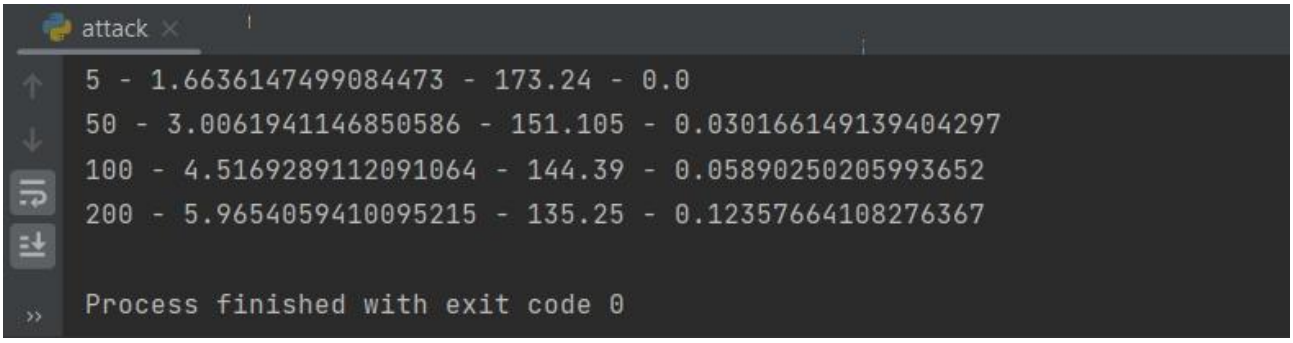

Рисунок 3.4 – Скриншот виконання програми

3.4 Результати дослідження

Проведені експерименти з реалізації атаки зсуву та порівняння її ефективності з атакою грубої сили на моделі блокового шифру з різною кількістю раундів надали чіткі та показові результати. Зібрані дані за 1100 операцій зламу (по 100 ключів для кожної з одинадцяти варіацій шифру: 5, 10, 20, 40, 50, 60, 80, 100, 125, 150 та 200 раундів) дозволили емпірично ілюструвати значення параметрів поведінки обох типів атак (див. таблицю 3.1).

Таблиця 3.1 – Результати проведеного експерименту

Кількість раундів	**Час зламу атакою зсуву, с**	**Середня кількість перевірених пар**	**Час зламу атакою грубої сили, с**
5	0.083	9.2	0.033
10	0.171	8.9	0.090
20	0.239	8.7	0.211
40	0.311	8.6	0.431
50	0.404	8.6	0.681
60	0.488	8.6	0.982
80	0.560	8.6	1.397
100	0.662	8.6	1.892
125	0.770	8.7	2.516
150	0.879	8.7	3.249
200	0.993	8.7	4.235

Аналіз даних, отриманих для атаки зсуву, демонструє зростаючу ефективність атаки зі збільшенням кількості раундів шифрування. Кількість затребуваних пар "відкритого-закритого тексту" виявилася відносно сталою при переході від 5 до 200 раундів (див. рисунок 3.5). З цього можна зробити висновок про цікаву особливість атаки зсуву: знаходження зсунутих пар в середньому потребує можливості перебору лише 12% множини усіх можливих пар відкритого тексту загалом. Оскільки збільшення кількості раундів не збільшує

простір пошуку для зсунутої пари, а лише збільшує кількість ітерацій, які потрібно пройти шифротексту. Також очевидно, що хоча її ефективність не залежить напряму від "глибини" шифрування, вона все ж демонструє покращення ефективності відносно зламу грубою силою при збільшенні кількості раундів.

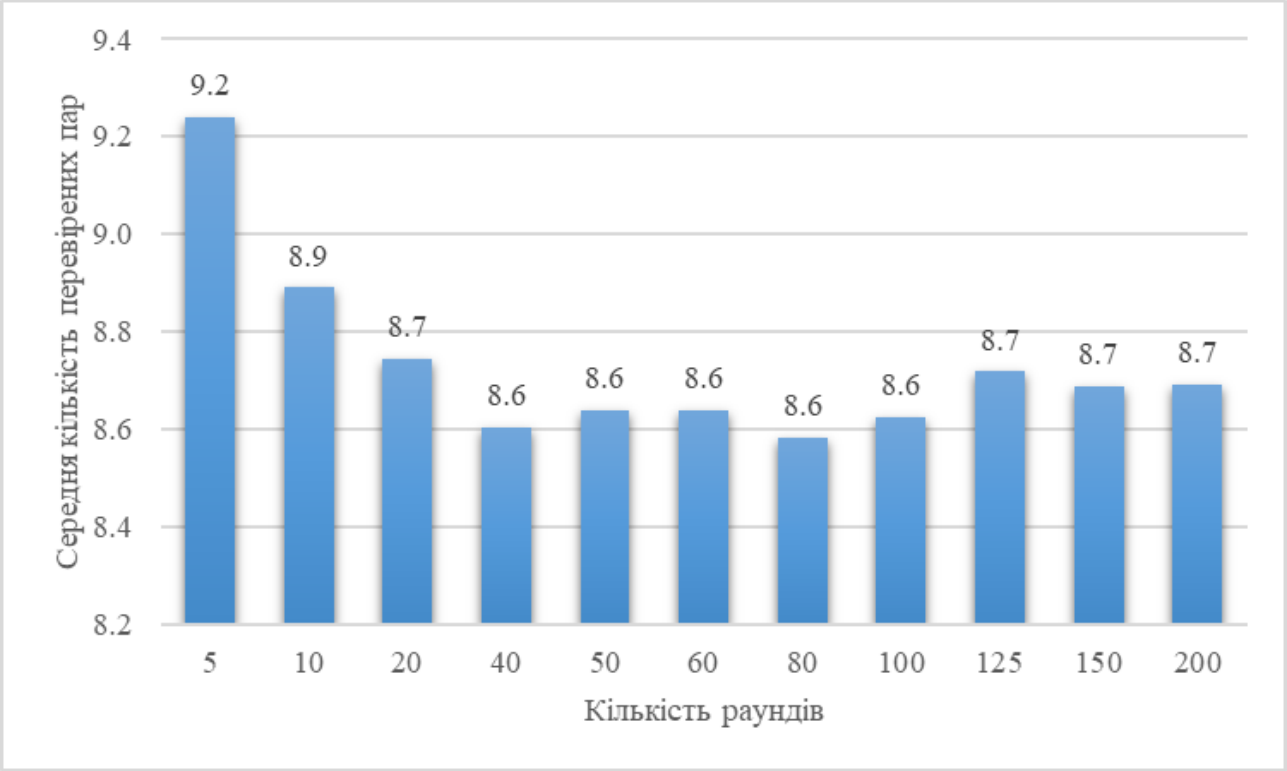

Рисунок 3.5 – Діаграма залежності кількості перевірених пар від кількості раундів шифру

На противагу цьому, результати для атаки грубої сили показали очікуване і значне погіршення ефективності з кожним збільшенням кількості раундів. Час, витрачений на злам, зростає пропорційно до кількості раундів шифрування, що відповідає теоретичним прогнозам. Для варіацій шифру від 5 до 20 раундів злам грубою силою ще був відносно швидким, але вже для 40, 50 і більше раундів час зламу став неприпустимо більшим за час зламу атакою зсуву, що підкреслює непрактичність цього підходу для шифрів з адекватною кількістю ітерацій. Цю тенденцію можна чітко побачити на рисунку 3.6, який ілюструє подані дані в вигляді діаграми.

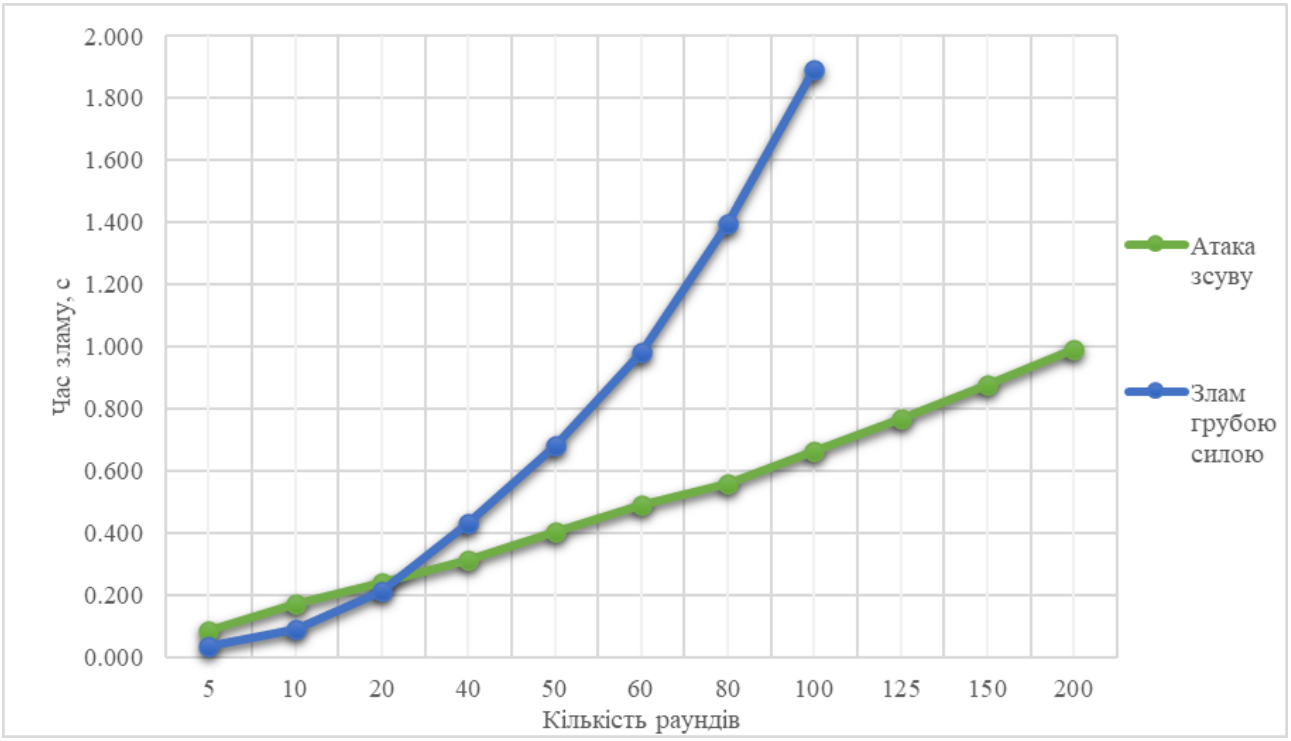

Рисунок 3.6 – Графік залежності часу зламу від кількості раундів

Порівняння обох методів виявило чітку тенденцію: для шифрів з невеликою кількістю раундів (наприклад, 10 або 20) атака грубої сили може бути порівнянною або навіть швидшою, якщо розмір ключа дозволяє це. Однак, при значному збільшенні кількості раундів (50, 100, 150), перевага атаки зсуву стає очевидною. Вона вимагає незрівнянно менше часу та обчислювальних ресурсів для відновлення ключа, оскільки її складність здебільшого не залежить від кількості ітерацій шифр — що є фатальною вразливістю для шифрів з ідентичними раундовими функціями — а від кількості ітерацій пошуку зсунутих пар.

Таким чином, експериментально отримані дані переконливо доводять, що атака зсуву стає більш ефективною при зростанні кількості повторюваних зламуваних операцій (раундів) у шифрі, оскільки її обчислювальна складність залишається відносно сталою. У той самий час, злам грубою силою стає катастрофічно менш ефективним зі збільшенням кількості раундів, оскільки його складність безпосередньо залежить від простору ключів та кількості ітерацій. Ці висновки підкреслюють критичну важливість уникнення симетрії раундових

функцій та забезпечення унікальності раундових ключів при проектуванні сучасних блокових шифрів для забезпечення їхньої стійкості до атаки зсуву.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вимоги ергономіки до організації робочого місця оператора ПК

Проведення дослідження результативності атаки зсуву на блокові шифри, що включає як теоретичний аналіз, так і інтенсивне програмне моделювання, вимагає значного часу, проведеного за персональним комп'ютером. Тому критично важливо забезпечити належну ергономічну організацію робочого місця оператора ПК. Тривала робота за комп'ютером без дотримання відповідних вимог може призвести до негативних наслідків для здоров'я, включаючи проблеми із зором, опорно-руховим апаратом та загальну втому.

Ергономічно оптимізоване робоче місце є запорукою не лише збереження здоров'я та комфорту дослідника, а й безпосередньо впливає на його продуктивність та ефективність. Правильне розташування монітора, клавіатури, миші, а також належний вибір столу та крісла, мінімізує фізичне навантаження, дозволяючи зосередитися на складних задачах криптоаналізу та програмування. Таким чином, увага до ергономіки є невід'ємною частиною забезпечення успішного та безпечного проведення наукових досліджень у сфері кібербезпеки.

Для забезпечення безпеки та комфорту праці, комп'ютеризовані робочі місця обов'язково повинні відповідати вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце під час виконання робіт сидячи» [30].

Згідно з ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце під час виконання робіт сидячи», конструкція робочого місця оператора комп'ютера та розташування всіх його елементів — таких як сидіння, органи керування (клавіатура, миша) та засоби відображення інформації (монітор) — повинні відповідати низці вимог. Ці вимоги охоплюють антропометричні, фізіологічні та психологічні аспекти, а також враховують специфіку виконуваної роботи. Метою є забезпечення оптимальних умов для людини, мінімізація фізичного навантаження та підвищення ефективності праці.

Особлива увага приділяється забезпеченню досяжності моторного поля. Конструкція робочого місця має бути такою, щоб усі трудові операції могли виконуватися без надмірних рухів або незручних поз [31]. Це означає, що клавіатура, миша та інші часто використовувані пристрої повинні знаходитись у зоні легкого доступу.

Висота сидіння та підставки для ніг (якщо висота робочої поверхні не регулюється) повинна бути встановлена відповідно до зросту користувача. Для стандартної висоти робочої поверхні (як для зросту 1800 мм) це значення є орієнтовним, а для людей меншого зросту необхідно збільшувати висоту сидіння та підставки для ніг на різницю між оптимальною висотою робочої поверхні для їхнього зросту та 1800 мм [30].

Щодо елементів робочого столу, мінімальна товщина стільниці залежить від міцності матеріалу та технічних вимог, проте вона не повинна перевищувати 30 мм. Це забезпечує оптимальну ергономіку та естетику. Важливим елементом є також підставка для ніг, яка повинна бути регульованою за висотою та мати ширину не менше 300 мм і довжину не менше 400 мм. Її поверхня має бути рифленою, щоб запобігти ковзанню ніг, а по передньому краю необхідно передбачити бортик висотою 10 мм для додаткової фіксації стоп [30].

Загалом, положення ДСТУ 8604:2015 формують комплексну основу для створення ергономічних робочих місць. Дотримання цього стандарту є запорукою не лише відповідності встановленим вимогам, а й забезпечення здоров'я, комфорту та високої продуктивності операторів ПК під час тривалої роботи [31].

Комплексне застосування цього стандарту дозволяє створити комфортні та безпечні умови праці, що безпосередньо впливає на збереження здоров'я працівників, підвищення їхньої працездатності та ефективності виконання завдань, зокрема під час тривалої та інтенсивної роботи над такими науковими дослідженнями, як криптоаналіз. Це забезпечує мінімізацію ризиків професійних захворювань та сприяє створенню сприятливого виробничого середовища.

4.2 Поведінкові реакції населення у надзвичайних ситуаціях

Надзвичайна ситуація (НС) є комплексом несприятливих обставин, що виникають внаслідок аварії, катастрофи, стихійного лиха або інших небезпечних подій. Ці ситуації характеризуються значним порушенням нормальних умов життєдіяльності населення, потенційною або реальною загрозою для життя та здоров'я людей, значними матеріальними збитками, а також неможливістю ліквідації їхніх наслідків власними силами. До таких подій можуть призводити як природні чинники, так і діяльність людини, створюючи широкий спектр викликів для суспільства та держави.

Поняття надзвичайної ситуації та основи цивільного захисту від них в Україні врегульовуються низкою нормативно-правових документів. Ключовим серед них є Кодекс цивільного захисту України, що визначає правові та організаційні засади у сфері цивільного захисту, повноваження органів державної влади, органів місцевого самоврядування, права та обов'язки громадян, а також суб'єктів господарювання [32]. Окрім цього, важливу роль у регулюванні реагування на масштабні кризи відіграє Закон України "Про правовий режим надзвичайного стану", який визначає умови та порядок введення особливого правового режиму в країні чи окремих її місцевостях у випадку виникнення надзвичайних ситуацій техногенного або природного характеру не нижче загальнодержавного рівня [33]. Додатково, численні постанови Кабінету Міністрів України, накази профільних міністерств та відомств деталізують процедури реагування, класифікацію НС та порядок взаємодії служб.

Окрім зазначених законодавчих актів, важливим елементом системи цивільного захисту є чітка класифікація надзвичайних ситуацій, що дозволяє уніфікувати підходи до їх оцінки та реагування. Цей порядок регламентується постановою Кабінету Міністрів України, яка визначає критерії та механізм класифікації надзвичайних ситуацій за їх рівнями [34]. Така класифікація (наприклад, державний, регіональний, місцевий, об'єктовий рівні) є ключовою для визначення масштабу загрози, розподілу повноважень між органами влади,

а також мобілізації необхідних ресурсів та сил для ефективної ліквідації наслідків НС.

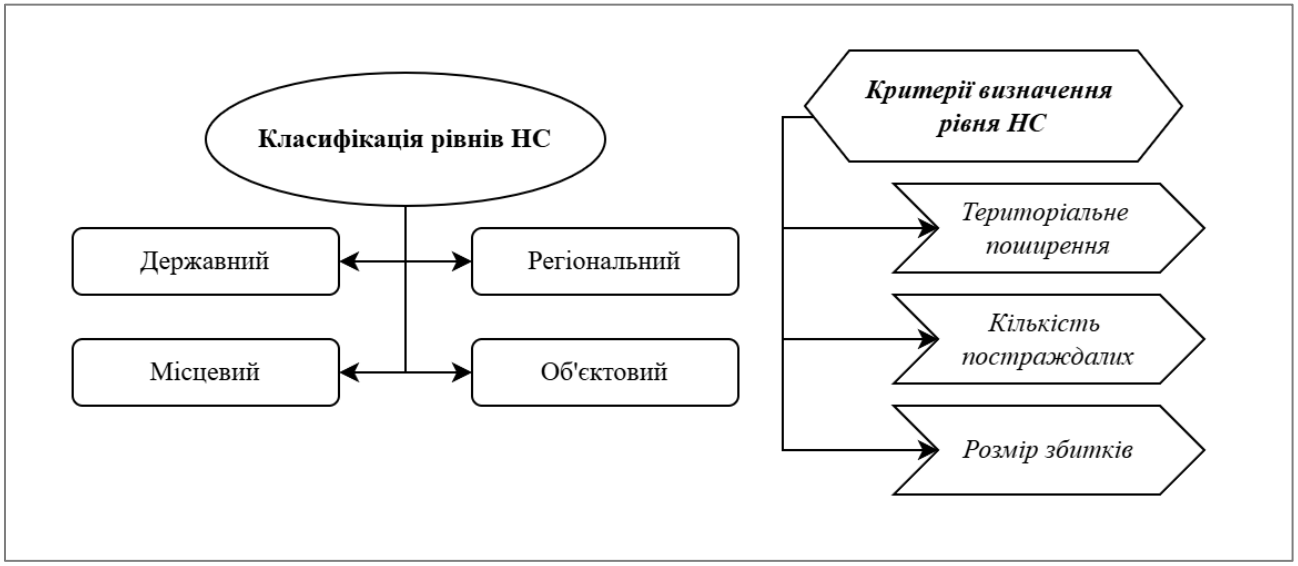

Рисунок 4.1 – Класифікація рівнів надзвичайних ситуацій та критерії їх визначення

Важливою ознакою надзвичайної ситуації є необхідність залучення значних матеріальних та фінансових ресурсів, а також спеціальних сил та засобів для її ліквідації. Це може включати діяльність державних органів, рятувальних служб, військових підрозділів та волонтерських організацій. Метою цих заходів є мінімізація втрат, порятунок людей, локалізація небезпеки та якнайшвидше відновлення нормального функціонування постраждалих територій [35].

Кожна людина у повсякденному житті, на роботі або під час відпочинку на природі може зіткнутися з надзвичайними ситуаціями. Ці ситуації характеризуються несподіваністю, значною тривалістю та інтенсивністю впливу несприятливих зовнішніх факторів, а іноді й прямою загрозою для життя та здоров'я [35]. Вони виходять за межі звичайних обставин і вимагають особливої реакції та підготовки.

Особи, які опиняються в зоні катастроф чи стихійних лих, тобто в екстремальних умовах, піддаються значним психотравматичним впливам. Загроза життю та безпеці викликає сильні емоційні перевантаження, що з часом

може призвести до розвитку різноманітних психосоматичних розладів та захворювань.

Людина потрапляє в екстремальні ситуації з різних причин, але часто це не є результатом прямої дії непереборних сил природи. Нерідко це відбувається через відсутність досвіду безпечної поведінки у природному чи соціальному середовищі, ігнорування норм і правил безпеки, непередбачливість або легковажність. Деякі люди не здатні адекватно діяти у надзвичайній ситуації через ілюзію власної безпеки або невірну, передчасну поведінку.

Щоб мінімізувати ризик потрапляння в екстремальні ситуації та підвищити шанси на збереження здоров'я і життя, необхідно володіти інформацією про фактори ризику, що супроводжують наше повсякденне існування. Запобіжні заходи повинні бути спрямовані на своєчасне попередження небезпечних для життя ситуацій та уникнення їх. Крім того, важливо вміти швидко оцінювати ситуацію, визначати своє місцезнаходження, приймати виважені рішення та діяти відповідно [36].

В екстремальних умовах у людини домінує особливе емоційне напруження, або стрес. Це комплекс фізіологічних захисних реакцій, що виникають в організмі у відповідь на дію несприятливих факторів. Вплив стресу на поведінку та можливості людини є надзвичайно індивідуальним [36]. У стані сильного емоційного напруження можуть проявлятися різкі зміни в поведінці, іноді аж до психологічного шоку, який супроводжується значною загальмованістю. Під впливом шоку рухи стають різкими, неточними, погано скоординованими, порушується пам'ять, людина може забувати послідовність дій, неправильно оцінювати ситуацію та допускати грубі помилки [37].

Будь-яка емоція активізує нервову систему, викликаючи появу в крові біологічно активних речовин, що впливають на функціонування різних органів, таких як кровообіг, дихання, травлення [36]. Зокрема, виділяються гормони надниркових залоз, як-от адреналін. Дофамін та інші медіатори імунної системи можуть бути пригнічені стресом, що знижує здатність організму захищатися від мікроорганізмів (вірусів, бактерій). Це пояснює, чому люди, які перебувають у

стані хронічного стресу, більш схильні до інфекційних та простудних захворювань.

Важливо розуміти, що стрес не слід боятися, а потрібно бути готовим до нього. Тренування стресостійкості здатне значно знизити нервові перевантаження та, відповідно, ймовірність несприятливих реакцій організму. Для цього необхідна постійна робота над собою, включаючи володіння своїми емоціями у повсякденному житті та контроль над реакціями, що можуть спричинити неадекватну поведінку. Хоча короткочасні стреси можуть мобілізувати сили організму, тривалий стрес, особливо при вичерпанні резервів адаптації, може призвести до розладів внутрішнього середовища, психосоматичних порушень, таких як стенокардія, інфаркт міокарда, інсульт, гіпертонія, виразкова хвороба і навіть онкологічні захворювання [37].

Тривала стресова дія призводить до поступового зниження активності організму та його працездатності, досягаючи максимального рівня лише після вичерпання резервів. Після вичерпання всіх адаптаційних резервів можуть виникати хронічні психічні розлади, включаючи фобії [37]. Отже, перебування в екстремальних умовах значно підвищує фактор ризику, що вимагає всебічного підходу до вирішення проблем, включаючи відповідальність за несподівані зміни та виконання завдань.

Комплексний аналіз поведінкових реакцій населення у надзвичайних ситуаціях, що охоплює як психофізіологічні механізми адаптації до стресу, так і наслідки довготривалого перебування у критичних умовах, є фундаментальним для розробки ефективних стратегій безпеки. Розуміння фаз працездатності та факторів, що впливають на них, дозволяє створювати програми підготовки та тренування, спрямовані на підвищення стресостійкості та формування адекватних поведінкових реакцій. Це, у свою чергу, є ключовим для мінімізації негативного впливу екстремальних ситуацій на здоров'я та життя людей, а також для забезпечення стабільного функціонування критичної інфраструктури.

ВИСНОВКИ

У рамках цієї кваліфікаційної роботи було всебічно досліджено результативність атаки зсуву на блокові шифри, що є актуальним завданням у сучасній криптографії. Проведено ґрунтовний теоретичний аналіз сутності блокових шифрів, їхніх основних архітектурних рішень та існуючих криптоаналітичних методів. Детально вивчено механізми атаки зсуву, включаючи математичні основи та умови її застосування, що дозволило поглибити розуміння її принципових відмінностей від інших атак.

Практична частина роботи полягала у програмній реалізації симуляції атаки зсуву та проведенні серії експериментів на обраному блоковому шифрі. Ці експерименти дозволили кількісно оцінити ефективність атаки зсуву за різної кількості раундів шифрування, фіксуючи такі ключові метрики, як час пошуку зсунутих пар, необхідна кількість відкритого тексту та успішність відновлення ключа. Отримані результати наочно демонструють, що ефективність атаки зсуву має тенденцію до зростання (або принаймні зберігає свою ефективність в більшості випадків) зі збільшенням кількості повторюваних операцій або раундів у шифрі. Це підтверджує її унікальну здатність експлуатувати ітераційну структуру алгоритму, незалежно від його глибини.

Ключовим результатом дослідження стало порівняння ефективності атаки зсуву з атакою грубої сили. Експериментально доведено, що злам грубою силою демонструє значне зниження ефективності зі збільшенням кількості раундів, оскільки для нього кожне додаткове повторення операцій означає збільшення розміру простору ключа. На противагу цьому, атака зсуву, завдяки використанню структурних закономірностей, виявилася менш чутливою до зростання кількості раундів, що робить її особливо небезпечною для шифрів, спроектованих без урахування її специфіки.

Таким чином, результати цієї роботи підтверджують, що атака зсуву становить серйозну загрозу для певних класів блокових шифрів, особливо тих, що мають симетричні та ітераційні раундові функції. Одержані висновки мають практичне значення для криптографічної спільноти, надаючи важливу

інформацію для оцінки безпеки існуючих шифрів та формування рекомендацій щодо розробки нових, більш стійких до атак такого типу. Це сприятиме підвищенню загального рівня кібербезпеки та захисту конфіденційної інформації в умовах постійного розвитку криптоаналітичних методів.

Отримані результати відкривають широкі можливості для подальших досліджень у галузі криптоаналізу та проектування блокових шифрів. Зокрема, перспективним напрямком є розширення експериментальної бази шляхом застосування атаки зсуву до відомих блокових шифрів, що мають ітераційну структуру, для перевірки її ефективності в різних архітектурах, які мають практичне застосування в сучасній криптографії. Крім того, цікаво було б дослідити модифікації атаки зсуву або її комбінації з іншими криптоаналітичними техніками (наприклад, диференціальним чи лінійним криптоаналізом) для виявлення потенційно більш потужних гібридних атак. Також важливим є розробка та впровадження контрзаходів, спрямованих на підвищення стійкості блокових шифрів саме до атаки зсуву, таких як застосування несиметричних раундових функцій або динамічної зміни ключів раунду, що могло б ускладнити формування "зсунутих пар". Ці напрямки дозволять глибше зрозуміти вразливості криптографічних алгоритмів та сприятимуть створенню надійніших систем захисту інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schneier, B. (1998). Cryptographic design vulnerabilities. Computer, 31(9), 29–33. https://doi.org/10.1109/2.708447

2. Chandra, S., Paira, S., Alam, S. S., & Sanyal, G. (2014). A comparative survey of symmetric and asymmetric key cryptography. У 2014 international conference on electronics,communication and computational engineering (ICECCE). IEEE. https://doi.org/10.1109/icecce.2014.7086640

3. Ahmadi, M., Kaur, J., Rani Nayak, D., Nutan, R., Taw, S., & Afaq, Y. (2024). A review of various symmetric encryption algorithms for multiple applications. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4491217

4. Nita, S. L., & Mihailescu, M. I. (2024). Symmetric encryption algorithms. У Cryptography and cryptanalysis in java (с. 165–182). Apress. https://doi.org/10.1007/979-8-8688-0441-0_10

5. Balli, S., & Yilmaz, M. (2020). Multi-criteria usability evaluation of symmetric data encryption algorithms in fuzzy environment. SN Applied Sciences, 2(8). https://doi.org/10.1007/s42452-020-3170-9

6. Ubaidullah, M., & Makki, Q. (2016). A review on symmetric key encryption techniques in cryptography. International Journal of Computer Applications, 147(10), 43–48. https://doi.org/10.5120/ijca2016911203

7. Бойко, М. І., & Boiko, M. (2018). Методи та засоби симетричного шифрування [Thesis Abstract]. ELARTU – Інституційний репозитарій ТНТУ імені Івана Пулюя. http://elartu.tntu.edu.ua/handle/lib/24088

8. Mandal, B. K., Bhattacharyya, D., & Bandyopadhyay, S. K. (2013). Designing and performance analysis of a proposed symmetric cryptography algorithm. У 2013 international conference on communication systems and network technologies (CSNT 2013). IEEE. https://doi.org/10.1109/csnt.2013.101

9. Maqsood, F., Ahmed, M., Mumtaz, M., & Ali, M. (2017). Cryptography: A comparative analysis for modern techniques. International Journal of Advanced Computer Science and Applications, 8(6). https://doi.org/10.14569/ijacsa.2017.080659

10. Yakymenko, I., Karpinski, M., Shevchuk, R., & Kasianchuk, M. (2024). Symmetric encryption algorithms in a polynomial residue number system. Journal of Applied Mathematics, 2024, 1–12. https://doi.org/10.1155/2024/4894415

11. Kuznetsov, O., Poluyanenko, N., Frontoni, E., Kandiy, S., Karpinski, M., & Shevchuk, R. (2024). Enhancing cryptographic primitives through dynamic cost function optimization in heuristic search. Electronics, 13(10), 1825. https://doi.org/10.3390/electronics13101825

12. Kharchenko, A., Halay, I., Zagorodna, N., & Bodnarchuk, I. (2015). Trade-off optimal decision of the problem of software system architecture choice. У 2015 xth international scientific and technical conference "computer sciences and information technologies" (CSIT). IEEE. https://doi.org/10.1109/stc-csit.2015.7325465

13. Kuznetsov, A., Karpinski, M., Ziubina, R., Kandiy, S., Frontoni, E., Peliukh, O., Veselska, O., & Kozak, R. (2023). Generation of nonlinear substitutions by simulated annealing algorithm. Information, 14(5), 259. https://doi.org/10.3390/info14050259

14. Dalmasso, L., Bruguier, F., Benoit, P., & Torres, L. (2019). Evaluation of SPN-Based Lightweight Crypto-Ciphers. IEEE Access, 7, 10559–10567. https://doi.org/10.1109/access.2018.2889790

15. Rivain, M., & Roche, T. (2013). SCARE of secret ciphers with SPN structures. У Advances in cryptology - ASIACRYPT 2013 (с. 526–544). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-42033-7_27

16. Zhang, W., Cao, M., Guo, J., & Pasalic, E. (2020). Improved security evaluation of SPN block ciphers and its applications in the single-key attack on SKINNY. IACR Transactions on Symmetric Cryptology, 171–191. https://doi.org/10.46586/tosc.v2019.i4.171-191

17. Albrecht, M. R., Grassi, L., Perrin, L., Ramacher, S., Rechberger, C., Rotaru, D., Roy, A., & Schofnegger, M. (2019). Feistel structures for MPC, and more. У Lecture notes in computer science (с. 151–171). Springer International Publishing. https://doi.org/10.1007/978-3-030-29962-0_8

18. JarJar, A. (2018). Improvement of Feistel method and the new encryption scheme. Optik, 157, 1319–1324. https://doi.org/10.1016/j.ijleo.2017.12.065

19. Pehlivanoglu, M. K., & Demir, M. A. (2022). A framework for global optimization of linear layers in SPN block ciphers. У 2022 15th international conference on information security and cryptography (ISCTURKEY). IEEE. https://doi.org/10.1109/iscturkey56345.2022.9931793

20. Srilaya, S., & Velampalli, S. (2018). Performance evaluation for DES and AES algorithms- an comprehensive overview. У 2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT). IEEE. https://doi.org/10.1109/rteict42901.2018.9012536

21. Curlin, P. S., Heiges, J., Chan, C., & Lehman, T. S. (2025). A survey of hardware-based AES sboxes: Area, performance, and security. ACM Computing Surveys. https://doi.org/10.1145/3724114

22. Karnaukhov, A., Tymoshchuk, V., & Orlovska, A. (2024). Use of authenticated aes-gcm encryption in vpn. У Актуальні питання розвитку галузей науки (D. Tymoshchuk, Chair). ТОВ УКРЛОГОС Груп. https://doi.org/10.62731/mcnd-14.06.2024.004

23. Kanda, M. (2001). Practical security evaluation against differential and linear cryptanalyses for feistel ciphers with SPN round function. У Selected areas in cryptography (с. 324–338). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44983-3_24

24. Lasry, G. [. (2018). A methodology for the cryptanalysis of classical ciphers with search metaheuristics / george lasry [Kassel : Kassel University Press]. http://d-nb.info/1153797542/34

25. Chatterjee, R., & Chakraborty, R. (2024). Statistical cryptanalysis of seven classical lightweight ciphers. International Journal of Information Technology. https://doi.org/10.1007/s41870-024-02175-4

26. Ярема, О., & Загородна, Н. (2025). Експериментальне дослідження впливу диференціальної рівномірності на стійкість S-блоків до різних типів криптоаналізу. Measuring and Computing Devices in Technological Processes, (2), 278–284. https://doi.org/10.31891/2219-9365-2025-82-39

27. Roman’kov, V. (2018). Two general schemes of algebraic cryptography. Groups Complexity Cryptology. https://doi.org/10.1515/gcc-2018-0009

28. Bar-On, A., Biham, E., Dunkelman, O., & Keller, N. (2017). Efficient slide attacks. Journal of Cryptology, 31(3), 641–670. https://doi.org/10.1007/s00145-017-9266-8

29. Zhang, K., Lai, X., Wang, L., Guan, J., & Hu, B. (2022). Slide attack on full-round ULC lightweight block cipher designed for iot. Security and Communication Networks, 2022, 1–8. https://doi.org/10.1155/2022/4291000

30. ДП «УкрНДНЦ». (б. д.). Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги (ДСТУ 8604:2015).

31. Грибан, В. Г., & Негодченко, О. В. (2009). Охорона праці. Підручник. Центр учбової літератури.

32. Кодекс цивільного захисту України, Кодекс України № 5403-VI (2025) (Україна). https://zakon.rada.gov.ua/laws/show/5403-17#Text

33. Про правовий режим надзвичайного стану, Закон України № 1550-III (2024) (Україна). https://zakon.rada.gov.ua/laws/show/1550-14#Text

34. Про затвердження Порядку класифікації надзвичайних ситуацій за їх рівнями, Постанова Кабінету Міністрів України № 368 (2021) (Україна). https://zakon.rada.gov.ua/laws/show/368-2004-п#Text

35. Скобло, Ю., Соколовська, Т., & Морозенко, Д. (2006). Безпека життєдіяльності. Навчальний посібник для вищих навчальних закладів 3-4 рівнів акредитації. Кондор.

36. Мохняк, С. М. (2009). Безпека життєдіяльності. Навчальний посібник. НУ "Львівська політехніка".

37. Толок, А. О., & Крюковська, О. А. (2011). Безпека життєдіяльності. Навчальний посібник. ДДТУ.

Додаток А Лістинг файлу slide_attack.py

```
import random
import time

__bits = 4
__s_box = list(range(0, 2**__bits))
random.shuffle(__s_box)


def s_box(input_s):
    return __s_box[input_s]


def reverse_s_box(output_s):
    return __s_box.index(output_s)


def round_enc(input_r, key_r):
    return s_box(input_r ^ key_r)


def encrypt(input_p, key1, key2, rounds_count):
    result = input_p
    for round_N in range(rounds_count):
        round_key = key1 if round_N % 2 == 1 else key2
        result = round_enc(result, round_key)
    return result


def slide_pair(key1, key2, rounds_count):
    plain1_pool,  plain2_pool    =  list(range(0,  2**__bits)),
list(range(0, 2**__bits))
    random.shuffle(plain1_pool)
    random.shuffle(plain2_pool)
    pairs_count = 0
    for plain1 in plain1_pool:
```

```
        secret1 = encrypt(plain1, key1, key2, rounds_count)
        for plain2 in plain2_pool:
            pairs_count += 1
            poss_keys, secret2 = [], encrypt(plain2, key1, key2,
rounds_count)
            exp_key1_pool,    exp_key2_pool    =    list(range(0,
2**__bits)), list(range(0, 2**__bits))
            for exp_key1 in exp_key1_pool:
                for exp_key2 in exp_key2_pool:
                    if  round_enc(round_enc(secret1,  exp_key1),
exp_key2) == secret2:
                        poss_keys.append([exp_key1, exp_key2])
            for exp_key1, exp_key2 in poss_keys:
                if     round_enc(round_enc(plain1,     exp_key1),
exp_key2) == plain2:
                    if exp_key1 == key1 and exp_key2 == key2:
                        return pairs_count
    return pairs_count


def brute_force(plain, secret, round_number):
    expec_key1_pool,  expec_key2_pool  =  list(range(0,  2  **
__bits)), list(range(0, 2 ** __bits))
    random.shuffle(expec_key1_pool)
    random.shuffle(expec_key2_pool)
    for exp_key1 in expec_key1_pool:
        for exp_key2 in expec_key2_pool:
            expec_secret  =  encrypt(plain,  exp_key1,  exp_key2,
round_number)
            if expec_secret == secret:
                return


slide_time, brute_time = 0, 0
pairs_count_total, pairs_count_numb = 0, 0
round_number_pool = [5, 50, 100, 200]
```

```
for round_number in round_number_pool:
    for i in range(100):
        key1, key2 = random.randint(0, (2**__bits) - 1),
random.randint(0, (2**__bits) - 1)
        slide_start_time = time.time()
        pairs_count = slide_pair(key1, key2, round_number)
        pairs_count_total += pairs_count
        pairs_count_numb += 1
        slide_time += time.time() - slide_start_time

        plain = random.randint(0, (2**__bits) - 1)
        secret = encrypt(plain, key1, key2, round_number)
        start_time = time.time()
        brute_force(plain, secret, round_number)
        brute_time += time.time() - start_time

    print(f'{round_number} - {slide_time} - {pairs_count_total /
pairs_count_numb} - {brute_time}')
```