НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

на тему АЛГОРИТМИ ЛІНІЙНОГО КРИПТОАНАЛІЗУ ШИФРУВАННЯ НА ПІДСТАНОВКАХ ДОВІЛЬНОЇ РОЗРЯДНОСТІ

Виконав: студент ІІ курсу, групи КВ-71мп
(шифр групи)

Чабан Ярослав Юрійович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Тесленко О.К. _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики

Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність – 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В. П. Тарасенко
«___»_____2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Чабану Ярославу Юрійовичу

1. Тема дисертації «Алгоритми лінійного криптоаналізу шифрування на підстановках довільної розрядності», науковий керівник дисертації Тесленко Олександр Кирилович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 2018 р. №_____

2. Термін подання студентом дисертації «12» грудня 2018 р.

3. Об'єкт дослідження: алгоритм шифрування на основі підстановок довільної розрядності.

4. Предмет дослідження: алгоритми лінійного криптоаналізу

5. Перелік завдань, які потрібно розробити:
   - Проведести систематизацію методів і алгоритмів шифрування та алгоритмів криптоаналізу.
   - Дослідити алгоритм шифрування на основі підстановок довільної розрядності, та його криптоаналіз
   - Провести експериментальне дослідження

6. Перелік ілюстративного матеріалу:

   – Презентація.

7. Перелік публікацій:
   – виступи на конференціях та дві наукова роботи

8. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вибір теми магістерської дисертації	03.09.2017	
2.	Вивчення та аналіз існуючих матеріалів	03.11.2017	
3.	Проходження практики	03.09.2018	
4.	Написання першого розділу	03.07.2018	
5.	Написання другого розділу	03.08.2018	
6.	Написання третього розділу	03.09.2018	
7.	Оформлення текстової частини магістерської дисертації	03.10.2018	
8.	Попередній розгляд магістерської дисертації на засіданні кафедри	26.11.2018	

Студент _____ Чабан Я. Ю.
(підпис)

Науковий керівник дисертації _____ Тесленко О. К.
(підпис)

РЕФЕРАТ

Актуальність теми.

Масове впровадження електронного документообігу супроводжується значним збільшенням обсягів використання криптографічних перетворень для захисту контенту. Це спонукає створювати та використовувати алгоритми шифрування, прості в експлуатації та орієнтовані на широкий діапазон співвідношень часу виконання в програмних і апаратних реалізаціях та криптостійкості. На данний час існує багато алгоритмів шифрування. Але наразі немає такого, який має підтверджену криптостійкість та високу швидкодію. Отже наразі, дана тама магістерської дисертації є актуальна.

Об'єкт дослідження – алгоритм шифрування на основі підстановок довільної розрядності.

Предмет дослідження – алгоритми лінійного криптоаналізу.

Методи досліджень – порівняльна характеристика найбільш вживаних алгоритмів шифрування, порівняння алгоритмів за швидкодією та криптостійкістю, дослідження переваг та недоліків алгоритму шифрування на основі підстановок довільної розрядності, дослідження алгоритмів криптоаналізу, та дослідження на криптоапаліз запропонованого алгоритму шифрування.

Мета роботи: є дослідження нового алгоритму шифрування(шифрування на основі підстановок довільної розрядності), розгляд можливих алгоримтів криптоаналізу та аналіз даного алгоритму. Для цього було визначено наступні завдання, які вирішуються в роботі:

1. Проведення систематизації методів і алгоритмів шифрування та алгоритмів криптоаналізу.
2. Дослідження алгоритму шифрування на основі підстановок довільної розрядності, та дослідження його залежностей не тільки у межах блоку, а й міжблокову залежність.
3. Експериментальне дослідження .

Наукова новизна одержаних результатів полягає в наступному: Показано, при яких умовах запропонований алгоритм шифрування реалізує нелінійну залежність усіх виходів від усіх змінних на вході.

Практична цінність в тому, що проведені експериментальні дослідження, які показали, що використання даного алгоритму дозволяє забезпечити нелінійну залежність усіх розрядів криптограми від всіх розрядів оригінального повідомлення.

Апробація роботи.

Основні положення і результати роботи представлені та обговорені на:

- XI конференція молодих вчених «Прикладна математика та комп'ютинг» ПМК-2018-2; м. Київ, 14-16 листопада 2018 р.
- IV Міжнародна науково-технічна Internet-конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами». м. Київ, 22-23 листопада 2018 р., Національний університет харчових технологій.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, трьох розділів та висновків.

У *вступі* подано загальну характеристику роботи, та описані основні риси роботи.

У *першому розділі* наведено відомих результатів та формулювання мети роботи

У *другому розділі* наведено та проаналізовано теоретичні дослідження алгоритму шифрування на основі підстановок довільної розрядності

У *третьому розділі* розроблено опис і реалізацію розробленого програмного продукту

У *висновках* представлені результати проведеної роботи.

Робота представлена на 80 аркушах, містить 20 рисунків, 10 таблиць і посилання на список використаних літературних джерел з 11 найменувань.

Ключові слова: AES, DES, Конструктивні модулі, Шифрування.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard, симетричний алгоритм блочного шифрування прийнятий як американський стандарт шифрування урядом США.

ASP – Active Server Pages, це технологія від компанії Microsoft, що дозволяє динамічно формувати автоматично оновлювані веб-сторінки з боку веб-сервера.

CLI – Common Language Infrastructure, специфікація загальномовної інфраструктури.

CLR – Common Language Runtime, це компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

DES – Data Encryption Standard, це симетричний алгоритм шифрування певних даних , стандарт шифрування прийнятий урядом США із 1976 до кінця 1990-х, з часом набув міжнародного застосування.

FEAL – Fast data Encipherment Algorithm, блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.У ньому використовуються 64-бітовий блок і 64-бітовий ключ.

IoC – Inversion Of Control

NESSIE – New European Schemes for Signatures, Integrity and Encryption, європейський дослідницький проект для визначення безпечних шифрувальних алгоритмів.

OOP – Object Oriented Programming, одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

RSA – Rivest–Shamir–Adleman, криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

SAC – Strict Avalanche Criterion, це узагальнення лавинового ефекту. Вона виконується якщо, завжди коли один вхідний біт був змінений, кожен біт на виході змінюється з імовірністю 50%.

XOR – Exclusive Disjunction, логічна та бітова операція, що приймає значення «істина» тоді і лише тоді коли значення «істина» має рівно один з її операндів.

ІТ – інформаційні технології.

КМ – Конструктивний Модуль

ОККМ – Одновимірні Каскади Конструктивних Модулів.

ПЛІС – Програмована логічна інтегральна схема електронний компонент, що використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС не визначається при виготовленні, а задається за допомогою програмування.

Вступ

Масове впровадження електронного документообігу супроводжується значним збільшенням обсягів використання криптографічних перетворень для захисту інформації. Це спонукає створювати та використовувати алгоритми шифрування, які будуть прості у використанні та напрявленні на великий діапазон співвідношень часу виконання в апартних і програмних реалізаціях та криптостійкості. Одним із ефективних алгоритмів шифрування може бути алгоритм на лінійних структурах, програмна так і апаратна реалізація таких структур порівняно проста. Сучасні алгоритми шифрування мають залежність усіх виходів від значення усіх змінних на вході тільки у межах блоку, даний алгоритм ж може мати залежність від усіх змінних не тільки у межах блоку, а й загалом, що говорить про криптостійкість.

Метою даної роботи є встановлення таких умов, при яких алгоритм матиме залежність усіх розрядів криптограми від усіх значеннь вхідних даних оригінального повідомлення, що підвищить можливість використання на основі подальших дослідженнь на криптостійкість.

РОЗДІЛ 1. ОГЛЯД ВІДОМИХ РЕЗУЛЬТАТІВ ТА ФОРМУЛЮВАННЯ МЕТИ РОБОТИ

1.1 Сучасні алгоритми шифрування

Шифрування це процес перетворення оригінального тексту, інформаційного повідомлення у інше, так, щоб доступ до оригінальної інформації мали тільки користувачі, у яких є на це доступ. У загальному представленні шифрування, можна виділити такі пункти:

- оригінальне слово(текст);
- шифр, який утворює шифротекст;
- шифротекст, який може бути прочитаним тільки опісля процесу дешифрування.

Через низку технічних причин, представлення шифрування використовує псевдовипадковий ключ шифрування, який є створений алгоритмом шифрування. Система побудована так, що, можливий процес дешифрування без знання відповідного ключа, однак це вимагає високо-професійних навиків та використання значної кількості ресурсів обчислювальних машин. Користувач, який має на це доступ, може з легкістю провести процес дешифрації шифротексту та отримати оригінальне повідомлення, з використанням ключа, який надсилається відправником тільки тим користувачам, які мають на це доступ(авторизованим).

Можна провести поділ алгоритмів шифрування за декількаами напрямками, для прикладу, класифікація може відбуватись на основі типу використаного ключа:

- Алгоритми шифрування з використанням симетричного ключа;

- Алгоритми шифрування, які використовують асиметричний ключ.

Аналогічно, якщо взяти до уваги тип не ключа, а оригінальних даних, то можна виділити:

- Алгоритми блокового шифруваннння;

- Алгоритми потокового шифруваннння (рис 1.1).

Шифрування, яке використовує один ключ для обох процесі шифрування та дешифрування називається симетричним. Якщо ключі різні – асиметричним. Процес шифрування називається блоковим, якщо він проходить з поділом оригінального тексту на деякі одиниці із певним розміром, якщо ж він відбувається без поділу та безперервно, то він класифікується як шифрування потокове. Більшого поширення набула класифікація на основі типу ключа, який був використаний.

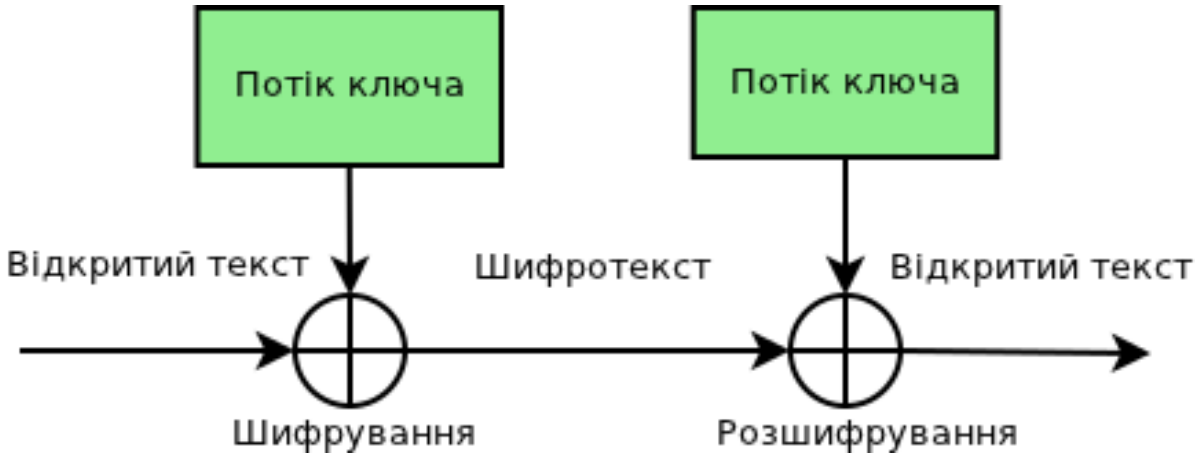

Рис. 1.1- Алгоритм шифрування з потоковими даними

Переваги алгоритмів шифрування, які використовують симетричний ключ:

- Мають велику можливу пропускну здатність;

- Можливий процес комбінування симетричних шифрів, що дозволяє отримати шифри сильніші;

- Ключі, які використовуються під час шифрування алгоритмами з симетричним ключем є короткими.

Серед недоліків те, що при передачі даних між двома авторизованими користувачами необхідно забезпечити безпеку передачі ключа в обох кінцевих пунктах, а також те, що на практиці використання вимагає частої кількості заміни ключа.

Головним недоліком симетричного алгоритму шифрування є той факт, що якщо ключ був перехопленим, то це гарантує можливість розшифрування інформації, що передається. Оскільки в асиметричних шляхах шифрування існують двоє ключів, ця проблема вирішується. Існують два типи ключів (рис. 1.2) які використовуються, це публічні ключі, доступ до яких мають всі авторизованим до шифрування користувачам,та приватний ключ, доступ до якого надається тільки тим авторизованим користувачам, які які мать доступ до процесу дешифрування оригінального повідомлення. Зазвичай ці типи ключів відрізняють один від одного розміром, оскільки приватний ключ необхідно повинен бути більшим.

Рис. 1.2- Принцип роботи асиметричних шифрів

Оскільки перший тип ключа – публічний, необхідності захищати передачу цього ключа немає. Оригінальне повідомлення, яке було перетворено у шифротекст, можливо перетворити назад у необхідну інформацію за допомогою того самого алгоритму, який використовувався при оберненому процесі, для цього необхідно мати авторизований доступ до другого типу ключа – приватного. Ця властивіть вирішує головний недолік симетричних типів шифрування, однак, водночас і є причиною головного недоліку асметричних видів шифрування – швидкодію. Через цей недолік, а також велика необхідністьу використанні значних можливостей обчислювальної техніки під час перетворення оригінального тексту в шифротекст, а також під

час оберненого процесу, зумовили використання алгоритмів, які поєднують асиметричний та симетрчний методи.

Прикладом асиметричного шифрування може бути алгоритм шифрування RSA, який має відкритий ключ, та основується на великому рівні складності обчислювальних процесі факторизації чисел. Неділоком алгоритму, як і в більшості алгоритмах з використанням публічних ключів є його швидкодія, тому він часто використовується поряд з симетричним алгоритмом DES.

1.2 Алгоритми шифрування з використанням симетричних ключів

DES - алгоритм шифрування оригінального повідомлення, який використовує симетричні ключі, та довгий час слугував стандартом для Сполучених Штатів Америки та на міжнародному рівні.[3] Згідно з класифікацією відносно використання початкових даних даний алгоритм являється блоковим алгоритмом, початкова інформація формується у частини які складаються з шісдесят чотирьох бітів. На вхід подається сформований блок вказаного розміру і аналогічний розмір блоку шифротексту подається на вихід. Оскільки алгоритм симетричний, то під час оберенених процесів шифрування оригінального повідомлення та дешифрування шифротексту використовується один алгоритм. Алгоритм складається з підстановки та перемішуванні[4], які формують цикл з використанням бітів оригінального слова, ключа та утворених підстановочних блоків.

Таблиця 1.1 – Початкова перестановка алгоритму

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
33	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
49	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Ключ, який використовує алгоритм складається з комбінації 64 біт, частина комбінації яких являється слабкими ключами, оскільки існує можливість їхнього легкого визначення, що і формує безпеку ключа. 64 біти у вигляді блоку подається у вигляді вхідних даних, які представлені у таблиці 1.1.

Опісля відбувається процес розширення блоку, за визначеною функцією розширення та акумуляція ключа раунда, початкова та кінцева підстановка, порозрядне додавання частин, впродовж 16 раундів циклу. (рис. 1.3)

На даний момент, алгоритм застосовується в більшості, тільки у своїй потрійній модифікації, через ненадійність зумовлена малою довжиною ключа і розміру одиниці поділу. Алгоритм втратив своє міжнародне поширення та звання стандарту на користь новішого симетричного алгоритму AES.

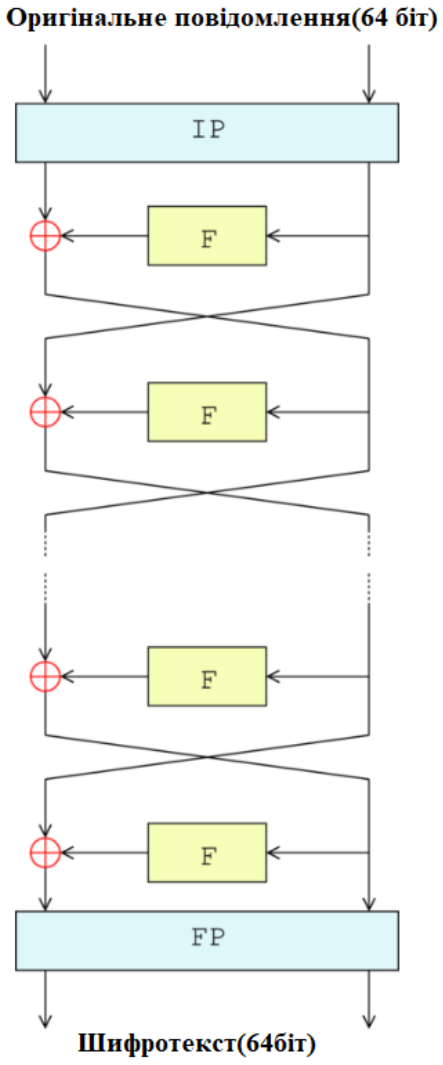

Рис. 1.3 – Конструкція Фейстеля алгоритму DES

AES[1] ще один алгоритм, який використовує симетричні ключі та вхідні дани у формі блока, ключ даного алгоритму може приймати значення 128, 192 чи 256 біт, а розмір одиниці блока дорівнює 128 біт. Став прийнятим стандартом Сполучених Штатів на заміну алгоритму DES, після тривалого і ретельного аналізу.[2] Досі залишається одним з найбільш використовуваних алгоритмів на міжнародному рівні. Алгоритм виконується у послідовності чотирьох послідновних функцій. У наступних рисунках зображені послідовні функції алгоритму.[1] У першій функції «subByts» (рис. 1.4) послідовно кожному байту знаходиться відповідність з сформованої таблиці пошуку.

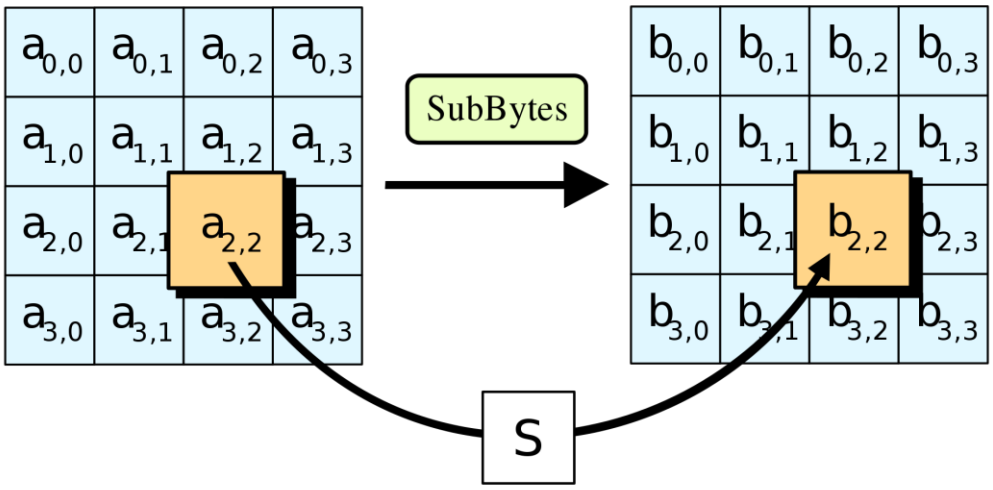

Рис. 1.4 – Перша функція алгоритму AES

Наступним кроком у алгоритмі AES є циклічний зсув рядка з таблиці станів горизлнтально на задану кількість байтів, яка залежить від нумерації рядка.

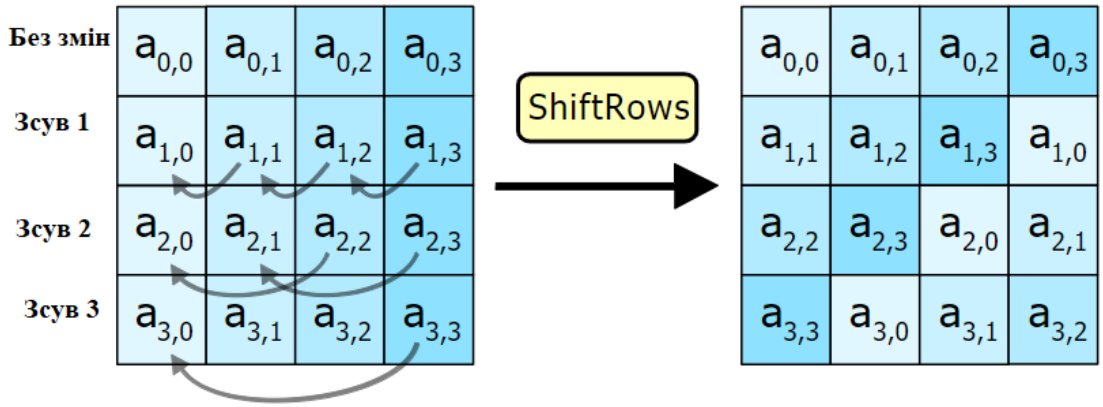

Рис. 1.5- Друга функція симетричного алгоритму AES

У наступному кроці, кожна колонка таблиці стану множиться на зафіксований многочлен, що зображено на рис. 1.6.[2]

У фінальній процедурі «addRoundKey» , з таблиці стану беруться усі байти і кожен з них об'єднується з RoundKey за допомогою операції XOR.(рис. 1.7)

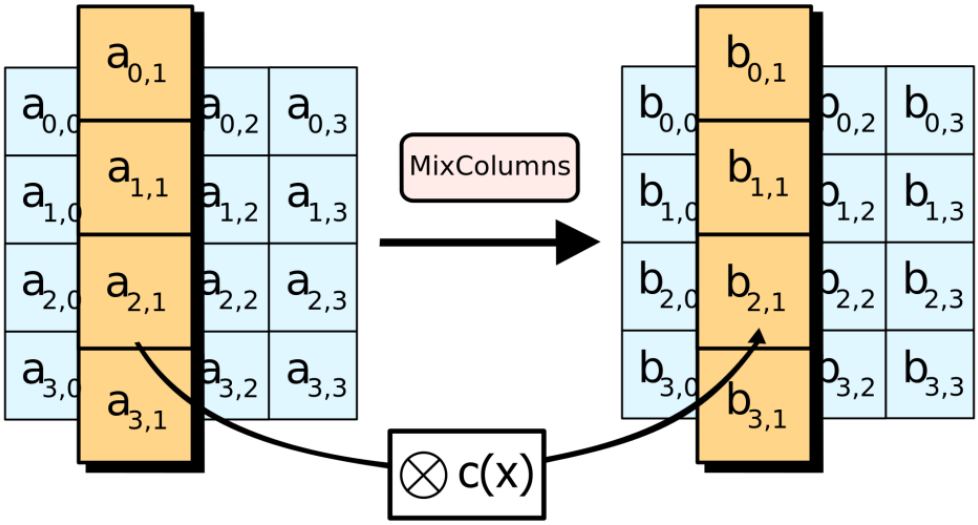

Рис. 1.6 – Третя функція симетричного алгоритму AES

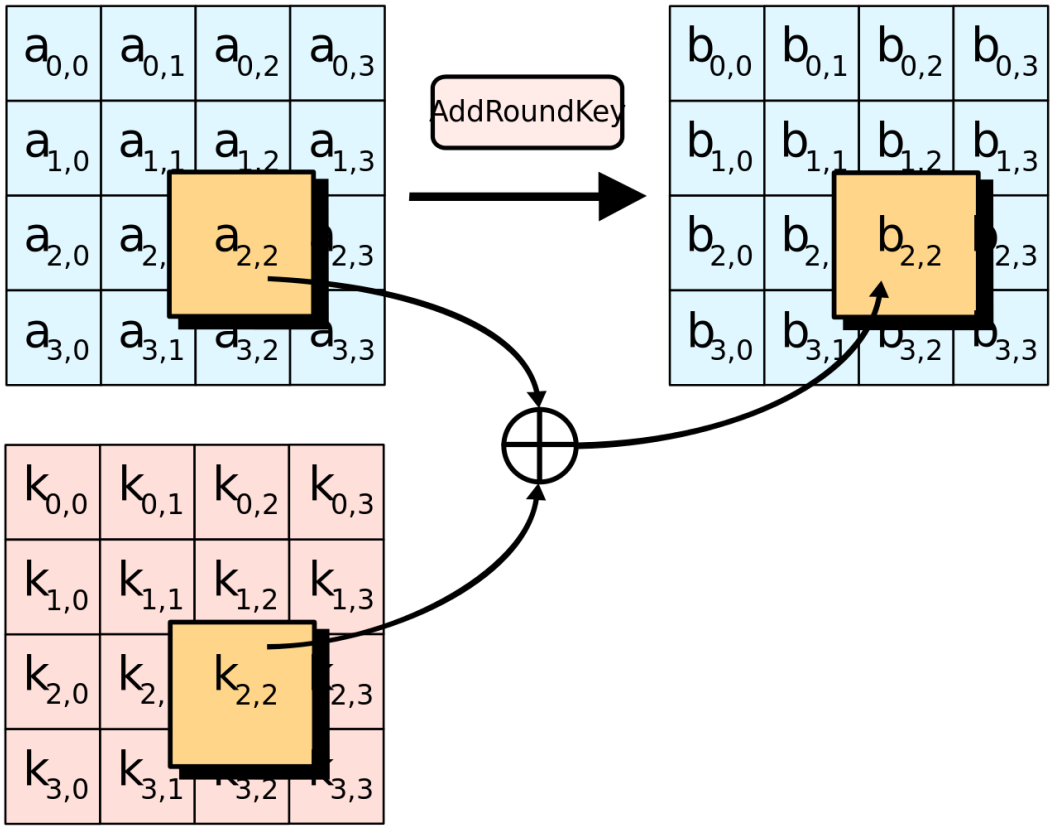

Рис. 1.7 – Остання з циклу процедур алгоритму

Симетричний блочний алгоритм шифрування «Калина» - прийнятий стандар, що використовується в Україні. Проте його програмна реалізація є значно повільнішою та менш ефективною в сучасних критеріях у порівнянні з рішеннями, такими як AES.

1.3 Алгоритм шифрування «Калина»

Розуміючи ефективність проведення відповідних тендерів, основуючись на досвіді держав Європи та США, в Україні був проведений тендер на блочні шифри, які претендували на звання державного стандарту. До претендентів ставили великі вимоги, такі як обмеження щодо розміру ключа для алгоритму, очевидно великий потенційний рівень безпеки від криптоаналізу та високі показники швидкодії у програмній реалізації. У тендері переміг алгоритм шифрування «Калина», яка відтоді являється державним стандартом, в якому описуються десять типів операцій щодо даного блочного шифру.

«Калина» – це блочний симетричний алгоритм шифрування, який буврозроблений ЗАТ «Інститут інформаційних технологій» у м. Харків.[5] Даний алгоритм має розміри блоку 128, 256 і 512 бітів також підтримує такі ж довжини ключів. При побудові шифру основна увага розробників була приділена забезпеченню високого рівня криптографічної стійкості, а також досягненню високих показників продуктивності в апаратній та програмній реалізаціях. Також був врахований інтелектуальний внесок в практику проектування такриптоаналізу блочних алгоритмів шифрування провідних криптографів, що отримав значний розвиток в ході проведення міжнародних проектів NESSIE і AES. Широко застосовувалися результати наукових робіт по методам криптоаналізу і проектування симетричних шифрів останніх років, що були опублікованіукраїнськими авторами, а також власні дослідження і розробки ЗАТ «ІІТ» у цьому напрямку.

Алгоритм шифрування «Калина» являється ітеративною процедурою, яка складається з кінцевої та початкової рандомізації, а також двох різних ітеративних послідовних шифрувальних перетворень. Структура алгоритму «Калина» аналогічна структурі AES, яка забезпечує гарне розсіювання і перемішування. На вхід кожного шифрувального перетворення подається поточний стан шифру та необхідна кількість ключів (раундовий та основний). Відкритий текст копіюється в поточну матрицю стану перед початком шифрування, по завершенню шифрування в поточному стані знаходиться зашифрований шифротекст. Кількість циклів шифрування залежить від довжини ключа (мастер-ключа), при цьому його довжина ключа не може бути менше розміру того блоку, який шифрується.[5]

Позначення в алгоритмі

0x	– префікс шістнадцяткових літералів
$GF(2^8)$	поле Галуа для поліному $x^8 + x^4 + x^3 + x^2 + 1$
$\oplus$	сума по модулю 2 (XOR)
$\lfloor x \rfloor$	ціла частина числа
$\lvert X \rvert$	довжина бінарної послідовності
$L_{l,r}(X)$	функція, якка повертає r менш значимих бітів бінарної послідовності довжиною l
$R_{l,r}(X)$	функція, що повертає r більш значимих бітів бінарної послідовності довжиною l
>>	логічний зсув праворуч
<<	логічний зсув ліворуч
>>>	циклічний зсув праворуч
<<<	циклічний зсув ліворуч

$+$	додавання по модулю 2^{64}
$\bigotimes$	скалярний добуток векторів
l	розмір блоку (128, 256, 512 біт)
k	розмір ключа (128, 256, 512 біт) ($k = l$або $k = 2l$)
c	кількість рядків в матриці стану
V_j	вектор розміру j
$T_{l,k}^{(K)}$	базове перетворення при шифруванні
$U_{l,k}^{(K)}$	базове перетворення при розшифруванні
$W_1 \| \| W_2$	Конкатенація бінарних послідовностей
$\delta \cdot \Lambda$	послідовне виконання операцій (справа наліво)
t	число перетворень при виконанні $T_{l,k}^{(K)}$ чи $U_{l,k}^{(K)}$
$\prod_{i=1}^{t} \Lambda^{(i)}$	послідовне виконання операцій $\Lambda^{(1)}$, $\Lambda^{(2)}$, … , $\Lambda^{(t)}$, починаючи з $\Lambda^{(1)}$
$\mu_l^{(j)}$	представлення невід'ємного цілого числа j у вигляді бінарної послідовності з l бітів
$Kalyna - l/k$	виконання $T_{l,k}^{(K)}$ чи $U_{l,k}^{(K)}$ для розміру блоку l та розміру ключа k

Таблиця 1.2- Кількість раундів для різних комбінацій довжин блоків і ключів

	Довжина блоку	Довжина ключа	Кількість раундів	Кількість рядків внутрішньої матриці стану
1	128	128	10	2
2		256	14	
3	256	256	14	4
4		512	18	
5	512	512	18	8

Базовий процес перетворення обробляє вхідний блок розміром lбіт (відкритого чи зашифрованого тексту). Вводиться поняття матриці внутрішнього стану G, яка представляється як $(g_{i,j})$, де $g_{i,j} \in V_8$, $i = \overline{0,7}$та $j = \overline{0,\ c-1}$. Матриця внутрішнього стану заповнюється вхідними байтами B_1, B_2, …, $B_{l/8}$ по рядкам.

Іншими словами, вхідний блок розбивається на колонки по 8 байт. Наприклад, якщо вхідний блок розміром 128 біт представлено як D5810DDF7F8157482705CADC256DF03B, то матриця внутрішнього стану матиме вигляд такий, як представлено у таблиці 1.3:[6]

Таблиця 1.3– Матриця стану

	j=0	j=1
i=0	D5	27
i=1	81	05
i=2	0D	CA
i=3	DF	DC
i=4	7F	25
i=5	81	6D
i=6	57	F0
i=7	48	3B

Процес шифрування можна описати формулою

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

де K – ключ шифрування довжиною kбіт,

$\eta_l^{(K_T)}$ – функція додавання по модулю 2^{64} матриці внутрішнього стану з раундовим ключем K_v,

π_l' - функція нелінійного взаємно-однозначного відображення (S-box), що обробляє V_8 вектори

τ_l – перестановка елементів $g_{i,j} \in GF(2^8)$матриці внутрішнього стану (циклічний зсув байтів вправо),

ψ_l – лінійне перетворення елементів матриці внутрішнього стану з використанням алгебри Галуа,

$K_l^{(K_v)}$ – функція додавання по модулю 2 (XOR) раундового ключа і матриці внутрішнього стану.

В функціях π_l', τ_l та ψ_l вхідним параметром є $x \in V_l$, а вихідне значення $\chi(x) \in V_l$, $\chi \in \{\pi_l', \tau_l, \psi_l\}$подається у вигляді матриць розміром $8 \times c$, $c \in N$ залежно від l.[5]

Таблиця 1.4- Підстановка π_0'

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

Таблиця 1.5- Підстановка π_1'

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

Таблиця 1.6- Підстановка π_2'

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

Таблиця 1.7- Підстановка π_2'

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

У загальному вигляді, алгоритм можна зобразити:

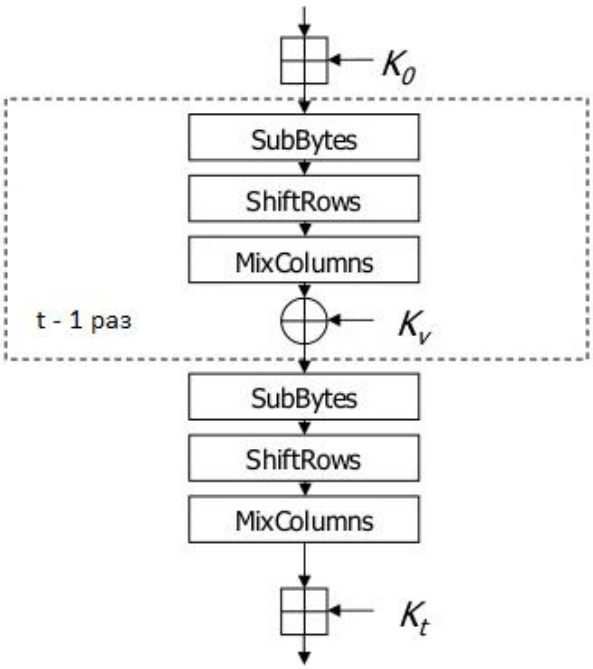

Рис. 1.8- Графічне подання алгоритму «Калина»

1.4 Шифрування на основі підстановок довільної розрядності

Одним із ефективних алгоритмів шифрування може бути алгоритм, який ґрунтується на суперпозиції підстановок довільної розрядності. В роботі [8] показано, що регулярні структури – одновимірні каскади конструктивних модулів (ОККМ), які ґрунтуються на одно розрядних однакових КМ, можуть реалізовувати до 850 різних підстановок з будь якою розрядністю. В свою чергу, як програмна так і апаратна реалізація таких структур порівняно проста.

Способом використання об’єднаних каскадів(одновимірних) конструктивних модулів(КМ), реалізується підстановка необмеженої розрядності на ПЛІС, що зображено на рис. 1.9.

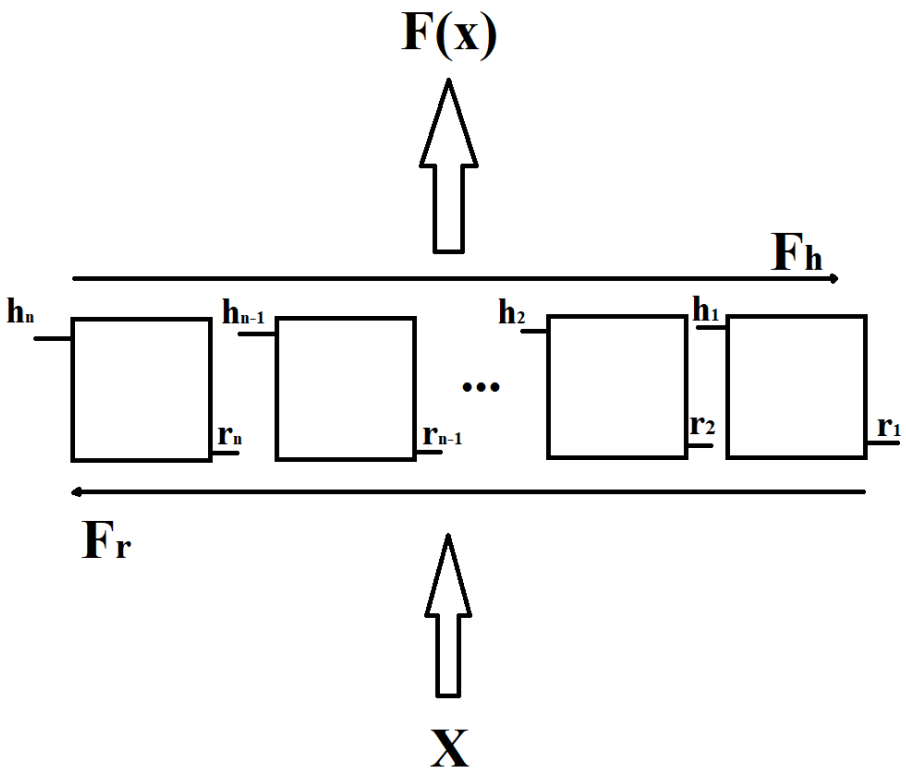

Рис. 1.9 – Одновимірний каскад КМ

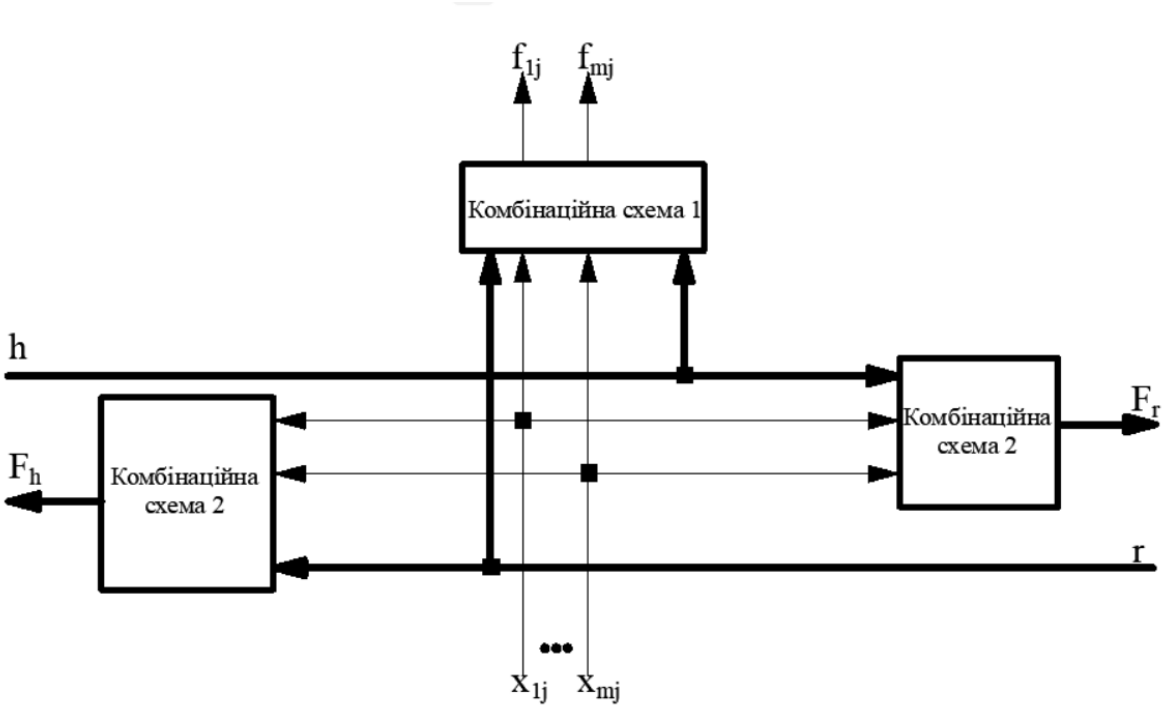

Рис.1.10- Структура конструктивного модуля (КМ) ОККМ

В даному випадку всі конструкктивні модулі однакові. Коструктивнйи модуль описується наступними булевими функціями:

1. Функція $f_h(h, x_1..x_m)$ на правому виході боковому (Комбінаційна схема номер 3).
2. Функція $f_r(r, x_1..x_m)$ на боковому лівому виході (комбінаційна схема номер 2).
3. М – функцій $f_i(h, x_1..x_m, r)$ $(i = 1,2..m)$ на первинному виході КМ. В подальшому, ці функції розглядаються як кортеж (впорядкований набір) $F(h, x_1..x_m, r) =< f_1(h, x_1..x_m, r)...f_m(h, x_1..x_m, r)$ (КС 1).[8]

У більшості відомих алгоритмів шифрування використовується наступний набір операцій – пересилання даних, порозрядна Xor, додавання (віднімання) та циклічний зсув. Для спрощеної аналітичної оцінки швидкості алгоритмів шифрування будемо вважати, що вказані операції виконуються з однаковою швидкістю, яку приймемо за одиницю. Згідно з попереднім швидкість алгоритму на підстановках можна оцінити як 2w, де w – кількість байт файлу, який шифрується, m кількість підстановок в ключі, або 2 операції на байт. В алгоритмі ГОСТ 28147-89 шифруються блоки по 8 байт, при цьому

використовуються 32 цикли Фейстеля. В кожному циклі використовується операція додавання, 8 операцій пересилання даних (реалізація значень 8 4-х розрядних підстановок), операція циклічного зсуву та операції Xor та пересилання для завершення циклу – всього 12 операцій. Таким чином маємо 32*12=384 операцій, або 48 операцій на один байт. В алгоритмі AES шифруються блоки по 16 байт (в офіційному стандарті), за 10 раундів. Кожний раунд має наступні операції – Xor, 16 пересилань (реалізація значень 16 8 - розрядних підстановок), 3 операції циклічного зсуву і операцію множення на матрицю, що при попередній підготовці таблиць множення в полі Галуа(2 8), потребує 16*4 операцій пересилання та 16*3 операцій Xor. [7]

Таким чином, загальна кількість операцій для 10 раундів, враховуючи, що останній раунд не потребує множення на матрицю, дорівнює приблизно 1200 операцій, або 75 операцій на байт. Отже шифрування підстановками може мати перевагу по швидкості, при умові, що кількість підстановок в ключі не перевищує 24 ($m \leq 24$), що дає можливість прогнозу хорошої крипостійкості при великій швидкості.

1.5 Криптоаналіз

Розділ науки криптології, який вирішує і досліджує математичні проблеми і способи порушення цілісності та конфіденційності оригінальної інформації, при цьому не маючи ключа, називається криптоаналізом. У загальному випадку, криптоаналіз можна розуміти як процес з’ясування невідомого ключа, а також методи виявлення можливих уразливих місці алгоритмів криптографічних. Коли термін був увведений науковцем криптологом з Сполучених штатів америки Вілльямом Фрідманном у 1920 році, він був оснований на деяких лінгвістичних властивостях і закономірностях природного оригінального тексту і, відовідно, грунтувались тільки на використанні підручних засобів, таких як олівець та папір. Однак, з часом в науці про криптоаналіз все більше зросла важливість методів технічних та математичних, які реалізуються на криптоаналітичних, спеціально призначених для цього комп’ютерах.

Атакою на конкретний шифр, називається спосіб спроби розкриття даного щифру, за допомогою способів криптоаналізу. Така атака, яка була проведена успішно називається розкриттям чи зломом.

1.5.1 Класичний криптоаналіз

Поряд з криптографічним розвитком розвинувся і криптоаналіз, новостворенні шифри швидко були зламаними і це вимагало створення нових, а щоб зламати нові алгоритми, розвитку потребували спросби криптоаналізу. Цей циклічний процес зумовив чітке поєднання термінів про криптографію та криптоаналіх, адже тільки перевіривгши та врахувавши усі можливі способи атак та злому можна побудувати систему яка буде стійкою до атак. Хоч як термін, поняття криптоаналізу з’явилось відносно, способи атак та злому існували століттями тому. Згаданий раніше, відомий арабський вчений Алль-Кінді написав свій «Дешифрування криптографіічних повідомлень: Манускрипт» ще у 9 столлітті. У тій праці описані методи частитного криптографічного аналізу, який є дуже поширеним способом злому великох кількості класичних, який оснований на можливому припущенні про те, що існує певний, не тривіальний статчний розподіл символів, окрім того, послідовностей цих символів, як і в оригінальному відкритому тексті, так і уже в шифротексті. Цей, не тривальній статичний розподіл буде зберігати з великою точністю до заміни символів в обох процесах, шифрування та дешифрування. Такий спосіб дає хороші результати на моно-алфавітних шифрах, адже умова існування доволі великої довжини оригінального тексту повідомленння защифрованого дозволяє припустити, що частота появи символу у оригніальній мові та наявність відповідного защифрованого символу у шифротексті приблизно рівні.

Цей факт дозволяє зробити припущення, що саме цим символом і була зашифрована дана літера. Прикладом роботи даного методу криптоаналізу можна навести підрахунок точної кількості усіх з присутніх символів та

подальшої процедури поділу отриманої кількості на певний конфіціент або на сто, щоб отримана відповідь була у відсотковому представленні. Наступним кроком порівнюються отримані відсоткові значення та відповідні значення з таблиці ймовірнісного розподілу символів для можливої мови оригінального повідомлення.

Можна виділити чотири основні та три додаткові криптоаналітичні методи, якщо криптоаналітикам відомий алгоритм шифру, який досліджується. Перші чотири, основні, методи криптоаналізу:

1. Здійснення атаки на основі шифротексту.

2. Проведення атаки на основы відкритих оригінальних текстів та їм відповідних значеннь шифротектсу.

3. Здійснення атаки на підібраний відркритий оригінальний текст, за умови наданої можливості вибору оригінального тексту для шифрування.

4.Проведення атаки на основні відкритого, оригінального, адаптивного відкритого тексту.

Додаткові дві можливі методи криптоаналізу:

1. Здійснюється атака на основі шифротексту, який підбирається.

2. Аналогічний проце атаки,на основі ключа, який підбирається.

Також можна виділити ще один напрямок дослідження на криптоаналіз, що займається вивченням шляхів та методів ататаки на можливі реалізації(апаратні, апаратно-програмні та програмні) криптографічниих перетворень. Такий спосіб проведення криптографічних досліджень називається інженерний криптоаналізом, у ньому досліджуються наступні:

- часовий проміжок криптографічних перетвореннь;
- споживання електроенергії та його коливання;

- збої які відбуваються у надслідок можливого впливу магнітних та електричних полів, негативного впливу експерементальних та критичних темпереатур, процес зміни частості тактових генераторів та інші.

- Здійснення атаки беручи за основу шифротекст

Якщо у науковця чи криптоаналітика є доступ до певної кількості шифротекстів, які були отримані як наслідок чи результат роботи певного алгортиму шифрування. Маючи таку інформацію, цей криптоаналітик має змогу атаку на шифротекст, мета якої це процес знаходження чим більшої кількості відкритих оригінальних текстів, які відповідають тим шифротекстам, які відомі, для того, щоб в результаті знайти ключ, який використовувався під час шифрування. На вході атак, які основуються на даному способі, є дані, які криптоаналітик зазвичай отримує в наслідок перехоплення шифротексту, який передається. Даний тип атак є одним з найменшефективних.

- Здійснення атаки беручи за основу відкриті тексти та відповідні їм шифротексти

Якщо ж у криптоаналітика є доступ як до шифротекстів, так і до відповідних їм оригінальних повідомлень, то існують два можливі наступні способи його дій:

- Пошук ключа, який був використаний для процесу перетворення знайденого відкритого тексту у відповідний йому шифротекст;
- Створення алгоритму, який має змогу провести дешифрацію будь-якого оригінального повідомлення, яке було закодоване використовуючи даний ключ.

Знаючи тексти оригінальних повідомлень здійснення атаки може мати вирішальну роль. Отримання їхнє відбувається з різних джерел, наприклад по розширенню файлу можна дізнатись його вмітс.

- Здійснення атаки беручи за основу підібратий відкритий оригінальний текст

Здійснення такої атаки вимагає від криптоаналітика наявність певної кількості відкритих оригінальних текстів та отриманих за їх допомогою певної кількості шифротекстів. У даному випадку завданням криптоаналітика має бути підбір декілької відкритих тескстів та визначення результату їхнього шифрування.

Вхідні данні для такого виду атаки можна отримати таким чином:

- Створення та відправлення не зашифрованих повідомлень від одного з користувачів, який нібито авторизований, які викоритсовуються шифрування.
- Отримати відповідь, у якій буде міститись шифротекст, який і цитує зміст повідомлення що було підроблене та відправлене.

Здійснення даної атаки надає криптоаналітика можливість підбору блоків оригінального тексту, а це за деяких умов надає дозвіл отримти ще більше можливої інформації про ключ, який використовувався під час шифрування.

- Здійснення атаки, беручи за основу адаптивно підібрані тексти оригінальних повідомлень

Найбільш зручним способом атаки на підібраний оригінальний текст є спосіб при якому окрім наданої можливості вибрати текст повідомлення для шифрування, у криптоаналітика є можливість вибрати який саме текст оригінального повідомлення він буде шифрувати, це рішення він може приймати на основі попередньо отриманих результатів проведення шифрування.

1.5.2 Лінійний криптоаналіз

Лінійним криптоаналізом у криптографії називається такий криптоаналітичний метод розкриття, який використовує лінійні приближення

для роботи його алгоритму. Перші праці про лінійний аналіз були зроблені японським вченим криптологом Міцуру Мацуї у 1993 році.[9] На самому початку цей алгорим був налаштований на спробуту розкриття DES I FEAL.

Надалі даний спосіб криптграфічного аналізу був розширений також на інші алгоритми і на даний час є найбільш поширеним методом криптографічного розкриття блокових шифрів, атаки на потоків шифри також були розроблені. Опісля алгоритм лінійного криптоаналізу був використаний для криптоаналізу інших алгоритмів шифрування. Криптографічні схеми та їх побудова стали наслідком відкриття даного виду криптоаналізу.

Лінійник криптоаналіз відбувається у двох етапах:

- Перший етап, який складається з побудови співвідношень між оригінальним відкритим текстом, зашифрованим текстом та ключем, який використовувався.

- Другий етап, на якому попередні співвідношення використовуються. Також беруться до уваги відомі пари між відкритим оригінальним текстом та зашифрованим текстом, для того, щоб отримати біти ключа.

У загальному зміст можна описати у процесі отримання співвідношень, які отримали назву лінійних апроксимацій.

$$P_{i_1} \oplus P_{i_2} \oplus \cdots \oplus P_{i_a} \oplus C_{j_1} \oplus C_{j_2} \oplus \cdots \oplus C_{j_b} = K_{k_1} \oplus K_{k_2} \oplus \cdots \oplus K_{k_c},$$

де P_n, C_n, K_n — n-ні біти тексту, шифротексту й ключа відповідно.

Ймовірність даного співвідношення для вибраних довільно бітів відповідно відкритого оригінального тексту, зашифрованого тексту та ключа, який використовувався приблизно рівна 0.5.

Співвідношення ж, які мають ймовірність, яка відрізняється від даного числа і беруться для розкриття відповідного алгоритму. Ідея алгоритму який запропонував Міцуру Маціуї[10], полягає у тому, щоб визначити співвідношення апроксимації які мали б дуже високу або дуже низьку ймовірність. Чим більша величина зміщення ймовірності, тим краще можна провести лінійний криптоаналіз, використовуючи меншу кількість відкритих оригінальних текстів для атаки. Спочатку необхідно визначити лінійну

апроксимацію з великою імовірністю зміщення, для кожного з раундів шифру. Опісля усі отриманні співвідношення поєднюються в одну лінійну апроксимацію, йомвірність здійснення якої є меншою. Якщо ж співідношення можна звести до іншої форми, не використовуючи певні біти, це скоротить процес обрахунку.

Висновки до розділу 1

У першому розділі було проведено дослідження та порівняння сучасних алгоритмів шифрування, вказана необхідність у використанні нових підходів для забепечення вимоги швидкодії. Одним з можливих способів досягення цього є використання алгоритму шифрування на основі підстановок довільної розрядності. Перед тим як алгоритм шифрування може претендувати на звання «стандарту» та використовуватись у національному чи міжнародному рівнях, зазвичай проходять роки досліджень та тендерів. Для того, щоб запустити цей процес, показана важливість дослідження на лінійний криптоаналіз, та вказаний можливий процес покращення на пришвидшення даного аналізу – провести дослідження нелінійної залежністі булевих функцій на виходах ОККМ кожного блоку від усіх змінних на виходах даної ОККМ.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ АЛГОРИТМУ ШИФРУВАННЯ НА ОСНОВІ ПІДСТАНОВОК ДОВІЛЬНОЇ РОЗРЯДНОСТІ

2.1 Криптографічно стійкі підстановки

Кожен із кортежів $F(0,x_1..x_m,0)$, $F(0,x_1..x_m,1)$, $F(1,x_1..x_m,0)$ та $F(1,x_1..x_m,1)$ реалізує підстановку – одну із $(2^m)!$ Програмно підстановки задаються масивами на 2^m елементів, кожний з яких повинен містить не менш ніж m-розрядне двійкове число. Позначимо масив для кортежу $F(0,x_1..x_m,0)$ як M_{00}. Аналогічно позначимо масиви M_{01}, M_{10}, M_{11}. Всього можливо задати $((2^m)!)^4$ різних кортежів $F(h,x_1..x_m,r)$ (Комбінаційних схем 1). В програмній моделі, краще використовувати масив M[h,X,r] розмірністю M[0..1,0.. 2^m-1,0..1], тобто

M[0,X,0] => M_{00}

M[0,X,1] => M_{01}

M[1,X,0] => M_{10}

M[1,X,1] => M_{11}

Масиви *M* задаються випадковим чином. Необхідно зауважити, що всі елементи кожного із масиву M_{xy} x,y$\in\{0,1\}$ різні та приймають значення із множини {0,….2^m-1} (як і аргумент X). Функції $f_h(h,x_1..x_m)$ і $f_r(r,x_1..x_m)$ приймають значення з множини {0,1}. Позначимо $f_h(0,x_1,...,x_m)$ як Mh0[0.. 2^m-1], $f_h(1,x_1,...,x_m)$ як Mh1[0.. 2^m-1], $f_r(0,x_1,...,x_m)$ як Mr0[0.. 2^m-1], $f_r(1,x_1,...,x_m)$ як Mr1[0.. 2^m-1].

В загальному вигляді конструктивний модуль (КМ) простого ОККМ описується наступними логічними функціями Fo(v,w1,w2,…,wm,u), fh(v,w1,w2,…,wm) fr(w1,w2,…,wm,u), де Fo(v,w1,w2,…,wm,u) – кортеж (впорядкований набір) із m первісних функцій fjo(v,w1,w2,…,wm,u), j=1,2,…,m, які залежать від m первісних змінних wi та змінних v та u, які поступають на бокові входи КМ, fh(v,w1,w2,…,wm) , fr(w1,w2,…,wm,u) функції, які реалізується на бокових виходах КМ. Шляхом з’єднання КМ в структуру з лінійною складністю (ОККМ) реалізується деяке n- розрядне перетворення

Y=F(X) з будь якою заданою, кратною m, розрядністю даних. В даному контексті змінна X приймає значення із множини n- розрядних двійкових без знакових цілих чисел. Якщо всі КМ одинакові, то такий ОККМ називається регулярним. Якщо для будь яких X1 та X2 , для яких, стверджується умови X1 ≠ X2 і F(X1) ≠ F(X2), то ОККМ реалізує повну підстановку. Задача полягає в визначені властивостей булевих функцій КМ, які б забезпечували реалізацію криптографічно стійких підстановок. Криптографічно стійкими прийнято вважать підстановки, логічні функції яких : а) залежать від всіх змінних, б) суттєво залежать (не лінійно залежать) від всіх змінних, с) відповідають критерію максимуму умовної ентропії (Strict Avalanche Criterion- SAC), суть якого полягає в ймовірності 0,5 зміни значення функції при зміні значення будь якого аргументу. Булева функція f(x1,...,xi,…,xn) не залежить від змінної xi , якщо

f(x1,...,xi-1,0,…,xn) = f(x1,...,xi-1,1,…,xn)

Булева функція f(x1,...,xi,…,xn) лінійно залежить від змінної xi , якщо

f(x1,...,xi-1,0,…,xn) = not f(x1,...,xi-1,1,…,xn).

В цьому випадку f(x1,...,xi,…,xn) = f(x1,...,xi-1,0,…,xn) Å xi . Булева функція f(x1,...,xi,…,xn) відповідає SAC по змінній xi , якщо функція f(x1,...,xi-1,0,…,xn) Å f(x1,...,xi-1,1,…,xn) є балансною (рівномірною).

Згідно з [8,9] для реалізації повних підстановок для будь яких значень розрядності n кратної m достатньо щоб в кожному КМ всі чотири кортежі Fo(h,w1,w2,…,wm,r), де h,r – фіксовані значення змінних v та u, реалізували повну підстановку на змінних w1,w2,…,wm , а тип КМ мав значення (2a, 2a) принаймні на один із боків. Це означає наступне. Кожна з підстановок Fo(h,w1,w2,…,wm,r) може розглядатись як перестановка послідовності кортежів значень змінних w1,w2,…,wm . Дві перестановки в свою чергу утворюють підстановку, яка характеризується своїми циклами. Якщо така підстановка, створена, наприклад, на базі підстановок Fo(0,w1,w2,…,wm,0) і Fo(0,w1,w2,…,wm,1) має принаймні два цикли, а функція (в бік старших розрядів) fh(0,w1,w2,…,wm) приймає одне те ж значення в межах таких циклів

то таку властивість позначено типом 2а (аналогічно для Fo(1,w1,w2,…,wm,0) і Fo(1,w1,w2,…,wm,1)). В загальному випадку при з'єднанні КМ в n-розрядний ОККМ та при надходженні на входи значення X, на виходах окремого КМ реалізуються функції foj(x1,x2,…,xi,xi+1,...,xi+m, xi+m+1,…,xn)= foj(fH(x1,x2,…,xi), xi+1,...,xi+m, fR(xi+m+1,…,xn)), j=1,2,…,m, де fH – результат суперпозиції функцій fh попередніх КМ, а fR - результат суперпозиції функцій fr наступних КМ, xi – значення розрядів X. Отже, для вирішення поставленої задачі необхідно дослідить 1)властивості відповідних суперпозицій булевих функцій.

2.2 Попередня підготовка теоретичних досліджень

Щоб переконатися в найменшій криптографічній стійкості методу, є лінійна залежність і незалежність. Булева функція $f(x_1,...,x_s,\ldots,x_n)$ не залежить від змінної x_s , якщо $f(x_1,...,x_{s-1},0,\ldots,x_n) = f(x_1,...,x_{s-1},1,\ldots,x_n)$

Булева функція $f(x_1,...,x_s,\ldots,x_n)$ лінійно залежить від змінної x_s , якщо $f(x_1,...,x_{s-1},0,\ldots,x_n) = \text{not } f(x_1,...,x_{s-1},1,\ldots,x_n)$.

В цьому випадку $f(x_1,...,x_s,\ldots,x_n) = f(x_1,...,x_{s-1},0,\ldots,x_n) \oplus x_s$.

Аналіз на лінійну залежність і незалежність в даному випадку проводився наступним чином (приклад наведено для двійкового коду з довжиною в три біти):

Нехай задано наступну підстановку:

Табл.2.1- Приклад підстановки

X2	X1	X0	F2	F1	F0
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	1	1	1

Залежність функцій від X2 визначається наступним чином:

Виділяємо відповідні стовбці, де X2 приймає значення 0, 1. Виділяємо значення функції F0 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.2

Таблиця 2.2 F0(X2, X1, X0).

X2	X1	X0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

F0
0
1
0
1
0
1
0
1

Табл. 2.3 Порозрядний xor F0(0, X1, X0) і F0(1, X1, X0)

F0(0, X1, X0)	F0(1, X1, X0)	F0(0, X1, X0) xor F0(1, X1, X0)
0	0	0
1	1	0
0	0	0
1	1	0

Сума значень результатів після порозрядної операції xor для значень F0(0, X1, X2) та F0(1, X1, X2) дорівнює 0. В такому разі можна зробити висновок що F0 не залежить від X2.

Виділяємо відповідні стовбці, де X2 приймає значення 0, 1. Виділяємо значення функції F1 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.4

Таблиця 2.4 F1(X2, X1, X0).

X2	X1	X0	F1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Табл. 2.5 - Порозрядний xor F1(0, X1, X0) і F1(1, X1, X0)

F1(0, X1, X0)	F1(1, X1, X0)	F1(0, X1, X0) xor F1(1, X1, X0)
1	1	0
0	0	0
0	0	0
1	1	0

Сума значень результатів після порозрядної операції xor для значень F1(0, X1, X2) та F1(1, X1, X2) дорівнює 0. В такому разі можна зробити висновок що F1 не залежить від X2.

Виділяємо відповідні стовбці, де X2 приймає значення 0, 1. Виділяємо значення функції F2 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.6

Таблиця 2.6 - F2(X2, X1, X0)

X2	X1	X0	F2
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Табл. 2.7 - Порозрядний xor F1(0, X1, X0) і F1(1, X1, X0)

F2(0, X1, X0)	F2(1, X1, X0)
0	1
1	0
0	1
0	1

F2(0, X1, X0) xor F2(1, X1, X0)
1
1
1
1

Сума значень результатів після порозрядної операції xor для значень F2(0, X1, X2) та F2(1, X1, X2) дорівнює 4. В такому разі можна зробити висновок що F2 лінійно залежить від X2. Залежність функцій від X1 визначається наступним чином:

Виділяємо відповідні стовбці, де X2 приймає значення 0, 1. Виділяємо значення функції F1 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.8

Таблиця 2.8(X2, X1, X0)

X2	X1	X0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

F0
0
1
0
1
0
1
0
1

Табл. 2.9 - Порозрядний xor F0(0, X1, X0) і F0(1, X1, X0) для X1

F0(X2, 0, X0)	F0(X2, 1, X0)
0	0
1	1
0	0
1	1

F0(X2, 0, X0) xor F0(X2, 1, X0)
0
0
0
0

Сума значень результатів після порозрядної операції xor для значень F0(X2, 0, X0) та F0(X2, 1, X0) дорівнює 0. В такому разі можна зробити висновок що F0 не залежить від X1.

Виділяємо відповідні стовбці, де X1 приймає значення 0, 1. Виділяємо значення функції F1 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.10

Таблиця 2.10- F1(X2, X1, X0) з X1

X2	X1	X0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

F1
1
0
0
1
1
0
0
1

Табл. 2.11- Порозрядний xor F1(X2, 0, X0) і F1(X2, 1, X0) відповідно X1

F1(X2, 0, X0)	F1(X2, 1, X0)
1	0
0	1
1	0
0	1

F1(X2, 0, X0) xor F1(X2, 1, X0)
1
1
1
1

Сума значень результатів після порозрядної операції xor для значень F1(X2, 0, X0) та F1(X2, 1, X0) дорівнює 4. В такому разі можна зробити висновок що F1 лінійно залежить від X2.

Виділяємо відповідні стовбці, де X2 приймає значення 0, 1. Виділяємо значення функції F2 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.12.

Таблиця 2.12- F2(X2, X1, X0).

X2	X1	X0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

F2
0
1
0
0
1
0
1
1

Табл. 2.13- Порозрядний xor F2(X2, 0, X0) і F2(X2, 1, X0) відповідно X1

F2(X2, 0, X0)	F2(X2, 1, X0)
0	0
1	0
1	1
0	1

F2(X2, 0, X0) xor F2(X2, 1, X0)
0
1
0
1

Сума значень результатів після порозрядної операції xor для значень F2(X2, 0, X0) і F2(X2, 1, X0) дорівнює 2. В такому разі можна зробити висновок що F2 залежить від всіх розрядів X2. Такий результат для нас є прийнятним.

Залежність від X0.

Виділяємо відповідні стовбці, де X0 приймає значення 0, 1. Виділяємо значення функції F0 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.14

Таблиця 2.14- F0(X2, X1, X0)

X2	X1	X0	F0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Табл. 2.15- Порозрядний xor F0(X2, X1, 0) і F0(X2, X1, 1) відповідно X0

F0(X2, X1, 0)	F0(X2, X1, 1)
0	1
0	1
0	1
0	1

F0(X2, X1, 0) xor F0(X2, X1, 1)
1
1
1
1

Сума значень результатів після порозрядної операції xor для значень F0(X2, X1, 0) і F0(X2, X1, 1) дорівнює 4. В такому разі можна зробити висновок що F0 лінійно залежить від X2.

Виділяємо відповідні стовбці, де X0 приймає значення 0, 1. Виділяємо значення функції F1 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.16

Таблиця 2.16- F1(X2, X1, X0)

X2	X1	X0	F1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Табл. 2.17- Порозрядний xor F1(X2, X1, 0) і F1(X2, X1, 1) відповідно X0

F1(X2, X1, 0)	F1(X2, X1, 1)
1	0
0	1
1	0
0	1

F1(X2, X1, 0) xor F1(X2, X1, 1)
1
1
1
1

Сума значень результатів після порозрядної операції xor для значень F0(X2, X1, 0) і F0(X2, X1, 1) дорівнює 4. В такому разі можна зробити висновок що F0 лінійно залежить від X2.

Виділяємо відповідні стовбці, де X0 приймає значення 0, 1. Виділяємо значення функції F0 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 2.18

Табл. 2.18- Порозрядний xor F2(X2, X1, 0) і F2(X2, X1, 1) відповідно X0

X2	X1	X0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

F2
0
1
0
0
1
0
1
1

Табл. 2.19 Порозрядний xor F2(X2, X1, 0) і F2(X2, X1, 1) відповідно X0

F2(X2, X1, 0)	F2(X2, X1, 1)
0	1
0	0
1	0
1	1

F2(X2, X1, 0) xor F2(X2, X1, 1)
1
0
1
0

Сума значень результатів після порозрядної операції xor для значень F2(X2, X1, 0) і F2(X2, X1, 1) дорівнює 2. В такому разі можна зробити висновок що F0 залежить від всіх розрядів X2.

Результуюча таблиця для цієї підстановки буде мати наступний вигляд вигляд:

Таблиця 2.20-Результати лінійного криптоаналізу

	F0	F1	F2
X0	4	4	2
X1	0	4	2
X2	0	0	4

2.3 Дослідження властивостей суперпозицій функцій

2.3.1 Твердження 1.

Для того, щоб функція f(x1,x2,…,xs)= f1(f2(x1,x2,…,xt), xt+1,...,xs) залежала від змінних x1,x2,…,xs необхідно і достатньо, щоб від всіх своїх змінних залежали функції f1(u, xt+1,...,xs) та f2(x1,x2,…,xt). [8]

Необхідність очевидна. Доведемо достатність. Для цього необхідно показати, що для будь якої змінної xk (k=1,2,…,t) завжди існує принаймні один кортеж <a1,…,ak-1,ak+1,…,as> значень решти змінних, де f(a1,…,ak-1,0,ak+1,…,as) ≠ f(a1,…,ak-1,1,ak+1,…, as). Оскільки функція f2 залежить від змінної xk , то завжди найдеться принаймні один кортеж <a1,…,ak-1,ak+1,…,at> значень решти змінних де f2(a1,…,ak-1,0,ak+1,…,at) ≠ f2(a1,…,ak-1,1,ak+1,…,at). Оскільки функція f1(u, xt+1,...,xs) залежить від змінної u, то завжди найдеться принаймні один кортеж <at+1,…, as> , де f1(0, at+1,…, as) ≠ f1(1,at+1,…, as). Достатність для змінних xt+1,...,xs ґрунтується на тому, що f2(x1,x2,…,xt) не є константою.

Із Твердження 1 випливає, що для залежності булевих функцій, які реалізовані на первісних виходах ОККМ, від всіх розрядів X необхідно, щоб всі функції із кортежу Fo(v,w1,w2,…,wm,u), а також функції fh(v,w1,w2,…,wm) fr(w1,w2,…,wm,u) залежали від всіх своїх змінних. Але ця вимога (назвемо її першою) не є достатньою для крайніх КМ, які характеризуються фіксованими значеннями u та v. Для крайніх КМ згідно з Твердженням 1 достатньою буде вимога (назвемо її другою) залежності від своїх змінних функцій Fo(v,w1,w2,…,wm,r), fr(w1,w2,…,wm,r), Fo(h,w1,w2,…,wm,u) та fh(h,w1,w2,…,wm), де h, r константи, які приймають значення 0 або 1. Для КМ регулярних ОККМ необхідне одночасне виконання обох вимог, оскільки залежність деякої батьківської функції від всіх своїх змінних в загальному випадку не означає залежність від всіх своїх змінних дочірніх функцій, які визначаються шляхом фіксації значень тих чи інших змінних батьківської функції.

2.3.2 Твердження 2.

Для того, щоб функція f(x1,x2,…,xs)= f1(f2(x1,x2,…,xt), xt+1,...,xs) нелінійно залежала від змінних x1,x2,…,xs достатньо, щоб функція f1(u, xt+1,...,xs) нелінійно залежала від всіх своїх змінних, а функція f2(x1,x2,…,xt) залежала від всіх своїх змінних.

Для доведення необхідно показати, що для будь якої змінної xk (k=1,2,…,t) завжди існує принаймні один кортеж <a1,…,ak-1,ak+1,…,as> значень решти змінних, де f(a1,…,ak-1,0,ak+1,…,as) ≠ f(a1,…,ak-1,1,ak+1,…,as) та існує принаймні один інший кортеж <b1,…,bk-1,bk+1,…,bs>, де f(b1,…,bk-1,0,bk+1,…,bs) = f(b1,…,bk-1,1,bk+1,…,bs). Нехай, в гіршому випадку, існує лише один кортеж <a1,…,ak-1,ak+1,…,at> де f2(a1,…,ak-1,0,ak+1,…,at) ≠ f2(a1,…,ak-1,1,ak+1,…,at). Оскільки функція f1(u, xt+1,...,xs) нелінійно залежить від змінної u, то існує кортеж <at+1,…, as> де f1(0, at+1,…, as) ≠ f1(1,at+1,…, as), та існує кортеж <bt+1,…, bs> де f1(0, bt+1,…, bs) = f1(1,bt+1,…, bs). Таким чином маємо

f(a1,…,ak-1,0,ak+1,…,at,at+1,…,as) ≠ f(a1,…,ak-1,0,ak+1,…,at,at+1,…,as)

f(a1,…,ak-1,0,ak+1,…,at,bt+1,…,bs) = f(a1,…,ak-1,0,ak+1,…,at,bt+1,…,bs)

Достатність для змінних xt+1,...,xs ґрунтується на тому, що f2(x1,x2,…,xt) не є константою. Легко бачить, що нелінійна залежність функції f1(u, xt+1,...,xs) від змінних xt+1,...,xs є не тільки достатньою, але і необхідною умовою для нелінійної залежності від цих змінних функції f(x1,x2,…,xs). Аналогічно попередньому можна довести, що достатньою умовою нелінійної залежності функції f(x1,x2,…,xs) від змінних x1,x2,…,xt є також нелінійна залежність функції f2(x1,x2,…,xt) від всіх своїх змінних та залежність функції f1(u, xt+1,...,xs) від змінної u.[8]

Із Твердження 2 випливає, що для нелінійної залежності булевих функцій, які реалізовані на первісних виходах ОККМ, від всіх розрядів X достатньо, щоб всі функції із кортежу Fo(v,w1,w2,…,wm,u) нелінійно залежали від усіх своїх змінних, а функції fh(v,w1,w2,…,wm) fr(w1,w2,…,wm,u) залежали від всіх своїх змінних. Аналогічно попередньому для крайніх КМ достатньо, щоб функції Fo(h,w1,w2,…,wm,u) та Fo(v,w1,w2,…,wm,r) нелінійно залежали від всіх своїх змінних, а функції fr(w1,w2,…,wm,r) та fh(h,w1,w2,…,wm) залежали від своїх змінних. Для КМ регулярних ОККМ достатнім буде виконання відповідних умов як для батьківських так і для відповідних дочірніх функцій.

Висновки до другого розділу

На основі одержаних результатів та результатів, наведених в [8,9] визначаються наступні властивості ОККМ в залежності від значення m.

При m=1 не існує регулярних ОККМ, які б реалізували повні підстановки, в яких будь-який розряд результату залежить від всіх розрядів аргументу, але існують не регулярні ОККМ з такими властивостями. Крім того, при m=1 не існує ОККМ, які б реалізували повні підстановки з нелінійною залежністю всіх розрядів результату від всіх розрядів аргументу.

При m=2 існують як регулярні так, відповідно, і не регулярні ОККМ, які реалізують повні підстановки, в яких будь який розряд результату залежить від всіх розрядів аргументу. Не існують як регулярні так і не регулярні ОККМ, які реалізують повні підстановки, в яких будь який розряд результату нелінійно залежить від всіх розрядів аргументу.

При $m \geq 3$ існують як регулярні так, відповідно, і не регулярні ОККМ, які реалізують повні підстановки, в яких будь який розряд результату нелінійно залежить від всіх розрядів аргументу.

Отже за допомогою ОККМ з регулярною структурою при $m \geq 3$ можна реалізувати стійкі до атак лінійного крипто аналізу підстановки довільної розрядності кратної m, що підтверджено експериментально.

РОЗДІЛ 3. ОПИС І РЕАЛІЗАЦІЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз технічних особливостей програмного забезпечення

Для розробки експеременталної частини була вибрана об'єктно-орієнтована мова програмування c#, яка була розроблена Андерсеном Хейлсбергом та іншими у компанії Microsoft. Статична і строга типізація, та такі властивості ООР, як поліморфізм, абстрація та наслідування надає широкий спектр можливостей, які необхідні для реалізації.

Для розробки технічно складніших програм необхідне використання бібліотек та фреймоврків, які б надавили більш широкі можливість розробнику. Такою технологією, яка була використана при розробці програми була технологія ASP.Net, яка надає широкий спектр можливостей. Оскільки написана майкрософтом насамперед, однак не тільки, для використання мовою c#, вона ідеально підходить для потреб, які необхідні даному додатку. Після компіляції мова переходить у проміжковий етап, який запускається компілятором загальномовного середовища та переводиться у машинний код.

Переваги технології ASP.Net, які зумовили її використання у даній роботі:

- Перевага у швидкості при порівннянні з технологіями, що засновані на скриптах;
- Широкий спектр бібліотек і елементів керування;
- Чіткий розподіл візуальної частини та використаних бізнес правил;
- Можливе розширення моделі обробки запитів.

Окрім мови програмування та технології також необхідно обрати принципи та парадигми програмування які будуть використанні для забезпечення можливості тестування та покращення додатку. У даному випадку були обрані два принципи: інверсія управління та парадигма програмування MVC.

Шаблон MVC дозволяє провести поділ розроблюваної системи на три структурні одиниці, одиниця даних, представлення інтерфейсу користувача та частина керування даними. Це дозволяє забезпечити мінімально можливі зміни з модуля інтерфейсу на внутрішню роботу включно з даними і, відповідно, зв’язок між змінами у структурній одиниці даних і кінцевого представлення користувача.

Метою даного шаблону є забезпечення можливості гнучкої програмної реалізації, що дозволяє надавати можлливість легких подальших змін для розширення додатку та можливість повторно використати узагальнені компоненти додатки. Впорядкованість, яка забезпечується використанням MVC робить систему з прозорою структурою більш зрозумілою.

Якщо розглядати традиційне програмування, то процес комунікації між головною функцією з потоку управління та зовнішньою функцією відбувається після виклику останньої головною функцією. Бібліотека, в якій знаходиться додаткова функція викликає функцію з додаткової бібліотеки, відображаючи весь список можливих команд і запрошучи користувача до вибору однієї. Результатом виклику функції і буде повернення вибору.

Якщо ж використовується інверсія управління, то утворюється спочатку програмний каркас, якому відомі загальні елементи управління, користувацька частина доповнює каркас, але головний процес слідкування за діями користувача і викликами пов'язаної другорядної функції чи підпрограми залишається прерогативою каркасу.

У даній роботі був використаний паттерн програмування який називається впровадження залежності, що реалізує принцип інверсії управління.

Паттерн можна сформулювати наступним чином:

- Стурктурні одиниці вищого рівня не мають бути залежними від структурних одниць які знаходяться на нижчому рівні. Як і в

першому, так і в другому випадку залежність повинна бути від абстракцій.

- У свою чергу абстракції не повинні залежить від внутрішніх деталейц реалізації, які повинні навпаки, залежати від абстракцій.

Переваги:

- Використання паттерну впровадження залежностей можна звести до покращення уже готового коду та проведення модульного тестування, при потребі.
- Ізоляція кінцевого кліента від можливих змін внутрішнього проектуваннння.
- Паралельна та незалежна розробка можлива через загальне використанння інтерфейсів та абстракцій.

Приклад роботи типового класу, що використовує паттерн впровадження залежностей показано на рис. 3.2:

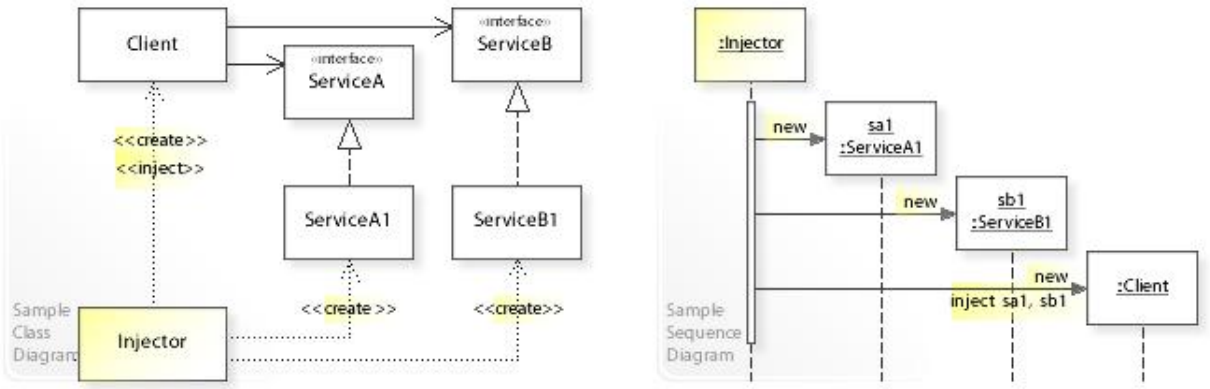

Рис. 3.1- UML класу та діаграми послідовності шаблону проектування Впровадження Залежностей

Усі ці, та багато інших переваг, які значно переважують недоліки та переваги аналогів, зумовили використання .Net фреймоворку для написання, а саме ASP.Net, з використанням MVC та Dependecy Inversion принципів, для успішної реалізації поставлених цілей.

3.2 Структура програми

Дану програму можна поділити на декілька модулів, згідно з використаними паттернами проектування. Зовнішні представлення ігноруються. У класі конструктивних модулів зосередженні два методи GetCustomConstructiveModules та GetConstructiveModules, один з яких являється перевіркою правильності виконання іншого:

```
public int[] GetCustomConstructiveModules(int[] firstArray, int[] secondArray)
{

    int[] result = Enumerable.Repeat(1, 16).ToArray();
    int i = 0;
    var firstElement = firstArray[0];
    var findMe = firstArray[0];
    var flag = true;
    while (flag)
    {
        findMe = firstArray[i];
        for (int j = 0; j <= 15; j++)
        {
            if (secondArray[j] == findMe)
            {
                result[i] = 0;
                i = j;
                if (i == 0) { flag = false; }
            }
        }
    }
    return result;
```

3.2.1 Блок contsractive modules

У даному модулі

```
public static int[] GetConstructiveModules(int[] funx, int[] funy)
{
    var fbok = new int[16]; //result
    var m = 4;
    var kmrl = Math.Pow(2, m) - 1;
    var randomIndex = (int)kmrl + 1;
    for (var j = 0; j <= kmrl; j++)
    {
        fbok[j] = 0;
        Random r = new Random();
        int src = r.Next(0, randomIndex);
        var next = fbok[src];
        while (next == 0)
        {
            fbok[src] = 1;
            var raby = funy[src];
            for (var i = 0; i <= kmrl; i++)
            {
                if (raby == funx[i])
                {
                    src = i;
                    next = fbok[i];
                    break;
                    //return fbok;
                }
    return fbok;
```

Задачою даного блока являється коректне формування конструктивних модулів, за допомогою даних які повертаються в цей модуль інферсією управління.

На основі функцій з DES алгоритму[3] сформовано конструктивний модуль КМ, де

Fo[0,X,0] => M00 : 14, 04, 13, 01, 02, 15, 11, 08, 03, 10, 06, 12, 05, 09, 00, 07

F1 [0,X,1] => M01 : 00, 15, 07, 04, 14, 02, 13, 01, 10, 06, 12, 11, 09, 05, 03, 08

F2 [1,X,0] => M10 : 04, 01, 14, 08, 13, 06, 02, 11, 15, 12, 09, 07, 03, 10, 05, 00

F3 [1,X,1] => M11 : 15, 12, 08, 02, 04, 09, 01, 07, 05, 11, 03, 14, 10, 00, 06, 13,

а X приймає значення із множини {0,1,…,15}, тобто m=4.

Утворені масиви:

- масив Fh0[0,0.. 15]: 0000000000001100;
- масив Fh1[1,0.. 15]: 0111001101010000;
- масив Fr0[0,0.. 15]: 0000010011111111;
- масив Fr1[1,0.. 15]: 0010001100000001;

По алгоритму, згідно [4], сформовано чотири масиви результатів на виходах ОККМ з різними початковими значеннями r та h. Усі сформовані масиви перевірено на підстановку.

Загальну структуру програми можна зобразити у вигляді схематичного зображення яке показано на рис.3.2.

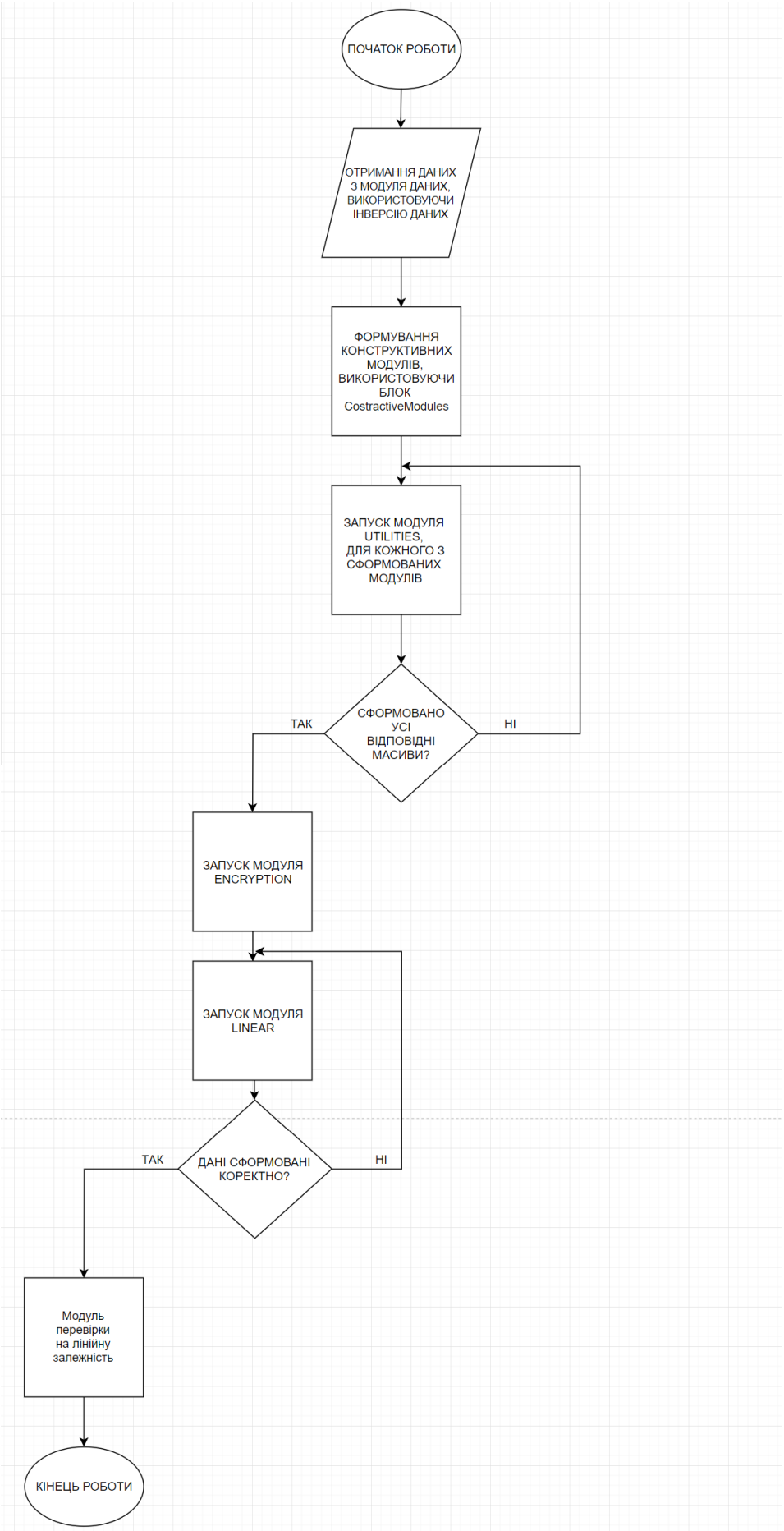

Рис.3.2 – Структурне представлення роботи програми.

Модуль Encryption зберігає у собі всі необхідні методи для роботи з, власне, шифруванням:

```
public string[] CustomConvertToBinary(int value)
    {
        string binary = Convert.ToString(value, 2);
        var temp = "";
        var File = new string[4];
        if (binary.Length < 4)
        {
            var countOfZeros = 4 - binary.Length;
            for (int i = 0; i <= countOfZeros - 1; i++)
            {
                temp += "0";
            }
        }
        else if (binary.Length > 4)
        {
            for (int j = 8; j <= 1024; j += 4)
            {
                var countOfExcessZeros = j - binary.Length;
                if (countOfExcessZeros > 0)
                {
                    for (int k = 0; k < countOfExcessZeros; k++)
                    {
                        temp += "0";
                    }
                    break;
                }
            }
        }
```

```
binary = temp + binary;

switch (binary.Length)
{
    case 4:
        {
            File[0] = binary;
            break;
        }
    case 8:
        {
            File[0] = binary.Substring(0, 4);
            File[1] = binary.Substring(4);
            break;
        }
    case 12:
        {
            File[0] = binary.Substring(0, 3);
            File[1] = binary.Substring(4, 3);
            File[2] = binary.Substring(7, 3);
            break;
        }
    case 16:
        {
            File[0] = binary.Substring(0, 3);
            File[1] = binary.Substring(4, 3);
            File[2] = binary.Substring(7, 3);
            File[3] = binary.Substring(10, 3);
            break;
        }
```

```
    }
    return File;
}
```

3.2.2 Блок encryption

У наступному блоці відбувається процес шифрування отриманих даних:

```
public void EncryptionOfConstructiveModeules(int[,] Mr, int[,] Mh, int[,,] M)
{
    int h0, r0 = 0;
    int Word = 4;
    var Temp = new int[16];
    var File = new string[4];
    var result = new int[100];
    for (int j = 0; j <= MaxValue; j++)
    {
        File = CustomConvertToBinary(j);
        //FORM SIDE VALUES
        for (var i = 0; i <= Word; i++)
        {
            Temp[i] = r0;
            r0 = Mr[r0, File[i]];
        }
        //FORM RESULT VALUES
        for (int i = Word; i <= 1; i--)
        {
            result[i] = M[h0, File[i], Temp[i]];
            h0 = Mh[h0, File[i]];
        }
    }
```

```
    }
```

3.2.3 Коротке представлення інших блоків

Модуль Utilitites назначений для багатьох підтримуючих роботу програми функцій:

```
public class Utilities
{
    public void FancyPrint(int[] arrayToPrint)
    {
        Console.WriteLine("*** Start of Array ***");

        for (int i = 0; i <= arrayToPrint.Length - 1; i++)
        {
            Console.WriteLine("{0} element: {1}", i, arrayToPrint[i]);
        }

        Console.WriteLine("*** End of Array ***");
    }
```

У ньому використовуються загальні функції, які неодноразово викликаються у процесі розробки і є скоріше утилітами для розробника ніж представляють якесь значення для кінцевого користувача.

```
    public void PrintToFile(int[] arrayToPrint)
    {
        string mydocpath =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        using (StreamWriter outputFile = new
StreamWriter(Path.Combine(mydocpath, "WriteLines.txt")))
        {
            outputFile.Write("START");
```

```
            foreach (int line in arrayToPrint)
                outputFile.Write(line);
            outputFile.Write("END");
        }
    }
```

У даній структурній одиниці відбувається вивід даних у файл, для більш зручної подачі отриманих результатів.

```
    public void PrintOfBinary()
    {
        for (int i = 16; i <= 18; i++)
        {
            var result = CustomConvertToBinary(i);
            Console.WriteLine(**START OF ARRAY*");
            Console.WriteLine("ORIGINAL VALUE IS: {0}", i);
            foreach (string value in result)
            {
                Console.WriteLine("This is value: {0}", value);
            }
            Console.WriteLine("*END OF ARRAY*");    }
    }
```

Необхідна перевірка під час початкової розробки.

```
    public int[,] PopulateTwoDimensionalArray(int[] firstArray, int[] secondArray)
    {
        int[,] result = new int[MAX, MAX];
        for (int i = 0; i <MAX_VA;UES; i++)
        {
            result[0, i] = firstArray[i];
            result[1, i] = secondArray[i];
        }
```

```
        return result;
    }
```

У даному модулі відбувається заповнення двохстроннього масиву на основі попередніх значень, які були отримані у блоці коструктивного модуля.

```
    public int[,] PopulateThreeDimensionalArray(int[] firstArray, int[] secondArray)
    {
        int[,,] result = new int[10, 10, 10];

        for (int i = 0; i <= result.Length; i++)
        {
            result[0, i] = firstArray[i];
            result[1, i] = secondArray[i];
            result[2, i] = thirdArray[i];
        } return result;
    }
```

Використовуючи значення які отримані у виконанні попереднього блоку формується трьохсторонній масив з використанням значень масивів Mr, Mh, та M

3.2.4 Результати роботи програми

Блок LinearValidation, з отриманих значень підстановки формує результат суми одиниць опісля побітового XOR відповідних значень функції від змінної яка перевіряється:

```
namespace Degree
{
   public class LinearValidation
   {
      public int GetSumOfOneValues(ArrayList BitwiseOperationResult)
      {
         return BitwiseOperationResult.Select(x => x == 1);
      }
      public ArrayList GetBitwiseOperationResult(int[,] completionValues)
      {
         var result = new int[,];
```

```
            foreach (completionValue in completionValues)
            {
               int[] Fi0 = GetValuesForFi0(completionValue.XValue, completionValues);
               int[] Fi1 = GetValuesForFi1(completionValue.XValue, completionValues);

               var count = 0;
               for (int i = 0; i<= Fi0.Capacity(); i++)
               {
                  if (Fi0[i] ^ Fi1[i] == 1)
                  {
                     count++;
                  }
               }
               result[completionValue.XValues, completionValues.FValues] = count;
            }
            return result;
         }
      }
   }
```

У результаті роботи блоків ConstractiveModules та EncryptionOfConstractiveModules, з використанням блоків Linear і Utilities, сформована підстановка:

Таблиця 3.1- Отримані значення підстановки

X 11	**X 10**	**X 9**	**X 8**	**X 7**	**X 6**	**X 5**	**X 4**	**X 3**	**X 2**	**X 1**	**X 0**	**F 11**	**F 10**	**F 9**	**F 8**	**F 7**	**F 6**	**F 5**	**F 4**	**F 3**	**F 2**	**F 1**	**F 0**
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1	1	0	1	0
0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	0	1	0	1	1	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	1

Для прикладу розглянемо залежність F11 функції від X0:

Виділяємо відповідні стовбці, де X0 приймає значення 0, 1. Виділяємо значення функції F11 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 3.2(показано перші 12)

Таблиця 3.2- F11(X11,…, X0)

X0	**F11**
0	0
1	0
1	1
0	1
1	0
0	1
1	0
0	0
1	0
1	1
0	1
0	0

Провівши порозрядний XOR F11(X11, …, X1, 0) та F11(X11, …, X1, 1) отримаємо наступні значення, які представленні у таблиці 3.3 та рис. 3.3.

Таблиця 3.3- Порозрядний XOR F11(X11, …, X1, 0) і F11(X11, …, X1, 1)

F11(X11, …, X1, 0)	F11(X11, …, X1, 1)	XOR
0	0	0
1	1	0
1	0	1
0	0	0
1	0	1
0	1	1

```
*BITWISE OPERATION RESULT OF F11(X11, :, X1, 0) AND F11(X11, :, X1, 1)  :
|Index|Result|
|0||0|
|1||0|
|2||1|
|3||0|
|4||1|
|5||1|
```

Рис. 3.3 – Отриманні результати для порозрядного XOR

Сума значень результатів після порозрядної операції XOR для значень F11(X11, …, X1, 0) та F11(X11, …, X1, 1) дорівнює 3 у прикладі, для всіх ж можливих значень - 1924, рис.3.4. Що означає нелінійну залежність F11 від X0.

```
|2045||1|
|2046||1|
|2047||1|
|2048||1|
*SUM OF BITWISE OPERATION RESULT = 1924
```

Рис. 3.4– Отриманна сума одиниць

Розглянемо залежність F10 функції від X1. Виділяємо відповідні стовбці, де X1 приймає значення 0, 1. Виділяємо значення функції F10 для цих стовбців (окремо для 0 і 1). Наступні дії наведені в табл. 3.4(показано перші 12)

Таблиця 3.4- F10(X11,…, X0)

X1	**F10**
0	1
0	1
1	1
0	1
1	1
0	1
1	1
0	0
1	0
1	1
1	1
0	1

Провівши порозрядний XOR F10(X11, …, 0, X0) та F10(X11, …, 1, X0) отримаємо наступні значення, які представленні у таблиці 3.5 та рис. 3.5.

Таблиця 3.5- Порозрядний XOR F10(X11, …, 0, X0) та F10(X11, …, 1, X0)

F10(X11, …, 0, X1)	F11(X11, …, 1, X1)	XOR
1	1	0
1	1	0
1	1	0
1	0	1
0	1	1
1	1	0

```
*BITWISE OPERATION RESULT OF F10(X11, :, 0, X0) AND F10(X11, :, 1, X0)  :
|Index|Result|
|0||0|
|1||0|
|2||0|
|3||1|
|4||1|
|5||0|
```

Рис. 3.5 – Отриманні результати для порозрядного XOR

Сума значень результатів після порозрядної операції XOR для значень F10(X11, …, 0, X0) та F10(X11, …, 1, X0) дорівнює 2 у прикладі, для всіх ж можливих значень - 1648, рис.3.6. Що означає нелінійну залежність F10 від X1.

```
|2048||1|
*SUM OF BITWISE OPERATION RESULT = 1648
```

Рис. 3.6– Отриманна сума одиниць

Проводячи аналогічні операції для кожної змінної та відповідної функції отримаємо результуючу таблицю:

Таблиця 3.6- Загальні значення суми одиниць

	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
X0	1844	1968	1994	1648	1860	1688	1960	1962	1800	1846	1640	1924
X1	1868	1984	1912	1914	1986	1984	1828	1844	1680	1824	1648	1968
X2	1912	1988	1914	1834	1860	1960	1844	1802	1848	1896	1960	1964
X3	1812	1900	1680	1626	1712	1992	1846	1854	1840	1680	1944	1966
X4	1912	1892	1946	2024	2026	1988	1902	1802	1966	1684	1968	1800
X5	1818	1902	1802	1660	1996	1914	1900	1840	1926	1844	1836	1996
X6	1800	1988	1650	1658	1864	1986	1998	1988	1968	1966	1872	1846
X7	1994	1966	1998	1860	1600	1684	1996	1994	1642	1962	1844	1924
X8	1600	1912	1812	1814	1804	1984	1694	1846	1972	1824	1862	1936
X9	1712	1814	1804	1900	1806	1602	1998	1866	1980	1844	1842	1644
X10	1968	1964	1800	1812	1996	1982	1640	1642	1888	1860	1648	1648
X11	1892	1854	1688	1998	1684	1842	1840	1996	1648	1640	1626	1840

Якщо сума одиниць дорівнєю нулю, то функція зовсім не залежить від значення змінної взагалі, якщо дорівнює 2048 – то залежить лінійно. Якщо ж сума одиниць відрізняється від крайніх значень, це говорить про не лінійну залежність.[12]

Висновки до третього розділу

У даному розділі було описано технічні особливості розробленого програмного додатку, та описано структурні одиниці програми, з показаними їхніми результатами, експереминтально було перевірено теоретичні дослідження з попереднього розділу.

Висновки

Для того щоб забезпечити необхідність високодії швидкодії процесу шифрування під час передачі даних у реальному часі, можна використовувати алгоритм шифрування на основі підстановок довільної розрядності. Перші ніж алгоритм буде вважатись надійним та може стати стандартом на нацьональному та міжнародному рівні, проходять роки різноманітних досліджень та публікацій. У даній роботі були закладені основи лінійного криптоааналізу даного алгоритму.

Були встановленні умови, при яких алгоритм матиме залежність усіх розрядів криптограми від усіх значень змінних вхідних даних, чого немає у сучасних блокових алгоритмах шифрування. Практичне використання даних умов дозволяє говорити про підвищену стійкість від криптоаналітичних атак. Була сформована практична частина, у якій теоретичне дослідження було перевірено експерементально.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Rechberger C. Biclique Cryptanalysis of the Full AES [Електронний ресурс] / Christian Rechberger. – 2013. – Режим доступу до ресурсу: https://www.webcitation.org/68GTcKdoD?url=http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf.

2. United States National Institute of Standards and Technology (NIST). Announcing the ADVANCED ENCRYPTION STANDARD (AES) [Електронний ресурс] / United States National Institute of Standards and Technology (NIST). – 2012. – Режим доступу до ресурсу: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

3. Diffie W. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard [Електронний ресурс] / W. Diffie. – 2014. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/1646525.

4. Davies D. W. Security for computer networks / D. W. Davies., 1989. – (2).

5. Somitra Kumar S. Single Key Recovery Attacks on 9-Round Kalyna-128/256 and Kalyna-256/512 [Електронний ресурс] / Sanadhya Somitra Kumar. – 2016. – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-30840-1_8.

6. YOUSSEF A. A Meet-in-the-Middle Attack on Reduced-Round Kalyna-b/2b [Електронний ресурс] / Amr M. YOUSSEF. – 2016. – Режим доступу до ресурсу: http://search.ieice.org/bin/summary.php?id=e99-d_4_1246.

7. Hellman M. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard" [Електронний ресурс] / Martin Hellman. – 1977. – Режим доступу до ресурсу: https://web.archive.org/web/20140226205104/http://origin-www.computer.org/csdl/mags/co/1977/06/01646525.pdf.

8. Невмержицький П.А., Тесленко О.К. "Алгоритм шифрування, який ґрунтується на суперпозиції підстановок із найпростіших регулярних ОККМ" Свідоцтво про реєстрацію авторського права на твір. № 66618 Дата реєстрації 13.07.2016

9. Matsui M. A new method for known plaintext attack of FEAL cipher / Matsui., 1992.

10. Matsui M. Linear cryptanalysis method for DES cipher [Електронний ресурс] / Matsui – Режим доступу до ресурсу: https://web.archive.org/web/20070926205624/http://homes.esat.kuleuven.be/~abiryuko/Cryptan/matsui_des.PDF.

11. Seberry J. Advances in cryptology [Електронний ресурс] / J. Seberry, Y. Zheng. – 1992. – Режим доступу до ресурсу: https://www.worldcat.org/title/advances-in-cryptology-auscrypt-92-workshop-on-the-theory-and-application-of-cryptographic-techniques-gold-coast-queensland-australia-december-13-16-1992-proceedings/oclc/29186866.

12. Тесленко О. К. Метод попередньої підготовки аналізу криптостійкості для підстановок довільної розрядності [Електронний ресурс] / О. К. Тесленко, Я. Ю. Чабан – Режим доступу до ресурсу: http://old.nuft.edu.ua/page/51adaed39c2a2/files/IK5.pdf.