Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Диференційні властивості блокового симетричного шифру з керованими підстановками
(тема)

Виконав: Васильєв В.О.
(прізвище, ініціали)
студент 2 курсу, групи БІКСм-18-1
Спеціальність 125 Кібербезпека
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва освітньої програми)

Керівник д.т.н., проф. Лисицька І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Халімов Г.З.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 125 Кібербезпека
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)

Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові *Васильєву Василю Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Диференційні властивості блокового симетричного шифру з керованими підстановками*
затверджена наказом по університету від "04" листопада 2019 р. № 1649Ст
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи *Теоретичні данні про БСШ ШУП*
4. Перелік питань, що потрібно опрацювати в роботі
1. Блокові симетричні шифри та перспективи їх використання у сучасності
2. Сучасні методи криптоаналізу
3. Блоковий симетричний шифр ШУП та особливості його побудови
4. Методика дослідження лінійних та диференціальних властивостей
5. Опис програмної реалізації та постанова експериментів
6. Результати досліджень
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *Презентаційний матеріал у вигляді слайдів*
6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
1. Сучасні методи криптоаналізу	д.т.н., проф. Долгов В.І.		
2. Блоковий симетричний шифр ШУП та особливості його побудови	д.т.н., проф. Долгов В.І.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Отримання завдання*	*2.09.19*	
2	*Пошук літератури*	*3.09.19-19.09.19*	
3	*Аналіз зібраних даних*	*20.09.19-19.10.19*	
4	*Розробка алгоритму та його програмної реалізації для дослідження диференційних та лінійних показників БСШ ШУП*	*20.10.19-10.11.19*	
5	*Аналіз отриманих результатів*	*11.11.19-30.11.19*	
6	*Оформлення пояснювальної записки*	*01.11.19-12.11.19*	

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи (проекту) _____ __д.т.н., проф. Лисицька І.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської атестаційної роботі містить 83 с., 5 табл., 10 рис., 21 джерел, 3 додатки.

У роботі проведено дослідження можливості використання випадкових таблиць підстановки в блоковому симетричному шифрі ШУП («шифра с управляемыми подстановками»). Для цього була розроблена скорочена програмна реалізація (зі зменшеною кількістю раундів шифрування з вісьмох до чотирьох) шифру ШУП-256 на мові програмування С++. Програма дозволяє зашифровувати заданий масив текстів (різниць) на основі випадкової та стандартної таблиць підстановок, записати проміжний результат на кожному раунді, побудувати таблиці різниць диференціалів та таблиці лінійної апроксимації для кожного циклу шифрування та знайти максимуми цих таблиць. Аналіз отриманих даних дозволяє порівняти максимуми для шифру з різними таблицями підстановки та зробити висновки щодо можливості використання випадкових таблиць підстановок як в шифрі ШУП, так і в інших блокових симетричних шифрах.

БЛОКОВИЙ СИМЕТРИЧНИЙ ШИФР, ТАБЛИЦЯ РІЗНИЦЬ ДИФЕРЕНЦІАЛІВ, ТАБЛИЦЯ ЛІНІЙНОЇ АПРОКСИМАЦІЇ, ШУП, ВИПАДКОВІ ТАБЛИЦІ ПІДСТАНОВКИ, ДИФЕРЕНЦІАЛЬНІ ВЛАСТИВОСТІ, ЛІНІЙНІ ВЛАСТИВОСТІ.

РЕФЕРАТ

Пояснительная записка к магистерской аттестационной работе содержит 83 с., 5 табл., 10 рис., 21 источников, 3 приложения.

В работе проведено исследование возможности использования случайных таблиц подстановок в блочном симметричном шифре ШУП («шифр с управляемыми подстановками») [12]. Для этого была разработана сокращенная программная реализация (с уменьшенным количеством раундов шифрования с восьми до четырех) шифра ШУП-256 на языке программирования C++. Программа позволяет зашифровать заданный массив текстов (разностей) на основе случайной и стандартной таблиц подстановок, записать промежуточный результат на каждом раунде, построить таблицы разностей дифференциалов и таблицы линейной аппроксимации для каждого цикла шифрования и найти максимумы этих таблиц. Анализ полученных данных позволяет сравнить максимумы для шифра с различными таблицами подстановок и сделать выводы о возможности использования случайных таблиц подстановок как в шифре ШУП, так и в других блочных симметричных шифрах.

БЛОКОВЫЙ СИММЕТРИЧНЫЙ ШИФР, ТАБЛИЦА РАЗНИЦ ДИФФЕРЕНЦИАЛОВ, ТАБЛИЦА ЛИНЕЙНОЙ АППРОКСИМАЦИИ, ШУП, СЛУЧАЙНЫЕ ТАБЛИЦЫ ПОДСТАНОВКИ, ДИФФЕРЕНЦИАЛЬНЫЕ СВОЙСТВА, ЛИНЕЙНЫЕ СВОЙСТВА.

THE ABSTRACT

The magister’s attestation work contain 83 p., 5 tabl., 10 pic., 21 sources, 3 supplements.

The paper investigates the possibility of using random tables of substitutions in a block symmetric cipher SHUP ("cipher with controlled substitutions"). For this purpose, was developed shortened software implementation (with a reduced number of encryption rounds from eight to four) of the SHUP-256 cipher in the C++ programming language. The program allows to encrypt a given array of texts (differences) based on random and standard substitution tables, record the intermediate result at each round, build differential difference tables and linear approximation tables for each encryption cycle and find the maxima of these tables. The analysis of the obtained data allows us to compare the maxima for the cipher with different substitution tables and draw conclusions about the possibility of using random substitution tables both in the SHUP cipher and in other block symmetric ciphers.

BLOCK SYMMETRIC CIPHER, DIFFERENTIAL DIFFERENCE TABLE, LINEAR APPROXIMATION TABLE, SHUP, RANDOM SUBSTITUTION TABLES, DIFFERENTIAL PROPERTIES, LINEAR PROPERTIES.

ЗМІСТ

ПОЗНАЧЕННЯ ТА СКОРОЧЕННЯ

БСШ – блочний симетричний шифр

ДСТУ – державний стандарт України

ЕОМ – електронна обчислювальна машина

ЕЦП – електронний цифровий підпис

ШУП – шифр с управляемыми подстановками

AES – Advanced Encryption Standard

CBC – Cipher Block Chaining

DDT – Differential Difference Table

DES – Data Encryption Standard

FEAL – Fast data Encripherment Algorithm

LAT – Linear Approximation Table

MAC – message authentication code

MDC – manipulation detection code

NESSIE – New European Schemes for Signatures, Integrity, and Encryptions

RSA – Rivest-Shamir-Adleman

SHA – Secure Hash Algorithm

SKIP – Simple Key-Management for Internet Protocol

ВСТУП

Відкриті канали зв’язку в наш час займають вагоме місце в повсякденному житті людства. Яскравим прикладом є глобальна мережа Internet. Люди все частіше використовують її для спілкування або для проведення різних операцій, в тому числі й економічних. Однак слід пам’ятати, що при передачі інформації відкритими каналами зв’язку вони можуть бути перехоплені зловмисником, який таким чином отримає доступ до конфіденційних даних. Питання забезпечення цілісності та конфіденційності інформації стає все більш актуальним, тому до стандартів забезпечення захисту висуваються жорсткі вимоги.

Один з провідних сучасних криптографів Брюс Шнайер сказав: «Безпека – це процес, а не результат». На сьогодні цей «процес» є однією з основних складових частин розвитку країн світу. Інформаційна сфера відіграє зростаючу роль у підтримці безпеки всіх об'єктів життєдіяльності суспільства, а також в житті людини, чим пояснюється той факт, що захист конфіденційності інформації є найважливішим напрямком діяльності держави. Немає жодної економічно розвиненої країни, в якій би не було свого власного стандарту шифрування або цифрового підпису. Для цього проводяться великомасштабні міжнародні конкурси, в яких приймають участь вчені, криптографи, математики зі всього світу. Серед таких конкурсів можна відмітити європейський конкурс NESSIE, американський конкурс AES, результатом котрого стало відкриття одного з найбільш широко застосовуваних та відомих алгоритмів шифрування сучасності Rijndael. Для алгоритмів хешування схожим конкурсом є SHA-3, на якому був прийнятий новий стандарт хешування Keccak (або SHA-3).

Наша країна в цій гонці «озброєнь» надійними та сучасними засобами забезпечення захисту інформації також бере активну участь. Так, у 2002 році в Україні був прийнятий національний стандарт електронного цифрового підпису

ДСТУ 4145. У 2014 році Україна розробила власний національний стандарт блочного симетричного шифрування ДСТУ 7624 («Калина») [1].

Однак дослідження в цій сфері ще тривають, і, враховуючи, що криптографія на сьогодні динамічно розвивається, необхідно, щоб стандарт шифрування був найбільш надійним та відповідав сучасним досягненням в області криптографії. У зв'язку з цим, очевидним фактом є необхідність проведення подальших детальних досліджень сучасних алгоритмів шифрування та їх криптографічних властивостей. Особливої уваги заслуговує оцінка стійкості шифрів до атак криптоаналізу. Процедура визначення ефективності засобів захисту найчастіше є більш трудомісткою, ніж їх розробка, та вимагає наявності спеціальних знань і, звичайно, більш високої кваліфікації, ніж задача розробки самого шифру.

На сьогодні існує безліч методик оцінки стійкості шифрів. Більшість з них ґрунтується на оцінці показників стійкості до атак лінійного та диференційного криптоаналізу.

Метою даної атестаційної дипломної роботи є дослідження можливості використання випадкових таблиць підстановки в БСШ ШУП («шифр с управляемыми подстановками»), а також їх вплив на диференційні та лінійні властивості шифрів. Методом оцінки цих властивостей є порівняння максимумів таблиць різниць диференціалів та таблиць лінійних апроксимацій. Для побудови таблиць використовується програмна реалізації блокового симетричного шифру ШУП.

Результати роботи (а також побічні висновки) були оформлені у вигляді статей та опубліковані в науковому журналі «Прикладна радіоелектроніка» (див. додаток В).

1 БЛОКОВІ СИМЕТРИЧНІ ШИФРИ ТА ПЕРСПЕКТИВИ ЇХ ВИКОРИСТАННЯ У СУЧАСНОСТІ

Людство використовує шифрування для приховування інформації від сторонніх очей вже протягом десятка століть. Цей метод перетворення різних даних пройшов довгий шлях, на якому еволюціонував та розвивався. І не дивлячись на то, що зазвичай шифрування використовувалося лише у військовій справі, у сучасному світі воно повсюдно увійшло в наше життя. Криптографія дозволяє здійснювати ділові операції, використовуючи методи ідентифікації та автентифікації, електронні цифрові підписи, здійснювати різні транзакції через мережу Інтернет і багато іншого. Шифрування в цивільному секторі ведеться для виконання міжнародних банківських операцій, електронного обміну інформацією, обміну електронною поштою та комерційних угод по мережах зв'язку.

Основним механізмом захисту інформації, що застосовується практично у всіх сферах криптографії, є блочне симетричне шифрування.

Блочний симетричний шифр – це симетричний шифр (тобто для шифрування та розшифрування використовується один і той же ключ), який розділяє текст на блоки розміром 64-256 біт та працює з кожним блоком окремо. Деякі сучасні шифри обробляють блоки довжиною 512 та 1024 бітів. Такий підхід дозволяє при використанні одного ключа відносно невеликого розміру шифрувати текст, що значно більший за ключ. Якщо розмір тексту менше за розмір блока, то цей блок доповнюється за певним алгоритмом.

Блочні симетричні шифри мають ряд переваг: швидкість шифрування інформації, простота застосовуваних операцій, відносно невеликий розмір ключа, схожі функції шифрування та розшифрування, гнучкість [2]. Ці переваги дозволяють використовувати БСШ у повсякденному житті для забезпечення практично всіх критеріїв безпеки (безпосередньо або побічно, наприклад, в

алгоритмі електронного цифрового підпису або алгоритмі автентифікації): конфіденційності, цілісності, спостереженості.

1.1 БСШ як засіб забезпечення конфіденційності інформації

Використання комп'ютерних технологій майже у всіх сферах людської діяльності змусило задуматися про питання захисту конфіденційної інформації. Головним чином це зв'язано з тим, що більша частина документообігу перейшла в нове середовище – електронне. Враховуючи вартість даних в сучасному світі та можливі наслідки її витоку, необхідність розвитку криптографічних методів захисту інформації вийшла на перший план державних пріоритетів.

Багаторічна практика використання криптографічних засобів захисту підтвердила, що найбільш прийнятним способом досягнення конфіденційності інформації є використання БСШ.

На сьогоднішній день всі інформаційно розвинуті країни мають свій власний національний стандарт блочного шифрування, і Україна з недавніх пір не є винятком. Проте цей факт не зменшує необхідності та актуальності досліджень у цій сфері для наших науковців, так як створення більш перспективних блочних симетричних шифрів може дозволити вийти криптографії України на новий рівень.

Для забезпечення безпеки даних до алгоритмів БСШ висувається ряд вимог [3]:

- високий рівень захисту інформації проти дешифрування та можливих модифікацій;
- захищеність даних повинна ґрунтуватися лише на знанні ключа та не залежати від того, чи відомий зловмиснику алгоритм (правило Кіркхоффа);
- невелика зміна вхідного тексту або ключа повинна приводити до значної зміни зашифрованого тексту (тобто забезпечувати «лавинний ефект»);

- потужність множини значень ключа повинна не допускати можливості дешифрування методами грубої сили;
- економічність реалізації алгоритму при достатній швидкодії;
- вартість криптоаналізу даних при відсутності ключа повинна значно перевищувати вартість цих даних.

Найбільш широко використовуваним на сьогоднішній день алгоритмом БСШ є AES (Advanced Encryption Standard або Rijndael) – національний стандарт блочного шифрування США, який став переможцем однойменного конкурсу в 2000 році.

Ще одним підтвердженням актуальності БСШ в області захисту конфіденційності інформації є той факт, що алгоритм Rijndael підтримується процесорами компанії Intel, що дає можливість кожному користувачу скористатися криптографічними перетвореннями для забезпечення конфіденційності своєї інформації.

Слід також зазначити, що алгоритм БСШ Rijndael також мав успіх і на європейському конкурсі NESSIE (New European Schemes for Signature Integrity and Encryption). Він став переможцем цього конкурсу разом з шифрами MISTY, Camellia та SHACAL-2.

Однак розробка нових алгоритмів БСШ не завершується з появою шифру Rijndael. У 2008 році американський криптограф Брюс Шнайер запропонував перспективний блочний шифр Threefish, головною особливістю якого є можливість шифрування блоків розміром 1024 біта (при максимумі в 256 біт у AES). Даний шифр знайшов своє застосування в алгоритмі хешування Skein.

Розвиток БСШ не стоїть на місці, а їх актуальність в питаннях забезпечення конфіденційності поки що незаперечна, тому найближчим часом можна очікувати ще більш вражаючих результатів.

1.2 БСШ як засіб забезпечення цілісності інформації

При передачі інформації по мережах зв'язку або під час її зберіганні на магнітних носіях електронної обчислювальної машини не завжди є можливість повністю обмежити фізичний доступ до даних. Опинившись за межами контрольованої зони, цифрова інформація може бути навмисно або випадково пошкоджена. І так як запобігти внесення несанкціонованих змін в реальній системі обробки інформації не представляється можливим, вкрай важливим є необхідність своєчасного виявлення факту такого псування. В цьому випадку власні втрати будуть найменшими та обмежаться лише вартістю «порожній» передачі або зберігання хибної інформації, що, звичайно, у всіх реальних ситуаціях незмірно менше можливого збитку від застосування таких даних.

На сьогоднішній день відомі два підходи для подолання проблеми несанкціонованої зміни даних [4]:

- вироблення MAC (код автентифікації повідомлення);
- вироблення MDC (код виявлення маніпуляцій).

В літературі MAC іноді називають не зовсім коректно, а саме криптографічною контрольною сумою, але більш правильно називати її криптографічною контрольною комбінацією. Вироблення MAC засновано на підрахунку контрольної комбінації за допомогою певного блокового шифру та застосуванням секретного ключа. Важливо також зазначити, що на основі БСШ можна створити алгоритм обчислення MAC для даних довільного розміру. Такий підхід до автентифікації повідомлень з використанням криптографічних перетворень даних є загальновизнаним та закріплений практично у всіх криптографічних стандартах. Наприклад, імітовставка, що передбачена в російському стандарті шифрування ГОСТ 28147-89, є типовим зразком MAC.

Основною перевагою підходу забезпечення цілісності даних за допомогою блокових шифрів є те, що зловмисник не має можливості обчислити MAC для обраного (сфабрикованого) їм повідомлення, тому MAC може передаватися по будь-яким каналах (в тому числі й по відкритим).

Рівняння для обчислення контрольної комбінації виглядає наступним чином:

$$C = C_k(T) = E_k\left(T_1 \oplus E_k\left(T_2 \oplus E_k(\dots \oplus E_k(T_m))\right)\right) \quad (1.1)$$

де C – контрольна сума, T – вхідний текст, E – шифрування обраним алгоритмом.

Схема застосування криптографічного перетворення E_K для вироблення MAC представлена нижче (рис 1.1).

Вхідними значеннями цього алгоритму є масив даних T, який розбивається на m блоків фіксованого розміру, що дорівнює розміру блоку даних застосовуваного шифру (для основних найбільш відомих шифрів – 128 біта або 256 бітів), $T = (T_1, T_2, \dots, T_m)$. В випадках, коли останній блок даних T_m має менший розмір, він доповнюється деяким способом до повного блоку даних. На початку MAC приймає нульове значення та на кожному кроці виконує побітове додавання поточного блоку даних T_i з поточним значенням MAC. Після цього до отриманих даних застосовується перетворенню за обраним алгоритмом зашифрування. В результаті отримуємо нов. значення MAC. Ця процедура виконується послідовно для кожного наступного блоку даних строго в порядку їх розташування. Результатом роботи алгоритму є MAC для повного вхідного масиву інформації [5].

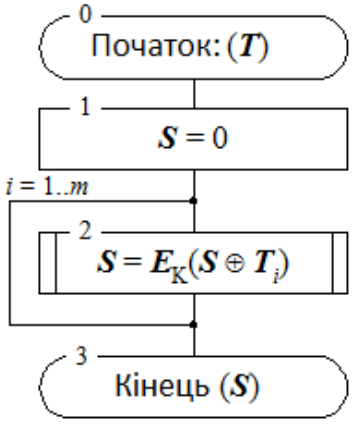

Рисунок 1.1 – Схема використання криптографічного перетворення E_K

Для використання кодів автентифікації повідомлення потрібно мати секретний ключ, а це значить, що використання MAC має ті ж самі недоліки, що й використання блочних симетричних шифрів – процедура потребує попереднього розподілу ключів між всіма учасниками інформаційного обміну або передачу ключа по закритим каналам зв’язку.

1.3 Використання БСШ в гібридних криптосистемах

Як вже було сказано вище, блочні симетричні шифри можуть застосовуватися не тільки в якості самостійного методу захисту даних, але й використовуватися в складі деяких криптографічних алгоритмів та взаємодіяти з іншими механізмами забезпечення безпеки. На фоні всіх своїх переваг БСШ володіють одним з найвагоміших недоліків – проблема розповсюдження ключів. Передавати секретний ключ можна або надійним закритим каналом зв’язку, або у зашифрованому виді. На сьогодні паралельно з симетричною криптографією розвивається так само й асиметричні методи шифрування, які дозволяють нівелювати цей недолік.

Асиметричні методи шифрування стали дуже корисними в підтримці класичних симетричних механізмів, вирішуючи проблему розповсюдження ключів, з якою раніше доводилося миритися (історично сформована ідеологія використання БСШ базувалася на поширення секретного ключа по надійно захищеним каналах зв'язку). Завдяки цьому стало можливим розробити технології, що дозволяють пересилати ключові дані по тим же каналам зв'язку, за якими здійснюється передача основного масиву даних, що, безумовно, підтримало інтерес у розвитку та використанні симетричних шифрів й дало новий імпульс для їх розповсюдження в багатьох існуючих та розроблюваних криптографічних системах захисту інформації. Яскравим прикладом такого ефективного симбіозу асиметричного та симетричного механізмів захисту даних став протокол SKIP, який є представником сімейства протоколів TCP/IP.

На сьогоднішній день ідеї протоколу SKIP реалізовані в деяких стандартах, наприклад [6], у вигляді гібридних шифрів, в яких "асиметричні шифри застосовуються для зашифрування секретного ключа, який використовується для зашифрування фактичного повідомлення з застосуванням симетричних криптографічних методів" (рис. 1.2).

Більшість гібридних криптосистем працюють наступним чином. Для симетричного алгоритму (IDEA NXT, 3DES, AES та інші) виробляється випадковий сеансів ключ. Такий ключ, в залежності від алгоритму, може мати розмір від 128 до 512 біт.

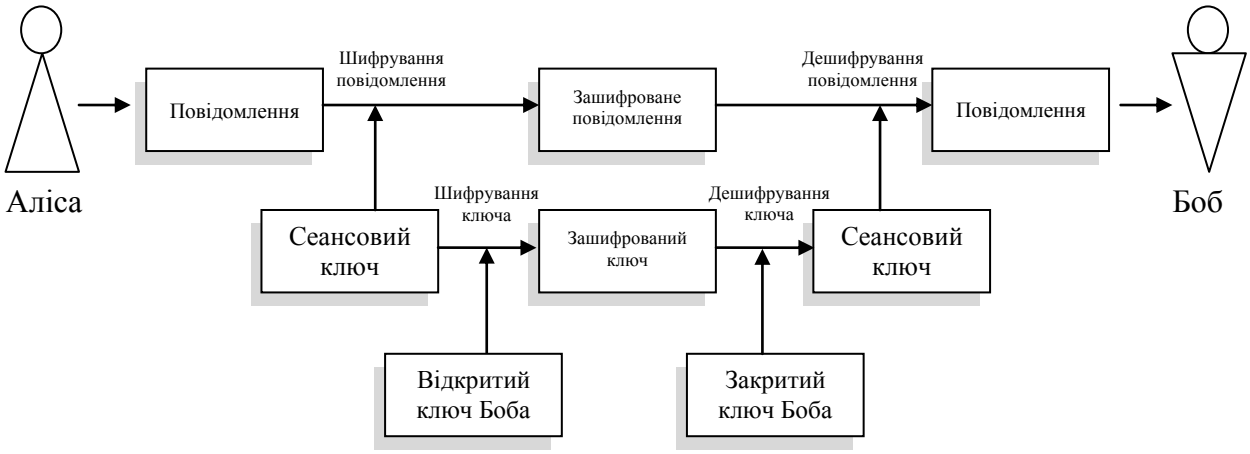

Рисунок 1.2 – Схема шифрування повідомлення з використанням гібридних криптосистем

Далі застосовується симетричний алгоритм шифрування повідомлення. У разі блочного шифрування також необхідно обрати режим шифрування, який дозволяє шифрувати повідомлення з довжиною, яка перевищує довжину блоку (наприклад, режим шифрування CBC). На наступному кроці використовується асиметричний алгоритм шифрування з відкритим ключем (RSA або алгоритм Діффі – Хеллмана), який застосовується для шифрування випадкового сеансового ключа симетричного алгоритму за допомогою відкритого ключа отримувача повідомлення [6]. В результаті одержувач розшифровує ключ за допомогою свого секретного ключа, та з його допомогою (розшифрованого ключа) отримує вхідне повідомлення.

Застосування такого підходу шифрування даних засноване на тому, що використання окремо асиметричного шифрування для всього повідомлення – задача обчислювально більш складніша, ніж реалізація даної системи. На шифрування ж секретного ключа симетричного алгоритму йде порівняно набагато менше часу та ресурсів, що дає можливість усунути недоліки блокових симетричних шифрів та застосовувати його для зашифрування повідомлення. Така технологія дозволяє з регульованим ступенем надійності захищати трафік всіх користувачів й прикладних систем в умовах повної прозорості (непомітності) засобів захисту інформації.

Що ж стосується протоколу SKIP, самі його розробники рекомендують шифрувати пакети з застосуванням симетричних шифрів, так як цю операцію необхідно виконувати з великою швидкодією (вузол мережі повинен для підтримання режиму обміну даних в реальному масштабі часу обробляти (шифрувати та дешифрувати) велике число транзакцій за достатньо стислі часові терміни). На сьогоднішній день це поки що залишається можливим тільки при використанні симетричних алгоритмів.

1.4 Перспективи розвитку БСШ

Основним напрямком розвитку блочних симетричних шифрів, безумовно, є збільшення стійкості алгоритмів до різних типів атак. Цього можна досягнути шляхом ускладнення алгоритму застосуванням ємних, з обчислюваної точки зору, операцій. Але такий метод не представляється можливим, бо одним з головних критеріїв до симетричних шифрів є прозорість та простота алгоритму, що досягається при використання операцій, виконання яких відбувається за один такт.

Вважається, що в майбутніх розробках та пропозиціях конструкції перспективних шифрів будуть засновуватися на використанні вже перевіреної часом технології багатоциклового (багаторазового) повторення комбінації лінійного та нелінійного перетворень. Конкуренцію сучасним шифрам можуть

скласти композиції перетворень, які мають більш високі криптографічні показники та, зокрема, композиції, які забезпечують (при збереженні високої швидкодії) досягнення багаторазовими шифрувальними перетвореннями властивостей випадкової підстановки при меншому числі циклів зашифрування.

На даний момент часу еталоном серед БСШ можна вважати федеральний стандарт США, тобто AES, тому можна вважати, що майбутнє стоїть за шифрами, які зможуть довести свою надійність при меншому числі раундів зашифрування, ніж у AES (10 раундів для 128-бітного ключа, 12 раундів для 192-бітного ключа, 14 раундів для 256-бітного ключа).

Після всього сказаного вище можна зробити висновок, що симетричні методи шифрування зберігають свою придатність та переваги для використання як в найближчий час, так й в далекій перспективі. Навіть у постквантовий період блокові симетричні шифри (наприклад, AES) будуть зберігати запас стійкості, але це відноситься лише для тих шифрів, які мають розмір ключа 256 біт та більше [3]. Питання їх подальшого вивчення та вдосконалення становлять одне з найважливіших та найактуальніших напрямків розвитку сучасної криптографії.

2 СУЧАСНІ МЕТОДИ КРИПТОАНАЛІЗУ

Криптографічна стійкість – здатність криптографічних алгоритмів протистояти атакам зловмисника на нього. Атакуючий криптографічний алгоритм застосовує методи криптоаналізу. Стійкими алгоритмами вважаються ті, успішна атака для котрих потребує від зловмисника недосяжних обчислювальних ресурсів, значний обсяг перехоплених відкритих та зашифрованих повідомлень або ж потребує стільки часу для розкриття даних, що захищена інформація вже втратить актуальність. Тобто вартість криптоаналізу буде значно перевищувати вартість зашифрованої інформації.

Існують абсолютно стійки до криптоаналізу алгоритми шифрування. Клод Шеннон доказав їх існування опублікував це в роботі [6], де зазначив вимоги до таких систем:

- ключ повинен генеруватися для кожного повідомлення (тобто використовується лише один раз);
- довжина ключа може дорівнювати або бути більше довжини повідомлення;
- ключ є статистично надійним (тобто символи в ключовій послідовності повинні бути випадковими);
- вхідний (відкритий) текст має деяку надлишковість (є критерієм оцінки правильності розшифрування).

Стійкість таких систем не залежить від обчислювальних можливостей зловмисника, а практичне застосування обмежена міркуваннями вартості та зручності користування.

Зазвичай використовуються обчислювально стійкі системи. Стійкість таких систем залежить від того, якими обчислювальними можливостями володіє криптоаналітик. Практична стійкість таких систем базується на теорії

складності й оцінюється виключно на певний момент часу та послідовно з двох позицій:

– обчислювальна складність повного перебору;
– відомі на даний момент вразливості та їх вплив на обчислювальну складність.

У кожному конкретному випадку також існують додаткові критерії оцінки стійкості алгоритму.

Оскільки атака методом грубої сили (повний перебір всіх ключів) можлива для всіх типів криптографічних алгоритмів, для нового алгоритму вона може бути єдиною існуючою [7]. Способи її оцінки засновуються на обчислювальної складності, яка потім може бути конвертована у часовий та грошовий еквівалент, і необхідної продуктивності обчислювальних ресурсів. Така оцінка поки є максимальною та мінімальною одночасно.

Подальший аналіз алгоритмів з метою пошуку вразливостей (криптоаналіз) додає оцінки стійкості по відношенню до відомих криптографічних атак (лінійний криптоаналіз, диференціальний крипто аналіз та більш специфічні) й можуть понизити відому стійкість.

Алгоритм буде вважатися практично стійким, якщо на поточний момент його застосування всі знайдені слабкості й вразливості не знижують стійкості алгоритму нижче сучасних обчислювальних можливостей техніки. Тобто це значить, що жодна з відомих атак не має практичного застосування. Крім того, довжина ключа не повинна бути менше, ніж мінімально допустиме значення, яке зазначене в національних стандартах.

На сьогоднішній день використання методів шифрування є важливим заходом для забезпечення захисту інформації. Необхідно бути впевненим у всіх механізмах захисту інформації в сферах фінансів, для захисту секретних військових даних й державних таємниць. Таким чином, детальні дослідження алгоритмів шифрування для пошуку лазівок, слабких ключів, вразливостей,

перевірка стійкості до відомих криптоаналітичних атак, безумовно, є необхідними.

2.1 Атаки на сучасні БСШ

Атаки на блокові симетричні шифри можуть нести в собі велику небезпеку. Першим чином це обумовлено тим, що ключ шифрування є статичним. Так, отримавши ключ, зловмисник зможе розшифровувати передані в системі повідомлення або видавати себе за інших авторизованих користувачів.

Нижче (рис. 2.1) наведена класифікація крипто аналітичних атак на блокові симетричні шифри [8].

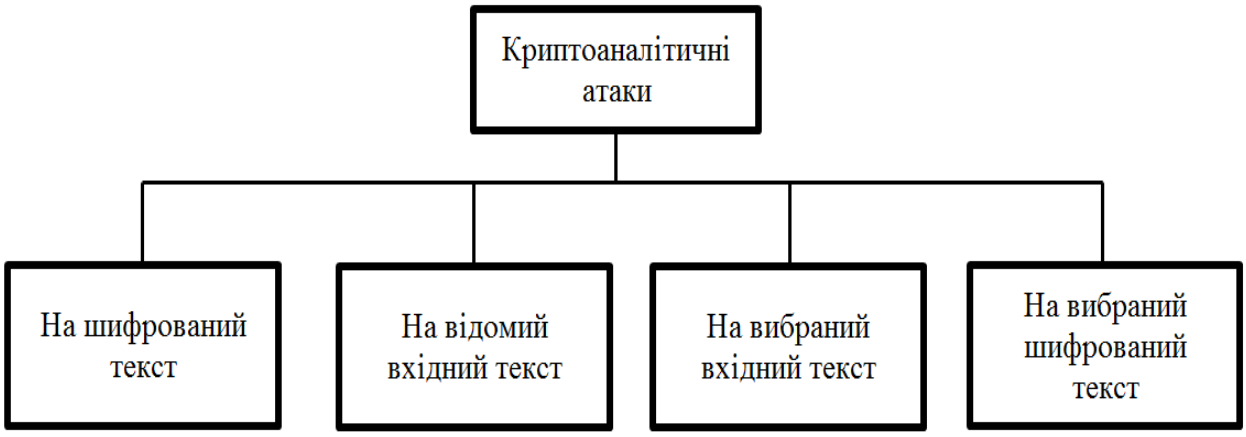

Рисунок 2.1 – Криптоаналітичні атаки на БСШ

При атаці тільки на зашифрований текст зловмисник має доступ тільки до деякого зашифрованого тексту. Він намагається знайти відповідний ключ і вхідний текст. При цьому, згідно до припущення, зловмисник знає алгоритм і може перехопити зашифрований текст. Атака тільки зашифрованого тексту є найімовірнішою, тому що зловмиснику для неї потрібен тільки сам шифротекст. Шифр повинен серйозно перешкоджати цьому типу атаки та не

дозволити дешифрування повідомлення противником. До таких атак відносяться:

- атака методом «грубої сили»;
- статистична атака;
- атака за зразком.

Атака методом «грубої сили» або повного перебору являє собою почергове використання всіх можливих ключів. При цьому вважається, що криптоаналітику є відомим алгоритм шифрування. Щоб запобігти цьому типу атаки, кількість можливих ключів повинна бути дуже великою (128-бітний набір символів дає 2^{128} варіантів можливих ключів і є мінімальним розміром ключа в більшості сучасних БСШ) [9].

Статистичний криптоаналіз виконується наступним чином:

1) за перехопленою криптограмою *E* обчислюється деяка статистика. Ця статистика така, що для всіх осмислених повідомлень *M* вона приймає значення, яке мало відрізняються від *Sk*, величини, що залежить тільки від приватного ключа, який використовується при шифруванні даних;
2) отримана таким чином величина служить для виділення тих можливих ключів, для яких значення *Sk* лежить в близькій межі від значення, що спостерігається.

Криптоаналітик може використовувати атаку за зразком, щоб отримати вхідні данні. Деякі шифри приховують характеристики мови, але створюють деякі статистичні зразки в зашифрованому тексті. Тому важливо використовувати шифри, які зробили б зашифрований текст наскільки це можливо невизначеним і абстрактним.

Атака з вибіркою вхідного тексту подібна атаці на знанні вхідного тексту, але пари "вхідний/зашифрований текст" обираються та виготовляються самим нападником. Найнебезпечнішою із загроз даного типу є застосування методу диференціального криптоаналізу.

Атака з вибором зашифрованого тексту подібна атаці з вибіркою вхідного тексту, за винятком того, що крипто аналітик вибирає певний зашифрований текст і розшифровує його, щоб сформувати пару " вхідний/зашифрований текст".

При атаці, заснованій на знанні вхідного тексту, криптоаналітик має доступ до деяких пар "вхідний/зашифрований текст" на додаток до перехопленого зашифрованого тексту, який він хоче зламати. Пари вхідного/зашифрованого тексту в такому чині збираються заздалегідь. Найнебезпечнішою із загроз даного типу є застосування методу лінійного криптоаналізу.

2.2 Лінійний криптоаналіз БСШ

Лінійний криптоаналіз був винайдений японським криптологом Міцуру Мацуи в 1993 році та вперше був застосований на алгоритмах шифрування DES та FEAL [10]. Згодом лінійний криптоаналіз поширився й на інші алгоритми. Сутність атаки полягає в знаходженні ситуацій, коли сума за модулем 2 окремих бітів відкритого тексту та деяких бітів відповідного йому зашифрованого тексту дорівнює сумі за модулем 2 окремих бітів ключа. Якщо така ситуація виконується з певною ймовірністю $p \neq 1/2$, то існує можливість застосовувати зібрані відкриті тексти й відповідні їм зашифровані тексти для визначення бітів ключа. Таке співвідношення текст/шифротекст/ключ називається лінійною апроксимацією.

$$\left(M_{i_1} \oplus M_{i_2} \oplus \ldots \oplus M_{i_a}\right) \oplus \left(C_{j_1} \oplus C_{j_2} \oplus \ldots \oplus C_{j_b}\right) = K_{k_1} \oplus K_{k_2} \oplus \ldots \oplus K_{k_c}, \quad (2.1)$$

де M_n, C_n, K_n – n-ні біти тексту, шифротексту та ключа відповідно.

Ймовірність виконання цієї рівності (2.1) визначається через ймовірності, які застосовуються при побудові характеристики переходів активних S-блоків згідно з наступною формулою:

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right), \tag{2.2}$$

де n – кількість циклів, p_i – ймовірність переходу на S-блоці, який використовується на i-му циклі [7].

Для виконання атаки лінійного криптоаналізу необхідно також мати деяку кількість пар відкритих/зашифрованих текстів, для яких перевіряється рівність (2.1).

2.3 Диференціальний криптоаналіз БСШ

Диференціальний криптоаналіз був запропонований ізраїльським фахівцями Елі Бихамом та Аді Шаміром в 1990 р.. Атака заснована на вивченні різниць (диференціалів) між шифрованими значеннями на різних циклах шифрування. В якості різниці, як правило, застосовується операція побітового додавання за модулем 2, хоча існують атаки й з обчисленням різниці за модулем 2^{32}. Диференціальний криптоаналіз є статистичною атакою й в результаті виконання пропонує список найбільш ймовірних ключів шифрування блокового симетричного шифру [11].

Для двох підібраних шифротекстів P_1 та P_2 зловмисник обчислює різницю $\Delta P = P_1 \oplus P_2$ й за допомогою отриманого диференціала ΔP намагається визначити диференціал пари текстів C_1 та C_2. У більшості випадків не можливо однозначно визначити ΔC, тому зловмисник обирає таке значення ΔC, яке має більшу ймовірність появи для заданого ΔP. Таким чином криптоаналітик може отримати частину ключа або навіть весь ключ цілком.

Підготовка до лінійного та диференціального криптоаналізу відрізняється тим, що для першого випадку пари «текст/шифротекст» зловмисник генерує самостійно.

2.4 Постановка задач дослідження

В ході виконання атестаційної магістерської роботи ставиться три основні задачі, а саме:

1) програмно дослідити диференціальні та лінійні властивості блокового симетричного шифру ШУП з стандартними таблицями підстановки (таблиці підстановки AES), а також з випадковою таблицею підстановки (додаток А);

2) на прикладі результатів практичних експериментів проаналізувати можливість використання випадкових таблиць підстановки у блоковому симетричному шифрі ШУП;

3) оцінити можливість збільшення мінімального числа активізуємих S-блоків першого циклу за рахунок використання функції надбудови.

Для досліджень використовується програмна реалізації (C++) БСШ ШУП зі скороченою кількістю циклів шифрування (з восьми до чотирьох).

3 БЛОКОВИЙ СИМЕТРИЧНИЙ ШИФР ШУП ТА ОСОБЛИВОСТІ ЙОГО ПОБУДОВИ

Блоковий симетричний шифр ШУП («шифр с управляемыми подстановками») є перспективним ітеративним шифром, який підтримує довжину блока тексту 256 та 512 бітів, а також дозволяє використовувати аналогічні ключі шифрування довжиною 256 та 512 бітів. При розробці шифру був закладений основний критерій щодо мінімізації кількості циклів, за яке шифр приходить до випадкової підстановки, тобто було поставлене завдання досягти стану випадкової підстановки за диференціальними показниками за 2 цикли [12]. Як і для шифру AES, до ШУП ставилися також вимоги до конструкції циклової функції, яка повинна бути простою та прозорою. Далі розглядається 256-бітна версія шифру.

3.1 Процедура шифрування

На вхід процедури подається відкритий текст та секретний ключ. Для кожного раунда шифрування потребується один 256-бітовий (або 512-бітовий) підключ, який отримується шляхом розгортання секретного ключа. Вхідний блок даних обробляється функцією надбудови, після чого переходить до циклової функції. Перед кожним раундом циклової функції до блоку тексту додається за модулем 2 черговий підключ. Усього текст проходить вісім раундів шифрування. В завершенні проводиться фінальна рандомізація за допомогою XOR та додаткового (дев’ятого) підключа. Отриманий у результаті блок даних є шифротекстом.

3.2 Функція надбудови

Функція надбудови дозволяє покращити умови досягнення «лавинного ефекту» за зменшене число циклів шифрування. Досягнення «лавинного ефекту» означає, що зміна малої кількості біт (навіть одного біту) у ключі або у

вхідному тексті приводить до зміни половини вихідних бітів шифротексту (тобто означає залежність вихідних бітів від кожного вхідного біту). Надбудову можна застосовувати як на «чистому» вхідному тексті, так і після додавання підключа з блоком даних.

На вхід цієї функції подається блок тексту, який поділяється на 32-бітові блоки. Після чого, ці блоки розбивають на два однакові за розміром сегменти (рис. 3.1). До першого сегменту додається за модулем 2 відповідно другий сегмент. Далі перший сегмент знов поділяється на два сегменти і другий сегмент додається до першого за модулем 2. Ця процедура (поділу сегменту та складання отриманих напівблоків за модулем 2) виконується ще раз. Таким чином, після проходження цієї функції, результуючий перший 32-бітовий блок, завжди буде змінюватися при зміні будь-якого біту вхідної послідовності, а це в свою чергу приведе к зміні інших 32-бітових блоків під час проходження циклової функції [12].

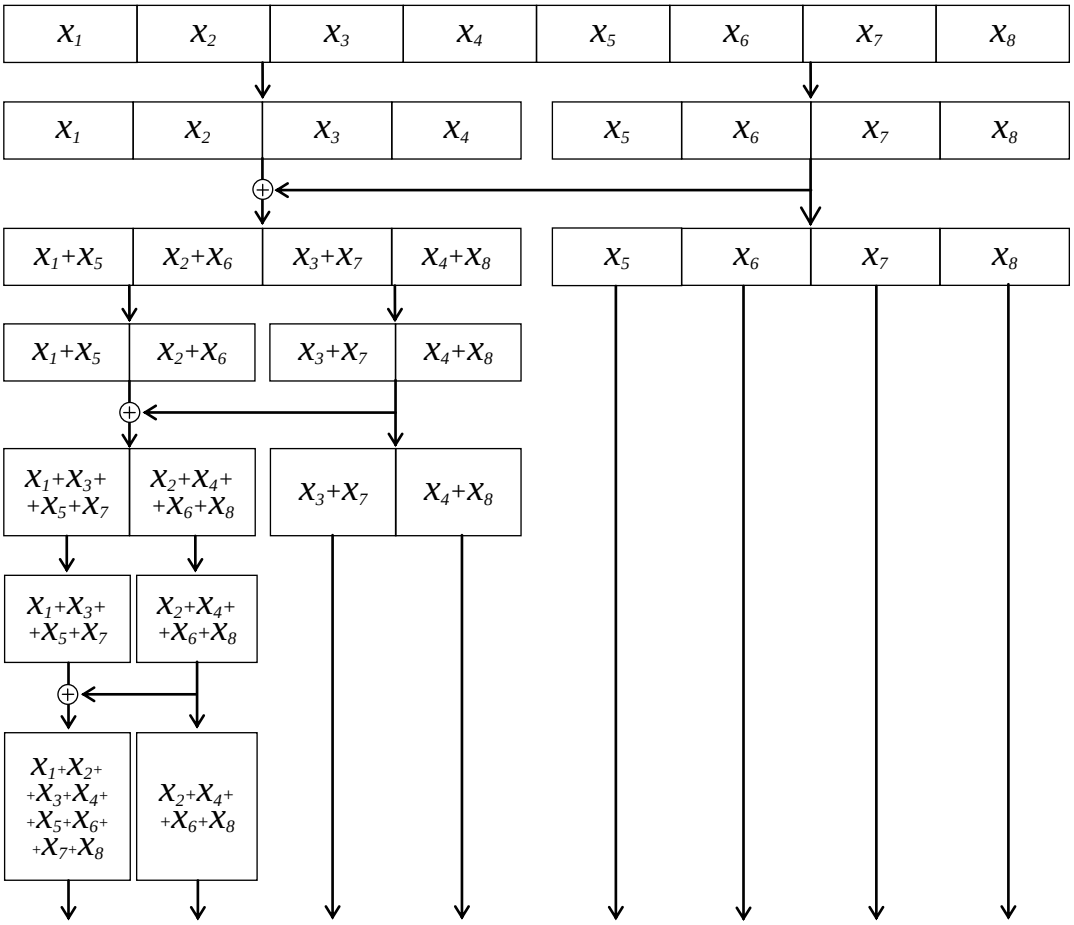

Рисунок 3.1 – Функція надбудови для ШУП-256

3.3 Циклова функція

На вхід циклової функції подається блок даних x, який збігається за розміром з блоком відкритого тексту (256 біт або 512 біт) та підключ K_i того ж розміру, що і блок даних (рис 3.2).

Вхідний блок та підключ розбиваються кожен на вісім рівних по тривалості підблоків по 32 біти. На початку раунду, вхідні підблоки тексту за допомогою XOR додаються до блоків підключа, а далі перший підблок даних проходить SL-перетворення та додається за модулем 2 до наступного підблока, який в свою чергу теж проходить SL-перетворення (після XOR) [13]. На останньому кроці, восьмий підблок додається за модулем 2 до першого. Таким чином, кожен попередній блок активує наступний і разом з функцією надбудови приводить до «лавинного ефекту».

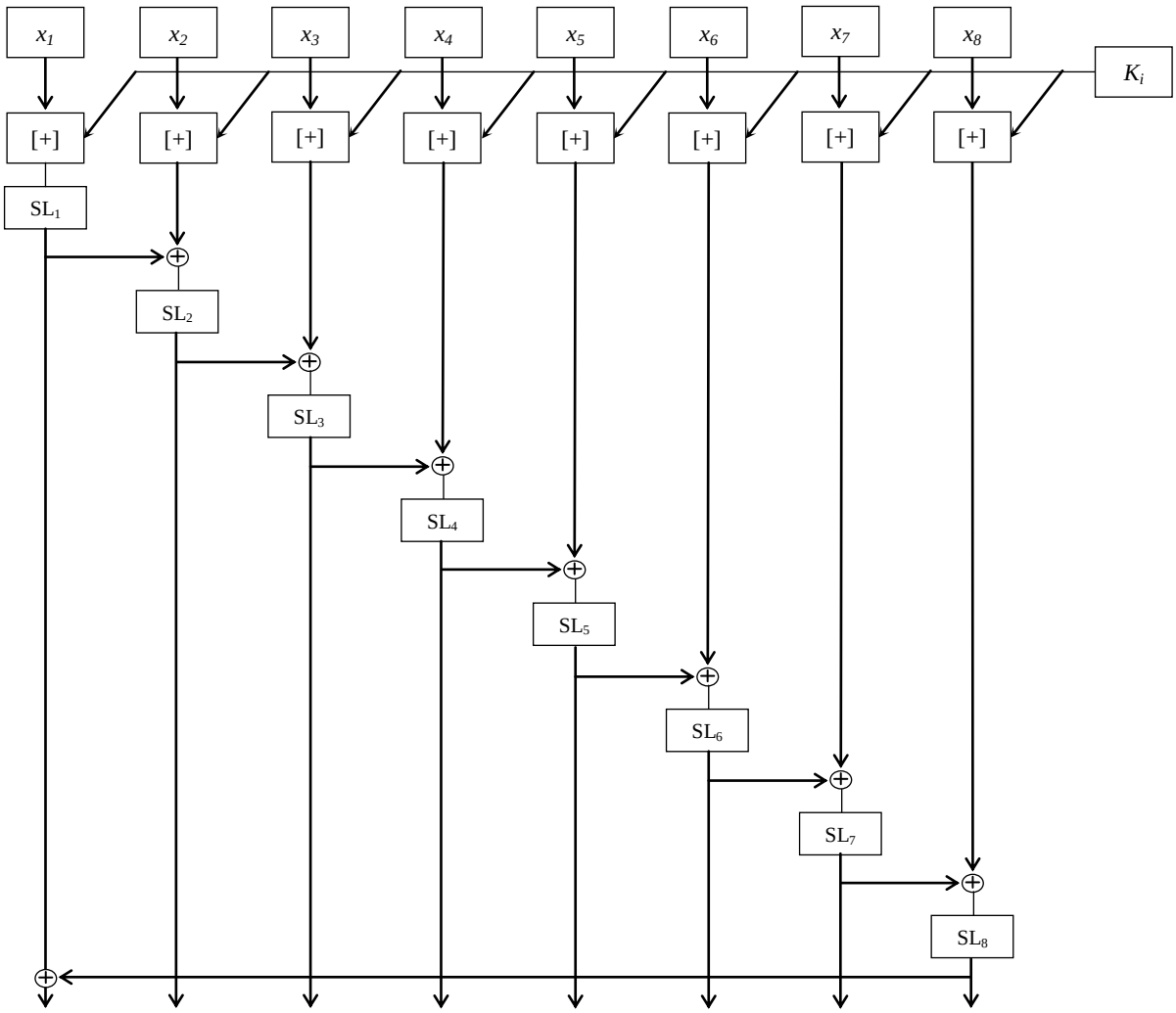

Рисунок 3.2 – Циклова функція ШУП

3.4 SL-перетворення

Основним перетворенням циклової функції є SL-перетворення, яке виконує перетворення 32-бітових блоків даних. Схема SL-перетворення наведена нижче (рис. 3.3).

Вхідне 32-бітове значення ділиться на 4 байта, кожен з яких замінюється відповідно до заданої таблицею підстановки (SybBytes). У перетворенні може використовуватися або одна таблиця на всі чотири байти, або чотири різні таблиці (по одній на кожен байт). В цієї атестаційній роботі також перевіряється можливість використання випадкових таблиць підстановки та вивчається їх вплив на диференціальні та лінійні властивості.

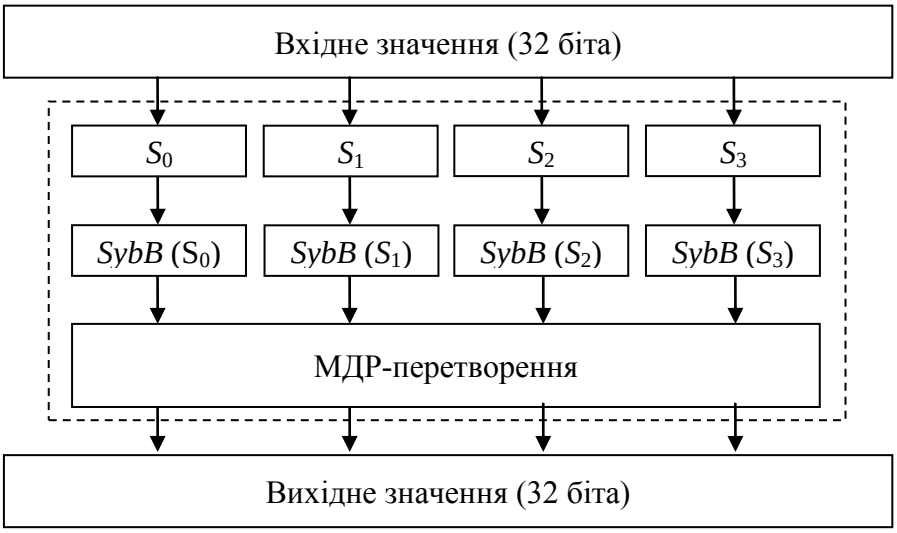

Рисунок 3.3 – Схема SL-перетворення

Після операції заміни в S-блоках 4 байти (a_0,a_1,a_2,a_3) подаються на вхід МДВ-перетворення, яке виконує матричне множення наступного виду[1]:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 02 \cdot a_0 \oplus 03 \cdot a_1 \oplus 01 \cdot a_2 \oplus 01 \cdot a_3 \\ 01 \cdot a_0 \oplus 02 \cdot a_1 \oplus 03 \cdot a_2 \oplus 01 \cdot a_3 \\ 01 \cdot a_0 \oplus 01 \cdot a_1 \oplus 02 \cdot a_2 \oplus 03 \cdot a_3 \\ 03 \cdot a_0 \oplus 01 \cdot a_1 \oplus 01 \cdot a_2 \oplus 02 \cdot a_3 \end{bmatrix} \quad (3.1)$$

[1] МДВ матриця – матриця лінійного коду з максимально допустимою відстанню.

Матриця МДВ-перетворення ШУП співпадає з матрицею перетворення шифру AES [14], але при обчисленні добутку елементів вектора на матричні коефіцієнти в шифрі ШУП використовується інший поліном:

$$m(x) = x^8 + x^4 + x^3 + x^2 + 1 \qquad (3.2)$$

Вихідний 32-бітовий вектор МДВ-перетворення (b_0,b_1,b_2,b_3) є вихідним значенням SL-перетворення.

3.5 Процедура розшифрування

Алгоритм розшифрування є зворотним до алгоритму зашифрування, але з інверсними функціями та таблицями підстановок. На вхід алгоритму подаються блоки зашифрованого тексту та підключі зашифрування у зворотному порядку. На самому початку процедури розшифрування здійснюється зняття фінальної рандомізації шифротексту, після чого отриманий блок даних у зворотному порядку обробляється цикловою функцією. Відповідно до того, як виконувалася надбудова (до XOR-операції підключа з текстом або після), вона виконується і при розшифруванні. Отриманий блок даних є блоком відкритого тексту.

3.6 Процедура розгортання ключів

При розгортанні використовується секретний ключ, в якому відсутнє повторення байтів. Сама процедура розгортання секретного ключа включає його поділ на окремі байти з наступним виконанням циклічного зсуву секретного ключа на величину, що задається значеннями суміжних (сусідніх) байтів, який і використовується як черговий підключ. В цьому випадку для формування всього набору циклових підключів потрібно дев'ять послідовних байтів [12].

4 МЕТОДИКА ДОСЛІДЖЕННЯ ЛІНІЙНИХ ТА ДИФЕРЕНЦІАЛЬНИХ ВЛАСТИВОСТЕЙ

Під вивченням лінійних та диференціальних властивостей шифру в більшості випадків розуміється побудова таблиць лінійних апроксимацій шифрів, таблиць розподілу диференціалів та визначення їх максимумів.

4.1 Лінійна апроксимація

Лінійна апроксимація алгоритму шифрування є основою концепції лінійного криптоаналізу – одного з більш важливих, загальних методів криптоаналізу. Під лінійною апроксимацією мається на увазі лінійне рівняння, яку описує відношення вхідних біт алгоритму щодо вихідних. Рівняння виконується з певною ймовірністю P для випадково обраного входу та відповідного йому виходу. Величина $|\Delta p| = |p - 1\backslash 2|$ являє собою ефективність апроксимації. Апроксимації з ненульовим значенням Δp вважаються ефективними.

У разі ітераційних блокових шифрів (саме цей випадок нас цікавить), розрахунок найбільш ефективної лінійної апроксимації здійснюється зазвичай в 2 етапи. По-перше, обчислюється ефективна лінійна апроксимація однієї ітерації шифру, як результат композиції апроксимацій складових функцій. Далі, як результат композиції апроксимацій послідовних ітерацій, виходить лінійна апроксимація шифру цілком.

Апроксимація алгоритму дозволяє виявити ключові біти для досить великого набору відомих пар відкритого тексту та відповідного йому шифротексту. На відміну від диференціального криптоаналізу, який по суті є атакою з вибором відкритих текстів, лінійний криптоаналіз відноситься до атак з відомим відкритим текстом і до того ж, в певних випадках [15], застосується до атак, що ґрунтується виключно на знанні шифротексту.

Загалом, лінійна апроксимація функції $Y = f(X)$: $\{0,1\}^n \longrightarrow \{0,1\}^m$ визначається як довільне рівняння виду:

$$\bigoplus_{i \in Y'} y_i = \bigoplus_{j \in X'} x_j, \tag{4.1}$$

яке виконується з ймовірністю апроксимації $p = N(X',Y')/2^n$, де $Y' \subseteq \{1,..,m\}$, $X' \subseteq \{1,..,n\}$, а $N(X',Y')$ позначає число пар (X, Y), для яких складається рівняння. Для простоти рівняння (4.1) записується в наступному вигляді:

$$Y[Y'] = X[X'] \tag{4.2}$$

Множини індексів X', Y' називаються вхідними та вихідними масками відповідно, а функція $N(X',Y')$ називається функцією апроксимації.

Характеристика лінійної апроксимації визначається як послідовність $(X',Y', \Delta p)$, де X', Y' – маски входу та маски виходу, а p – імовірність апроксимації.

Лінійні апроксимації довільної функції, як і у випадку з диференціальним аналізом, зручно представляти за допомогою спеціальних таблиць – таблиць лінійних апроксимацій (ЛАТ).

Для побудування ЛАТ ми будемо використовувати методику, яка була розроблена для зменшених моделей, тобто для повномасштабного шифру будемо будувати таблицю при активізації великого шифру зменшеними до 16-ти біт вхідними і вихідними блоками даних як для зменшених до 16 біт версій симетричних шифрів [16], При побудові ЛАТ будуть перевіряться різні 16-бітні значення входів шифру (відкритих текстів), які переходять в різні 16-бітні виходи шифру (шифротекстів). При цьому обчислюються порозрядні суми по модулю 2 добутків 16-ти бітних входів шифру на деякі фіксовані 16-ти бітні числа (вхідні маски) і відповідні порозрядні суми добутків по модулю 2 16-ти бітних виходів шифру на фіксовані 16-ти бітні маски виходу. Вхідні та вихідні маски є індексами входів в комірки таблиці розміру $2^{16} \times 2^{16}$. Сама ж таблиця

отримується шляхом заповненням осередків числами, відповідними кількостям лінійних співвідношень, що виконуються для композицій вхідних бітів, що пройшли маски по стовпцях, і вихідних бітів, що пройшли маски по рядках, при варіації по всій безлічі входів шифру. Залишається зауважити, що сутність кожного лінійного співвідношення полягає в рівності нулю суми по модулю 2 вхідних і вихідних біт, що пройшли обидві маски.

Якщо маску входів позначити через α, а маску виходів через β, то число лінійних переходів α в β для a-го шифру $NS_a(\alpha, \beta)$ (число лінійних співвідношень, що виконуються для маски входів α і маски виходів β) визначається виразом:

$$NS_a(\alpha,\beta)^{def} = \#\{x \mid 0 \le x < A, \oplus \sum_{s=0}^{input-1} x[s]\cdot\alpha[s] = \\ = \oplus \sum_{t=0}^{output-1} cipher(x)[t]\cdot\beta[t]\} - A/2, \tag{4.3}$$

де x – змінна, що описує вхідне значення a-го шифру на поточному кроці, A і $input$ – кількість біт, що складають вхід шифру, $\oplus \sum_{s=0}^{k} x[s]\cdot\alpha[s]$ – операція побітового підсумовування за модулем 2 однойменних компонентів співмножників (скалярний добуток k-бітних векторів).

Алгоритм побудови таблиці наведено нижче:

1) побудова таблиці $2^m \times 2^m$ (m – довжина блоку);
2) заповнення комірок значеннями -2^n;
3) перебір масок входу і виходу a і b;
4) перебір всіх варіантів входу x;
5) обчислення виразу;
6) *HemingWeight*(a & X) + *HemingWeight*(b & *Cipher*[X]);
7) інкрементація комірки [a][b], якщо результат по модулю 2 дорівнює 0;
8) визначення максимального значення таблиці.

Віднімання числа, рівного половині входів, введено для формування величин зміщення щодо середнього значення. В результаті ймовірність p переходу α і β на a-м шифрі визначаться виразом:

$$p = \frac{NS_a(\alpha, \beta)}{A} \tag{4.4}$$

Складність обчислення одного елемента таблиці ЛАТ при застосуванні даного алгоритму становить $O(2^n)$. Враховуючи, що в даній роботі використовується повна версії шифру ШУП, але для аналізу береться 16-бітна частина входу та виходу, загальна складність розрахунку таблиці ЛАТ становить $2^{n+m} \cdot 2^n$ та дорівнює 2^{48}. Дана складність, на сьогоднішній день є досить великою, а з огляду на той факт, що в рамках даної роботи необхідна побудова набору таких таблиць, даний алгоритм не можливо застосовувати на практиці.

Для побудови таблиць лінійних апроксимацій в даній роботі використовується прискорений алгоритм підрахунку ЛАТ.

4.2 Прискорений алгоритм підрахунку ЛАТ

Прискорений алгоритм підрахунку ЛАТ розроблений польським вченим Кшиштофом Шмелем з Познанського Університету Технологій. Він представлений в роботі [17] і описує алгоритм обчислення таблиць апроксимацій *TAf* для випадкової функції *f*, визначеної у вигляді $Y = f(X) : \{0,1\}^n \to \{0,1\}^m$.

Нехай TAf – таблиця апроксимацій функції $Y = f(X) : \{0,1\}^n \to \{0,1\}^m$, а TA$f_0$, TA$f_1$ – таблиці апроксимацій залишкових функцій:

$$f_0 = f(X_{n-1} = 0) = f(0, X_{n-2}, \ldots, X_0) \text{ и } f_1 = (X_{n-1} = 1) = f(1, X_{n-2}, \ldots, X_0). \tag{4.5}$$

Крім того, нехай TAf^{0} та TAf^{1} позначають половини TAf для $X'_{n-1} = 0$ та $X'_{n-1} = 1$ відповідно (рис. 4.1).

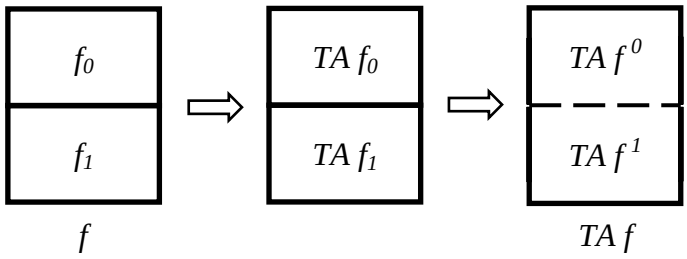

Рисунок 4.1 – Ілюстрація теореми 1

Швидкий метод розрахунку ЛАТ ґрунтується на наступній теоремі.

Теорема 1. для довільної функції f: $\{0,1\}^n \to \{0,1\}^m$, де $n > 1$ та $m \geq 1$ є справедливим вираз

$$\text{TA}f^{0} = \text{TA}f_0 + \text{TA}f_1,\ \text{TA}f^{1} = \text{TA}f_0 - \text{TA}f_1 \qquad (4.6)$$

Швидкий алгоритм обчислення таблиці апроксимацій *TAf* складається з двох кроків. На першому кроці обчислюється початкове значення таблиці *TAf*. Процес отримання початкового значення *TAf* включає використання елементарних апроксимаційних таблиць *TP* всіх залишкових функцій *f*, залежних від однієї зміною X_0. На другому кроці обчислюється кінцеве значення *TAf*, як результат додавання та віднімання цих елементарних таблиць для послідовних значень. Важливою особливістю швидкого алгоритму є те, що обчислення таблиці *TAf* може бути виконано колонка за колонкою, без зберігання всієї таблиці *TAf* в пам'яті.

Процедура обчислення початкового значення колонки таблиці апроксимацій *TAf* для Y' представлена нижче.

INI-TAC(*TAC,Y′, f, n, m, TP*)

1) **for** $X' = 0$ **to** $2^n - 1$ **do**

2) $TAC[X'] = \text{BIT-XOR}(f[X']$ **and** $Y', m)$

3) **for** $X' = 0$ **to** $2^n - 2$ **step** 2 **do**

4) $(TAC[X'], TAC[X'+1]) = TP[TAC[X'], TAC[X'+1]]$

5) **Return**

Ілюстрація даного алгоритму наводиться далі (рис. 4.2).

	$Y=f(X)$
0	0
1	1
2	2
3	3
4	1
5	2
6	3
7	0
8	2
9	3
10	0
11	1
12	3
13	0
14	1
15	2

(a)

$X`$	$Y`$			
	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	1	1
3	0	1	1	0
4	0	1	0	1
5	0	0	1	1
6	0	1	1	0
7	0	0	0	0
8	0	0	1	1
9	0	1	1	0
10	0	0	0	0
11	0	1	0	1
12	0	1	1	0
13	0	0	0	0
14	0	1	0	1
15	0	0	1	1

(b)

$X`$	$Y`$			
	0	1	2	3
0	1	0	1	0
1	0	1	0	1
2	1	0	-1	0
3	0	0	0	-1
4	1	0	0	-1
5	0	-1	1	0
6	1	0	0	1
7	0	-1	-1	0
8	1	0	-1	0
9	0	1	0	-1
10	1	0	1	0
11	0	1	0	1
12	1	0	0	1
13	0	-1	-1	0
14	1	0	0	-1
15	0	-1	1	0

(c)

Рисунок 4.2 – Процедура обчислення початкового значення колонки *TAf* для Y'

В схемі X', Y' – маски входу та виходу функції *f*, *TAC* (column of approximation table) – колонка апроксимаційної таблиці.

Допоміжна функція BIT-XOR(*X*, *n*) обчислює вагу Хеммінга *n*-бітного вектора *X* (кількість одиниць в двійковій послідовності).

BIT-XOR(*X*, *n*)

1) $w = 0$

2) **for** $i = 0$ **to** $n - 1$ **do** $w = w \oplus X_i$

3) **return** *w*

Початкове значення колонки *TAf* для *Y′* обчислюється процедурою INI-TAC(…). На початку (крок 1-2) для кожної маски *X′* обчислюється значення *Y[Y′]* з використанням допоміжної функції BIT-XOR(…). Далі (крок 3-4) кожна пара суміжних значень замінюється парою, що зберігається в таблиці пар *TP*. Сама таблиця *TP* представлена в таблиці 4.1:

Таблиця 4.1 – Таблиця пар *TP*

ν_0	ν_1	$TP[\nu_0, \nu_1]$
0	0	(1, 0)
0	1	(0, 1)
1	0	(0, -1)
1	1	(-1, 0)

Для кожної функції є одна змінна, яка визначається значеннями ν_0 і ν_1 та містить пару значень з правої колонки елементарної таблиці апроксимацій *TP*.

Алгоритм обчислення кінцевого значення колонки *TAf* для *Y′* наведено нижче (рис 4.3).

X`	Y`			
	0	1	2	3
0	2	0	0	0
1	0	2	0	0
2	0	0	2	0
3	0	0	0	2
4	2	0	0	0
5	0	-2	0	0
6	0	0	0	-2
7	0	0	2	0
8	2	0	0	0
9	0	2	0	0
10	0	0	-2	0
11	0	0	0	-2
12	2	0	0	0
13	0	-2	0	0
14	0	0	0	2
15	0	0	-2	0

(a)

X`	Y`			
	0	1	2	3
0	4	0	0	0
1	0	0	0	0
2	0	0	2	-2
3	0	0	2	2
4	0	0	0	0
5	0	4	0	0
6	0	0	2	2
7	0	0	-2	2
8	4	0	0	0
9	0	0	0	0
10	0	0	-2	2
11	0	0	-2	-2
12	0	0	0	0
13	0	4	0	0
14	0	0	-2	-2
15	0	0	2	-2

(b)

X`	Y`			
	0	1	2	3
0	8	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	8	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	4	-4
11	0	0	4	4
12	0	0	0	0
13	0	0	0	0
14	0	0	4	4
15	0	0	-4	4

(c)

Рисунок 4.3 – Процедура обчислення кінцевого значення колонки *TAf*

CALC-TAC(*TAC, i, j*)

1) **if** $j - i > 2$ **then**
2) $k = (i + j)$ **div** 2
3) CALC-TAC(*TAC, i, k*)
4) CALC-TAC(*TAC, k*+1, *j*)
5) SUMSUB-TAC(*TAC, i, k, k*+1, *j*)

Кінцеве значення колонки *TAf* для *Y´* обчислюється рекурсивною процедурою CALC-TAC(…).У першому виклику процедури повинно бути використано початкове значення колонки *TAf* і діапазон рядків: $i = 0$, $j = 2^n - 1$. Для більшого діапазону друге завдання розбивається на дві підзадачі (крок 3-4). Колонка таблиці апроксимацій обчислюється допоміжною процедурою SUMSUB-TAC(…).

SUMSUB-TAC(*TAC*, i_1, j_1, i_2, j_2)

1) **for** $i = 0$ **to** $j_1 - i_1$ **do**
2) $(TAC[i_1+i], TAC[i_2+i]) = (TAC[i_1+i] + TAC[i_2+i], TAC[i_1+i] - TAC[i_2+i])$
3) **return**

Процедура SUMSUB-TAC(…) для двох частин колонки *TAf* (отриманих в результаті рішення двох підзадач) замінює першу з них їх сумою, а другу – їх різницею.

Далі представлений швидкий алгоритм обчислення значення *maxTA* для функції *f*:

maxTA(*f, n, m*)

1) max = 0
2) **for** $Y' = 0$ **to** $2^m - 1$ **do**

3) INI-TAC(*TAC, Y′, f, n, m, TP*)
4) CALC-TAC(*TAC*, 0, $2^n - 1$)
5) **for** $X' = 0$ **to** $2^n - 1$ **do**
6) **if** $X' \neq 0$ **or** $Y' \neq 0$ **then**
7) **if** max < |*TAC*[*X′*]| **then** max = |*TAC*[*X′*]|
8) **return** max

Процедури INI-TAC(…) и CALC-TAC(…) обчислюють колонку *TAf* для *Y′* за лінійний час *O*(*n*+*m*) для одного елемента. Швидкий алгоритм *maxTA*(…) обчислює значення *maxTA*, яке відповідає кращій ненульовій лінійній апроксимації функції *f*. Обчислення проводиться колонка за колонкою за час *O*(*n*+*m*) для одного елемента. Загальна складність побудови ЛАТ становить 2^{32} і дозволяє застосовувати даний алгоритм на практиці (час побудови таблиці 2-30 хвилин, в залежності від потужності системи).

4.3 Таблиці різниць диференціалів

Як вже було сказано раніше, в роботі використовується повна версія ітераційного симетричного шифру ШУП, а для процедури побудування таблиці розподілу диференціалів використовується лише 16 біт вхідної та вихідної послідовності. Це означає, що таблиця диференціалів має розмір $2^{16} \times 2^{16}$. Як і у випадку S-блоків в атаці Ф. Шаміра і Е. Бихама, при побудові таблиці проглядаються всі можливі різниці (суми по модулю 2) різних пар відкритих текстів (входів в таблиці по рядках) і відповідні різниці пар зашифрованих текстів (входів в таблицю по стовпцях). Отримані числа є індексами входів (по рядках і по стовпцях) в комірках таблиць розподілу різниць (диференціалів) шифру [18]. Сама таблиця різниць виходить заповненням комірок числами, які відповідають кількості пар вхідних різниць (входів в таблиці по рядках), що утворюють задані значення вихідних різниць (входів в таблицю по стовпцях) при варіації по всьому безлічі пар входів шифру, що формують задану вхідну

різницю, тобто кожен рядок таблиці відповідає конкретному значенню вхідної різниці, а кожен стовпець відповідає конкретному значенню вихідної різниці, а значення відповідних комірок таблиці відповідають кількості можливих пар з цими значеннями вхідних різниць, що утворюють задані значення вихідних різниць.

Зрозуміло, що значення в комірках таблиці не перевищують числа 2^{16}. Суми всіх значень для кожного рядка також дорівнюють 2^{16}. Крім того, значення 2^{16} зустрічається в таблиці всього один раз для значень вхідних та вихідних різниць [19], рівних 0, тобто при проходженні через шифр пар однакових відкритих текстів, які перетворюються в пари однакових шифротекстів. Цей випадок з розгляду виключається як неінформативний.

Слід зауважити, що в роботі не планується побудувати атаку на шифр, тому корисність цього підходу бачиться в отриманні середніх показників стійкості шифру.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТА ПОСТАНОВА ЕКСПЕРИМЕНТІВ

5.1 Побудова програмної реалізації БСШ ШУП

Для досягнення поставлених задач, необхідно програмно реалізувати блоковий симетричний шифр ШУП-256 (або принаймні виконання перших чотирьох циклів). Програма повинна виконувати шифрування заданого масиву текстів (65536 текстів), які змінюються між собою на один біт (зміна 4-го байту або зміна 32-го байту), а також зберігати проміжні результати шифрування на кожному раунді. Програма виконує побудування таблиць різниць диференціалів (DDT) і таблиць лінійної апроксимації (LAT) на основі шифрованих текстів для кожного раунду та здійснює пошук їх максимумів. При реалізації програми можна обмежитися виконанням лише перших чотирьох раундів, тому що максимум на останніх циклах будуть майже однаковими.

5.1.1 Опис програми

В рамках даної дипломної роботи була створена програмна реалізація БСШ ШУП-256. Алгоритм був скорочений до чотирьох циклів, а також проводиться без операції розгортання ключа (з одним підключем на всіх циклах). В рамках експерименту нас цікавлять лише данні, які були отримані після перших чотирьох раундів. Використання повного алгоритму приводить до збільшення часу обробки даних, тому є нераціональним.

Програма має назву «SHUP.exe» та реалізована у вигляді консольного додатка. Код програми знаходиться в файлі «SHUP.cpp».

Програма реалізована на мові C++ і може бути запущена з використанням практично будь-якої операційної системи, наприклад, операційних системах сімейства Windows не нижче Windows 98. Для компіляції коду потрібне

програмне середовище, яке підтримує мову C++, наприклад, Microsoft Visual Studio 2010.

5.1.2 Функціональне призначення

Програма призначена для отримання експериментальних даних та не може використовуватися на практиці, бо використовує скорочений алгоритм, який не сертифікований в Україні.

5.1.3 Опис логічної структури

Програма реалізована відповідно до наступного алгоритму:

1) Вибір допоміжних функцій (генерація ключа та текстів);
2) вибір режиму шифрування (з надбудовою або без надбудови);
3) виконання чотирьох раундів алгоритму шифрування;
4) Побудування таблиць DDT та LAT для кожного раунду, пошук їх максимумів;
5) вивід результатів до файлу.

Застосовуються користувальницькі функції:

1) Multiply(u8 a, u8 b), де a та b – числа, що перемножуються. Функція виконує МДР-перемноження та повертає число типу unsigned __int8.

2) gen_key_256(). Функція генерує ключ розміром 256 біт та записує його до файлу «key.txt» в шістнадцятковому вигляді.

3) char_to_u32(char* key_char, u32* key_u32), де key_char – зчитаний з файлу масив даних; key_u32 – 32-бітове слово. Функція необхідна для перетворення зчитаного ключа/тексту в 32-бітове слово. Без використання цієї функції текст буде зчитуватися в десятковому вигляді, а не в шістнадцятковому.

5) u8_to_u32(const u8* state_u8, u32& state_u32), де state_u8 – 8-бітове слово; state_u32 – 32-бітове слово. Функція перетворює чотири окремих байти в одне 32-бітове слово.

6) u32_to_u8(const u32 state_u32, u8* state_u8), де state_u32 – 32-бітове слово; state_u8 – 8-бітове слово. Функція ділить одне 32-бітове слово на чотири окремих байта.

7) rebuild(u32* text), text – 256-бітовий масив вхідних даних. Функція виконує надбудову. Ознайомитися з алгоритмом надбудови можна в розділі 3.2.

8) SL(u32& state), де state – 32-бітовий стан шифрування. Функція виконує SL-перетворення (SubBytes та МДР-перемноження). Ознайомитися з алгоритмом функції можна в розділі 3.4.

9) round(u32* state), де state – 32-бітовий стан шифрування. Функція виконує основний етап програми – раунди шифрування. Ознайомитися з алгоритмом функції можна в розділі 3.3.

10) encryption(bool trig), де trig – механізм активування надбудови (0 – шифрування без надбудови, 1 – шифрування з надбудовою). Функція зчитує ключ з файлу «key.txt» та генерує тексти, проводить шифрування кожного з 65536 текстів.

11) dif(). Функція будує таблицю різниць диференціалів для кожного з чотирьох раундів, а також знаходить максимум для всіх циклів шифрування.

12) heming(u32 value), де value – число, що перевіряється. Функція рахує вагу Хемінга та повертає 1 якщо вона більше нуля. В противному випадку повертає 0.

13) lat(). Функція будує таблицю лінійної апроксимації для кожного з чотирьох раундів, використовуючи швидкий алгоритм підрахунку ЛАТ, та знаходить максимум для всіх циклів шифрування.

З повним кодом програми можна ознайомитися в «Додаток Б».

5.1.4 Використовувані технічні засоби

Програма настільки невимоглива до ресурсів, що може бути запущена на будь-якому сучасному комп'ютері.

5.1.5 Вхідні та вихідні дані

Для запуску програми необхідно відкрити файл «SHUP.exe». Перед виконанням експерименту згенерувати ключ. Далі необхідно вибрати режим шифрування (без надбудови або з надбудовою). Програма перейде до генерації текстів, їх шифрування та почне побудування таблиць (DDT та LAT). Цей процес потребує багато часу (приблизно годину на одну таблицю, тобто 8 годин на одне виконання програми), тому в консоль відображується який з раундів обробляється та прогрес виконання в процентах (рис. 5.1). Результати максимумів таблиць різниць диференціалів та лінійних апроксимацій після виконання виводяться до файлів «DDT.txt» та «LAT.txt» відповідно.

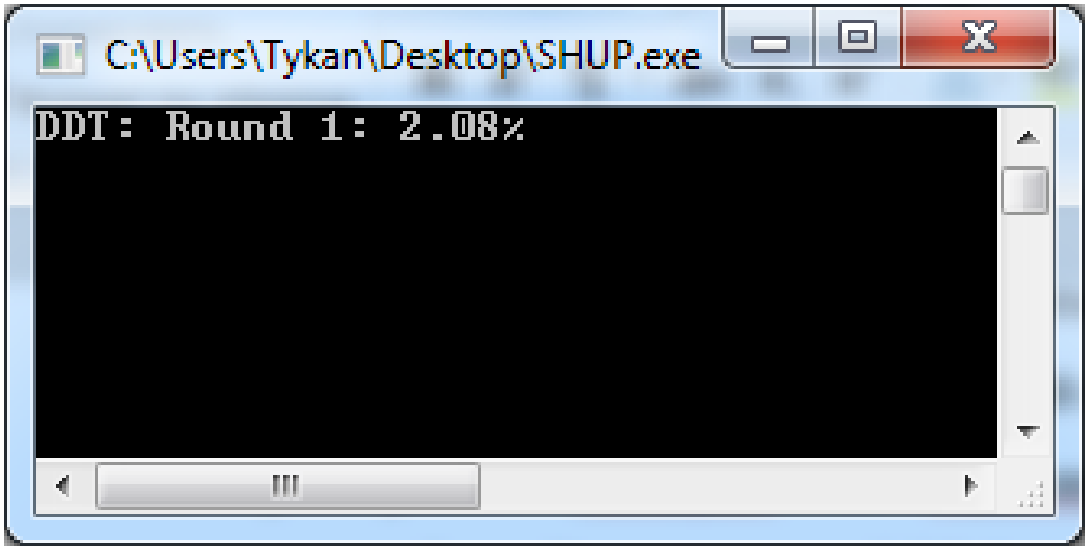

Рисунок 5.1 – Приклад роботи програми «SHUP.exe» для першого експерименту без надбудови

5.2 Постанова експериментів

В ході виконання дипломною роботи ставиться задача виконати шістнадцять тестів, в кожному з котрих потрібно побудувати 8 таблиць (4 таблиці DDT та 4 таблиці LAT) та знайти їх максимуми. Для цього генеруються 65535 пар текстів, що відрізняються один від одного бітами різниць. Різниця

подається на 32 або на 4 вхідний байт. Від шифрованого тексту на кожному раунді береться перші або останні 16 біт, які записуються та використовуються для побудування таблиць. Після чого послідовно запускаються алгоритми побудування таблиць DDT та LAT та алгоритм пошуку максимумів.

Нижче наведено короткий опис тестів:

1) різниця подається на 31-32-ий байт, на виході береться перші 16 біт, тест виконується без надбудови;
2) різниця подається на 31-32-ий байт, на виході береться перші 16 біт, тест виконується з надбудовою;
3) різниця подається на 31-32-ий байт, на виході береться останні 16 біт, тест виконується без надбудови;
4) різниця подається на 31-32-ий байт, на виході береться останні 16 біт, тест виконується з надбудовою;
5) різниця подається на 3-4-ий байт, на виході береться перші 16 біт, тест виконується без надбудови;
6) різниця подається на 3-4-ий байт, на виході береться перші 16 біт, тест виконується з надбудовою;
7) різниця подається на 3-4-ий байт, на виході береться останні 16 біт, тест виконується без надбудови;
8) різниця подається на 3-4-ий байт, на виході береться останні 16 біт, тест виконується з надбудовою.

Той самий же набір тестів виконується для випадкової та для стандартної таблиці підстановок.

6. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

Згідно з викладеною вище методикою, а також за відповідними вхідними даними була створена програмна реалізація на мові програмування C++ (Додаток Б), за допомогою якої був виконаний аналіз. Слід зазначити, що дослідження проводилися з використанням одного й того ж ключа на всіх етапах.

В ході виконання практичних досліджень було виконано шістнадцять тестів. В кожному з них було побудовано 8 таблиць: 4 таблиці різниць диференціалів (DDT) та 4 таблиці лінійної апроксимації (LAT) (одна таблиця на один раунд шифрування). В результаті були отримані 128 значень максимумів DDT та LAT.

Основні умови проведення кожного тесту описані нижче:

1) різниця подається на 31-32-ий байт, для побудування таблиць береться перші 16 біт виходу;
2) різниця подається на 31-32-ий байт, для побудування таблиць береться останні 16 біт виходу;
3) різниця подається на 3-4-ий байт, для побудування таблиць береться перші 16 біт виходу;
4) різниця подається на 3-4-ий байт, для побудування таблиць береться останні 16 біт виходу.

До таблиці 6.1 занесені результати максимумів таблиць різниць диференціалів для шифру без надбудови. Такий варіант є найгіршим для шифру, бо функція надбудови значно покращує властивості шифру та дозволяє активувати «лавинний ефект» на першому циклі шифрування [16].

Таблиця 6.1 – Максимуми DDT для шифру без надбудови

№ тесту	Раунд	Стандартна таблиця підстановки	Випадкова таблиця підстановки
1	1	1024	3072
	2	18	18
	3	18	20
	4	20	22
2	1	1024	3072
	2	20	20
	3	20	18
	4	18	20
3	1	20	20
	2	20	20
	3	18	18
	4	20	20
4	1	18	18
	2	18	20
	3	18	18
	4	20	18

Ситуація, коли різниця подається на 31-32-ий байт і шифр не використовує функцію надбудови, приводить то того, що шифр приходить до стану випадкової підстановки на один цикл пізніше. Згідно з таблицею вище (1-2 тест), це також впливає на диференційні показники шифру. В цьому випадку добре помітно збільшення максимумів на першому циклі при використанні випадкових таблиць підстановки. Але вже на другому циклі показники приходить до нормальних значень [20].

При змінні 3-4-го байту (3-4 тест) алгоритмічні особливості дозволяють активувати всі подальші байти шифру, що забезпечує ті ж самі показники, що й з надбудовою (табл. 6.2).

Таблиця 6.2 – Максимуми DDT для шифру з надбудовою

№ тесту	Раунд	Стандартна таблиця підстановки	Випадкова таблиця підстановки
1	1	20	18
	2	20	20
	3	18	20
	4	18	18
2	1	20	20
	2	20	18
	3	18	18
	4	20	20
3	1	20	20
	2	20	20
	3	18	18
	4	20	20
4	1	18	18
	2	18	20
	3	18	18
	4	20	18

Результати максимумів лінійної апроксимації (табл. 6.3) дають приблизно ту ж саму картину, що і максимуми різниць диференціалів.

Таблиця 6.3 – Максимуми LAT для шифру без надбудови

№ тесту	Раунд	Стандартна таблиця підстановки	Випадкова таблиця підстановки
1	1	4096	9728
	2	859	812
	3	815	815
	4	831	820
2	1	4096	9728
	2	810	815
	3	809	828
	4	802	785

Продовження табл. 6.3

№ тесту	Раунд	Стандартна таблиця підстановки	Випадкова таблиця підстановки
3	1	843	804
	2	868	825
	3	816	881
	4	806	833
4	1	824	814
	2	796	864
	3	835	781
	4	844	793

Зміна 31-32-го байту приводить до погіршення показників в перших двох тестах на перших раундах, але вони приходять до норми вже на другому раунді. Якщо різниця подається на 3-4-ий байт, це приводить до тих же результатів, що і в шифрі з надбудовою (табл. 6.4).

Таблиця 6.4 – Максимуми LAT для шифру з надбудовою

№ тесту	Раунд	Стандартна таблиця підстановки	Випадкова таблиця підстановки
1	1	879	795
	2	820	812
	3	792	805
	4	820	782
2	1	848	847
	2	820	802
	3	809	819
	4	874	830
3	1	843	804
	2	868	825
	3	816	881
	4	806	833
4	1	824	814
	2	796	864
	3	835	781
	4	844	793

При використанні випадкових таблиць підстановки диференційні та лінійні властивості не відрізняються від результатів максимумів при використанні стандартної таблиці підстановки та відповідають нормальним значенням [20]. Збільшення максимумів спостерігається лише на перших циклах шифрування без надбудови, але враховуючи, що функція надбудови є однією з основних нововведень шифру ШУП, то застосування шифру без неї є недоцільним. З цього можна зробити висновок, що погіршення лінійних та диференційних показників при використанні випадкових таблиць підстановки (у випадку ШУП) є неістотними і цим можна знехтувати [21].

Використання випадкових таблиць підстановки дозволить ввести додатковий секретний параметр шифру, що підвищить складність криптоаналізу. Якщо ж шифр буде оперувати чотирма випадковими таблицями, це може суттєво ускладнити роботу зловмисника.

Цей метод також має і свої недоліки. Використання випадкових таблиць підстановки при шифруванні буде потребувати їх і при розшифруванні інформації. Тобто окрім проблеми розповсюдження ключа доведеться вирішувати також проблему розповсюдження самих таблиць підстановки. В деяких випадках такий підхід може бути доцільним.

ВИСНОВКИ

Дослідження криптографічних властивостей блокових симетричних шифрів проводяться вже десятки років, однак це питання залишається актуальним й в наші дні. На сьогодні основні зусилля науковців спрямовані на розробку методик аналізу стійкості блокових симетричних шифрів. Технології БСШ розвиваються дуже стрімко. Ще недавно алгоритм шифрування DES з довжиною ключа 56 біт й блоками шифрування в 64 біта з високою швидкістю, вважався найбільшим досягненням в області криптографії. На сьогоднішній день вже створені алгоритми, які можуть шифруваті блоки по 1024 біт. Оцінити практичним шляхом стійкість таких алгоритмів до криптографічних атак неможливо, бо обчислювальна здатність комп'ютерних засобів ще не дозволяє обробляти такий обсяг даних. Таким чином з'явилася гостра необхідність створення методик для теоретичної оцінки стійкості шифрів до атак криптоаналізу, зокрема оцінки стійкості до атак лінійного та диференційного криптоаналізу – найбільш ефективних атак на БСШ [19].

Дана робота була спрямована на вивчення блокових симетричних шифрів та їх диференційних та лінійних властивостей. На даний момент часу, блокові симетричні шифри є найбільш застосовуваним засобом криптографічного захисту інформації, бо дозволяють швидко та надійно шифрувати та розшифровувати данні, при цьому використовуючи ключ меншим розміром, ніж асиметричні засоби шифрування. БСШ може застосовуватися як самостійний інструмент, так й в інших алгоритмах захисту даних. Наприклад, у складі MAC-кодів або в гібридних системах шифрування. Такі системи дають можливість вирішити питання розповсюдження ключів за допомогою асиметричного шифрування та нівелювати недоліки блочних симетричних шифрів.

В ході виконання роботи досліджено блоковий симетричний шифр ШУП («шифр с управляемыми подстановками») та створена його програмна реалізація, що дозволяє експериментальним шляхом вивчати диференційні та лінійні властивості цього алгоритму.

Шифр ШУП – перспективний БСШ, що дозволяє шифрувати блоки тексту розміром 256 або 512 бітів з використанням аналогічних 256-бітних та 512-бітнних секретних ключів відповідно. Застосування функції надбудови в алгоритмі ШУП дозволяє досягти «лавинного ефекту» (тобто зміни більшості біт вихідної послідовності при зміні хоча б одного біта у вхідному тексті або ключі) вже на першому циклі [16]. Це в свою чергу приводить шифр до випадкової підстановки на цикл шифрування раніше, ніж більшість сучасних БСШ. Для багатьох сучасних шифрів цей показник досягається за три та більше раундів [12].

В роботі була розглянута можливість використання випадкових таблиць підстановки шляхом оцінки диференційних та лінійних максимумів, а також з'ясування впливу на ефективність перетворення функції надбудови. В результаті були отримані наступні данні. При використанні випадкових таблиць відбувається збільшення максимумів на першому циклі шифрування без функції надбудови, але на другому циклі ці показник приходять до нормальних значень. Враховуючи, що функція надбудови є основоположною в шифрі ШУП, цим збільшенням можна знехтувати. При використанні шифру з функцією надбудови, а також з використанням випадкових таблиць підстановки, диференційні та лінійні максимуми не відрізняються від показників при використанні стандартної таблиці підстановки вже з першого циклу, тобто функція надбудови не покращує показників випадковості шифру.

Таким чином розглянутий підхід дозволяє покращити показники випадковості шифру ШУП в порівнянні з шифром Rijndael. (для приходу шифру Rijndael до стану випадкової підстановки потрібно мінімум 3-4-ри цикли), причому шифр ШУП дозволяє без зниження стійкості використовувати в шифрі випадкові S-блоки.

Наостанок хотілося б відзначити, що задача оцінки надійності криптографічних засобів забезпечення захисту інформації буде актуальна у всі часи. Як точно підмітив знаменитий американський письменник Едгар Аллан По: «Можна з точністю стверджувати, що людський розум не може породити такий шифр, який би людський розум не зумів розгадати».

ПЕРЕЛІК ПОСИЛАНЬ

1. Oliynykov R.V. A new encryption standard of Ukraine: The Kalyna block cipher. / R.V. Oliynykov, I.D. Gorbenko, O.V. Kazymyrov et. al. // Cryptology ePrint Archive. http://eprint.iacr.org/2015/650.
2. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: Триумф, 2002. – 610 с.
3. Баричев С. Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов // – М.: Диалог-МИФИ, 2011. – 176 с.
4. Gustavus J. Simmons. Advances in Cryptology: Proceedings of CRYPTO 84. Berlin: Springer. – 1985. – pp. 411–431.
5. Винокуров А.Ю. Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86. – Рукопись. – 1997. – с. 25.
6. Шеннон К. Работы по теории информации и кибернетике. – М., ИЛ, 1963. – с. 333-369.
7. Hong S. Provable Security against Differential and Linear cryptanalysis for SPN Structure. B / S. Hong, S. Lee, J. Lim // FSE 2000, LNCS 1978, pp. 273-283, 2001.
8. Лисицкая И.В. Подход к криптоанализу современных шифров. / В.И. Долгов, И.В. Лисицкая, Р.В. Олейников // Материалы второй международной конференции ″Современные информационные системы. Проблемы и тенденции развития″, Харьков-Туапсе, Украина. – 2007. – с. 435-436.
9. Thomas H. Cormen. Introduction to Algorithms. – MIT Press, 2001. – p. 1292.

10. Кузнецов А.А. Линейные свойства блочных симметричных шифров, представленных на украинский конкурс / А.А Кузнецов, И. В. Лисицкая, С.А. Исаев // Прикладная радиоэлектроника. – 2011. – с. 135-140.
11. Nyberg K. Provable security against differential cryptanalysis. / K. Nyberg, L. Knudsen // Journal of Cryptology, vol.8, no.1, 1995.
12. Долгов В.И. Новая концепция проектирования блочных симметричных шифров / В.И. Долгов, И.В. Лисицкая, К.Е. Лисицкий // 0485-8972. – Радиотехника. – Всеукр. межвед. научн.-техн. сб. – 2016. – Вип.186. – с. 132-152.
13. Лисицький К.Є. Удосконалена конструкція початкового перетворення для SPN шифрів / К.Є.Лисицький // XIX Міжнародна науково-практична конференція «Безпека інформації в інформаційно-телекомунікаційних системах». – с. 133-153.
14. Баричев С. Г. Стандарт AES. Алгоритм Rijdael / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов // Основы современной криптографии. – 3-е изд. – М.: Диалог-МИФИ, 2011. – с. 30–35.
15. Matsui, M. Linear Cryptanalysis Method for DES Cipher. / M. Matsui. // Advances in Cryptology Eurocrypt'93, 1993.
16. Лисицька І.В. Дослідження показників випадковості шифру ШУП / І.В. Лисицька, В.О. Васильєв // Прикладна радіоелектроніка. – 2019 (в друці).
17. Chmiel, K. Fast computation of approximation tables. / K. Chmiel // Information Processing and Security Systems 2005, Part II, 125-134.
18. Долгов В.И. Исследование дифференциальных свойств мини-шифров BABY-ADE и BABY-AES. / В.И. Долгов, А.А. Кузнецов, Р.В. Сергиенко, О.И. Олешко // Прикладная радиоэлектроника. – 2009. – Т.8. – №3. – с. 252-257.

19. Олейников Р.В. Исследование дифференциальных свойств подстановок / Р.В. Олейников, И.В. Лисицкая, А.В. Широков, К.Е. Лисицкий // Компьютерные науки и технологии: сб. научн. тр. первой межд. НТК – Ч. I. – Б., 2009. – с. 59-63.
20. Сорока Л.С. Исследование дифференциальных свойств блочно-симметричных шифров. / Л.С. Сорока, А.А. Кузнецов, И.В. Московченко, С.А. Исаев // Системи обробки інформації. – 2010. – №6. – с. 286-294.
21. Лисицкая И.В. О минимальном числе S-блоков, активизируемых на первых циклах SPN шифров / И.В. Лисицкая, В.А. Васильев // Прикладная радиоэлектроника. – 2019 (в печати).