НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК 003.26

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Оцінювання стійкості біт-орієнтованих блокових шифрів до атак усічених диференціалів спеціального виду»**

Виконав:
студент IV курсу, групи ФІ-04
Медведцький Костянтин Анатолійович _____

Керівник:
доцент кафедри ММЗІ, к.т.н.
Яковлєв Сергій Володимирович _____

Науковий консультант:
асистент кафедри ММЗІ, аспірант
Якимчук Олексій Петрович _____

Рецензент:
старший викладач кафедри ММАД, д.ф.
Яйлимова Ганна Олексіївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Медведцький Костянтин Анатолійович

1. Тема роботи: «*Оцінювання стійкості біт-орієнтованих блокових шифрів до атак усічених диференціалів спеціального виду*», науковий керівник дисертації: доцент кафедри ММЗІ, к.т.н. Яковлєв Сергій Володимирович,

затверджені наказом по університету №__ від «__» _____ 2024 р.

2. Термін подання студентом роботи: «__» _____ 2024 р.

3. Об'єкт дослідження: інформаційні процеси в системах захисту інформації.

4. Предмет дослідження: моделі та методи диференціального криптоаналізу блокових шифрів.

5. Перелік завдань:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) перевірити можливість застосування усічених диференціалів на основі масок спеціального вигляду для побудови багатораундових диференціалів та диференціальних характеристик;

3) продемонструвати застосовність усічених диференціалів на основі масок спеціального вигляду на біт-орієнтованому блоковому шифрі;

4) розробити правила застосування операції додавання за модулем два до масок, що розглядаються;

5) продемонструвати застосовність усічених диференціалів на основі масок спеціального вигляду на блоковому шифрі, що використовує операцію додавання за модулем два у своїй структурі.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:Презентація доповіді

7. Орієнтовний перелік публікацій:

1) XXII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», м. Київ, 13 — 17 травня 2024 року, доповідь «Аналіз підходів до пошуку високоймовірнісних усічених диференціалів спеціального вигляду» [9];

2) Всеукраїнська науково-практична конференція «Theoretical and applied cybersecurity», м. Київ, 30 травня 2024 року, доповідь «Пошук усічених диференціальних характеристик спеціального виду для шифру LBlock» (збірник матеріалів в процесі публікування).

8. Дата видачі завдання: 10 вересня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2023 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2023 р.	Виконано
3	Підготовка першого розділу роботи	Жовтень-грудень 2023 р.	Виконано
4	Виконання завдань дослідження	Січень-квітень 2024 р.	Виконано
5	Підготовка інших розділів роботи	Квітень-травень 2024 р.	Виконано
6	Підготовка публікацій по темі роботи на конференції	Квітень-травень 2024 р.	Виконано
7	Оформлення дипломної роботи	Травень 2024 р.	Виконано

Студент _____ Костянтин МЕДВЕДЦЬКИЙ

Керівник _____ Сергій ЯКОВЛЄВ

РЕФЕРАТ

Кваліфікаційна робота містить: 51 стор., 3 рисунки, 13 таблиць, 10 джерел.

Метою роботи є уточнення методів диференціального криптоаналізу біт-орієнтованих симетричних шифрів. Об'єктом дослідження є інформаційні процеси в системах захисту інформації. Предметом дослідження є моделі та методи диференціального криптоаналізу блокових шифрів.

У ході проведення дослідження було запропоновано покращені диференціальні характеристики для шифрів $PRESENT$ та $GIFT-64$ з використанням усічених диференціалів спеціального виду. Було розроблено правила побітового додавання за модулем два для масок, що використовують усічені диференціали спеціального виду. Також було побудовано диференціальну характеристику для шифру $LBlock$, що дозволяє передбачити зміну 2 біт після 10 раундів шифрування.

БЛОКОВИЙ ШИФР, ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ, ДИФЕРЕНЦІАЛЬНА ХАРАКТЕРИСТИКА, УСІЧЕНІ ДИФЕРЕНЦІАЛИ

ABSTRACT

Qualification work contains: 51 pages, 3 figures, 13 tables, 10 sources.

The purpose of the work is to clarify the methods of differential cryptanalysis of bit-oriented symmetric ciphers. The object of research is information processes in information security systems. The subject of research is models and methods of differential cryptanalysis of block ciphers.

In the course of the study, improved differential characteristics for the $PRESENT$ and $GIFT-64$ ciphers were proposed using truncated differentials of a special kind. The rules of bitwise addition modulo two for masks using truncated differentials of a special kind were developed. Also, a differential characteristic for the *LBlock* cipher was constructed, which allows us to predict the change of 2 bits after 10 rounds of encryption.

BLOCK CIPHER, DIFFERENTIAL CRYPTANALYSIS, DIFFERENTIAL CHARACTERISTIC, TRUNCATED DIFFERENTIALS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\oplus$ — операція побітового додавання за модулем два

V_n — множина двійкових векторів довжини n

V_n^* — множина векторів довжини n з елементами з множини $\{0, 1, *\}$

$\big[P\big]$ — дужки Айверсона: $\big[P\big] = 1$, якщо P — істинне, в інакшому випадку $\big[P\big] = 0$

$\Delta(a)$ — множина усіх можливих різниць для маски a

$TDP^f(\alpha, \beta)$ — ймовірність переходу маски α в маску β при шифруючому перетворенні f

$TD^f(\alpha, \beta)$ — множина двійкових векторів, що з вхідної різниці α дають вихідну різницю β

$EDP(\alpha \xrightarrow{f} \beta_1 \xrightarrow{f} \ldots \xrightarrow{f} \beta_r)$ — ймовірність r-раундової диференціальної характеристики

ВСТУП

Актуальність дослідження. Хоча диференціальний криптоаналіз успішно використовували до деяких симетричних шифрів, але він був не ефективний у застосуванні до всіх шифрів. Тому у 1995-му році Ларсом Кнудсеном [4] було запропоновано дещо інший підхід - використання усічених диференціалів, що виявився більш ефективним за звичайний диференціальний криптоаналіз.

У роботі О. П. Якимчука було запропоновано формалізовану теорію, яка описує підхід до диференціального криптоаналізу на основі усічених диференціалів та дозволяє проводити оцінку стійкості шифрів то такого виду криптоаналізу.

Метою дослідження є уточнення методів диференціального криптоаналізу біт-орієнтованих симетричних шифрів. Для досягнення мети необхідно виконати такі **завдання**:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) перевірити можливість застосування усічених диференціалів на основі масок спеціального вигляду для побудови багатораундових диференціалів та диференціальних характеристик;

3) продемонструвати застосовність усічених диференціалів на основі масок спеціального вигляду на біт-орієнтованому блоковому шифрі;

4) розробити правила застосування операції додавання за модулем два до масок, що розглядаються;

5) продемонструвати застосовність усічених диференціалів на основі масок спеціального вигляду на блоковому шифрі, що використовує операцію додавання за модулем два у своїй структурі.

Об'єктом дослідження є інформаційні процеси в системах захисту інформації.

Предметом дослідження є моделі та методи диференціального криптоаналізу блокових шифрів.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи теорії ймовірності, комбінаторного аналізу, лінійної алгебри.

Наукова новизна: у роботі було виявлено неможливість побудови багатораундових усічених диференціалів спеціального виду та розроблено правила застосування операції побітового додавання за модулем два до масок, що розглядаються.

Практичне значення результатів полягає у можливості застосування методу пошуку диференціальних характеристик з використанням усічених диференціалів до блокових шифрів, зокрема до таких, що використовують операцію побітового додавання за модулем два у своїй структурі.

Апробація результатів та публікації. Результати даної роботи були частково представлені на таких конференціях:

1) XXII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», м. Київ, 13 — 17 травня 2024 року, доповідь «Аналіз підходів до пошуку високоймовірнісних усічених диференціалів спеціального вигляду» [9];

2) Всеукраїнська науково-практична конференція «Theoretical and applied cybersecurity», м. Київ, 30 травня 2024 року, доповідь «Пошук усічених диференціальних характеристик спеціального виду для шифру LBlock» (збірник матеріалів в процесі публікування).

1 ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ НА ОСНОВІ УСІЧЕНИХ ДИФЕРЕНЦІАЛІВ

В даному розділі буде розглянуто теоретичні відомості про диференціальний криптоаналіз, різні підходи до визначення усіченого диференціалу та атаки з їх використанням. Також наведено специфікації до шифрів $PRESENT$, $GIFT$-64 та $LBlock$, що будуть використовуватись в другому та третьому розділах.

1.1 Основні поняття диференціального криптоаналізу

Нехай $V_n = \{0, 1, *\}$ - простір бітових векторів довжини n, і кожна з операцій $\circ, \bullet$ визначає структуру абелевої групи з нейтральним елементом на V_n.

Для скорочення позначення n-символьного вектора з великою кількістю однакових символів будемо використовувати нотацію $b_1^t b_2^{n-t}$ — що означає, що вектор має вигляд: t символів b_1 спочатку і $n-t$ символів b_2 потім.

Нехай M - множина відкритих текстів, K - множина ключів, C - множина шифротекстів.

Означення 1.1. *Шифруюче перетворення* f — це відображення вигляду $f : M \times K \to C$, що для будь-якого $k \in K$ відображення f_k є бієктивним.

Означення 1.2. *Ітеративний* n*-раундовий шифр* E — це відображення виду $E : V_q \times K^n \to V_q$, що є композицією r шифруючих перетворень:

$$E(x) = f_{k_n}^n(f_{k_{n-1}}^{n-1}(\dots(f_{k_1}^1(x))\dots)),$$

де $f_{k_i}^i$ — шифруюче перетворення i-того раунду шифрування. Також

вважається, що раундові ключі $k = (k_1, k_2, \ldots, k_n)$ - випадкові, незалежні та рівномірно розподілені на всьому просторі ключів.

Атаки, що базуються на диференціальному криптоаналізі, відносять до класу атак на останній раунд. Ціллю таких атак є відновлення раундового ключа останнього раунду шифрування.

Введемо такі позначення:

1) $\circ, \bullet$ — операції в просторі V_n;

2) f_r — шифруюче перетворення останнього раунду шифрування;

3) $F_{1,r-1}$ — композиція шифруючих перетворень з першого по $r-1$ раунд;

4) E — r-раундовий шифр та $E(x) = f_r(F_{1,r-1}(x))$;

5) $\alpha, \beta \in V_n$ — деякі двійкові вектори, або ж різниці;

6) $x, x' \in V_n$ — відкриті тексти та $x' = x \circ \alpha$;

7) $y, y', c, c' \in V_n$ — шифротексти, де $c = E(x)$, $c' = E(x')$.

Нехай для вказаного α з ймовірністю $p > 2^{-n}$ виконується $y' = y \bullet \beta$. Тоді криптоаналітик може побудувати атаку наступним чином наступним чином:

1) Криптоаналітик накопичує певну кількість пар відкритих текстів (x, x') та відповідних їм шифротекстів (c, c').

2) Для кожного кандидата в ключі k криптоаналітик розшифровує пари шифротекстів (c, c') та отримує (y, y').

3) Далі криптоаналітик перевіряє чи $y' = y \bullet \beta$, та знаходить кількість таких пар, що цьому задовольняють, позначимо їхню кількість як N. Далі знаходить $p_1 = \frac{N}{\text{загальна к-сть пар}}$, якщо p_1 близька до p, то це означає, що ключ вгадано правильно; якщо ж p_1 близька до 2^{-n}, то це означає, що ключ вгадано неправильно.

Означення 1.3. *Диференціал перетворення* f_k — це пара довільних двійкових векторів (α, β), для яких виконується: $f_k(z \circ \alpha) = f_k(z) \bullet \beta$.

Для кожного диференціала існує його ймовірність, яка визначається наступним чином:

Означення 1.4. *Ймовірність диференціала* (α, β) *для перетворення* f — це величина

$$DP^f_{\circ,\bullet}(\alpha, \beta) = \frac{1}{2^n} \sum_{x \in V_n} \big[f(x \circ \alpha) = f(x) \bullet \beta \big],$$

де $\big[P\big]$ — дужки Айверсона, результат яких дорівнює одиниці, якщо P — істинне, і нулю, якщо $\big[P\big]$ — хибне.

Для будь-якого шифруючого перетворення f_k виконуються такі співвідношення:

1) $\forall \beta \in V_n \setminus \{0\} : DP^{f_k}(0, \beta) = 0,$
2) $\forall \alpha \in V_n \setminus \{0\} : DP^{f_k}(\alpha, 0) = 0,$
3) $\forall \alpha \in V_n : \sum\limits_{\forall \beta \in V_n} DP^{f_k}(\alpha, \beta) = 1,$
4) $\forall \beta \in V_n : \sum\limits_{\forall \alpha \in V_n} DP^{f_k}(\alpha, \beta) = 1$

Означення 1.5. *Множиною входів* x*, що задовольняє диференціалу* (α, β) є така множина:

$$D^{f_k}_{\circ,\bullet}(\alpha, \beta) = \{x \in V_n : f_k(x \circ \alpha) = f_k(x) \bullet \beta\}$$

Надалі замість операцій $\circ$ та $\bullet$ в цій роботі буде використовуватись операція побітового додавання за модулем два ($\oplus$). Оскільки $\forall x \in M$, $\forall k \in K$, $f_k(x) = f(x \oplus k)$, тоді рівність для диференціала (α, β) можна записати як $f(x \oplus k \oplus \alpha) = f(x \oplus k) \oplus \beta$. Нехай $y = x \oplus k$, $y \in M$, тоді $f(y \oplus \alpha) = f(y) \oplus \beta$. Тобто значення ключа для нас не важливе в такій конфігурації. Тому надалі раундове перетворення будемо записувати як $f(x), x \in M$.

1.2 Диференціальний криптоаналіз на основі усічених диференціалів

Основна відмінність звичайного диференціального криптоаналізу від криптоаналізу на основі усічених диференціалів полягає в тому, що звичайний диференціальний криптоаналіз може передбачити усі n бітів після певної кількості раундів шифру. В той час криптоаналіз, що базується на усічених диференціалах, може передбачити не всі біти. У деяких випадках навіть достатньо передбачити всього один біт.

Наразі існує багато визначень усічених диференціалів, та різні атаки на шифри з їх застосуванням. Тому розглянемо деякі з них.

1.2.1 Визначення усіченого диференціала за Кнудсеном

Вперше означення усіченого диференціала увів Ларс Кнудсен [4] у 1994 році. У своїй роботі він представляв усічений диференціал як певну підпослідовність звичайного диференціала:

Означення 1.6. Нехай (α, β) є диференціалом i-того раунду шифрування, тоді така пара (α_1, β_1), що α_1 є підпослідовністю α та β_1 є підпослідовністю β буде називатись *усіченим диференціалом i-того ранду шифрування.* [4]

Оскільки маска може бути підпослідовністю одразу для декількох бітових векторів, можемо об'єднати їх у множину можливих різниць.

Означення 1.7. *Множина можливих різниць для маски* α:

$$\Delta(\alpha) = \{\alpha' \in V_n \setminus \{0\} : \alpha' \vee \alpha = \alpha\}$$

1.2.2 Визначення диференціальної характеристики

В роботі Марії Айхлзедер [3] наведено визначення характеристики диференціала. У той час, як розглядаючи диференціал, ми знаємо лише вхідну та вихідну різниці (α, β), характеристика дозволяє побачити усі проміжні різниці на виході після кожного раунду.

Означення 1.8. *Диференціальною характеристикою* диференціала (α, β) є послідовність різниць $(\alpha_0, \alpha_1, ..., \alpha_r)$, де $\alpha_0 = \alpha$ та $\alpha_r = \beta$, та $\forall i$, $f(x \oplus \alpha_i) = f(x) \oplus \alpha_{i+1}$.

Тобто кожна проміжна різниця є різницею виходу для попереднього раундового перетворення та різницею входу для наступного раундового перетворення.

Теорема 1.1. [3] В r-раундових шифрах з незалежними раундовими ключами,

$$EDP(\alpha_0 \xrightarrow{R_0} \alpha_1 \to ... \to \alpha_{r-1} \xrightarrow{R_{r-1}} \alpha_r) = \prod_{i=0}^{r-1} TDP^{R_i}(\alpha_i, \alpha_{i+1}),$$

що є ймовірністю диференціальної характеристики $(\alpha_0, \alpha_1, \ldots, \alpha_{r-1}, \alpha_r)$ при раундових шифруючих перетвореннях $R_0, R_1, \ldots, R_{r-1}$.

1.2.3 Визначення розширеного усіченого диференціала

В роботі О. Якимчука [10] було запропоновано альтернативний вигляд усіченого диференціала:

Означення 1.9. *Усіченим диференціалом спеціального виду* називається пара векторів (α, β), де $\alpha, \beta \in V_n^* = \{0, 1, *\}^n$.

В такому випадку $\Delta(\alpha)$ буде містити такі різниці α':

1) якщо в α на певній позиції стоїть 0, то і в α' на тій же позиції стоїть 0;

2) якщо в α на певній позиції стоїть 1, то і в α' на тій же позиції стоїть 1;

3) якщо в α на певній позиції стоїть $*$, то і в α' на тій же позиції може бути як 0, так і 1.

В подальшій роботі буде використовуватись саме цей вид усіченого диференціала задля перевірки можливості його використання для знаходження високоймовірних диференціальних характеристик симетричних блокових шифрів.

Означення 1.10. *Ймовірністю усіченого диференціала* (α, β) при шифруючому перетворенні f визначається такою формулою:

$$TDP^f(\alpha,\beta) = \frac{1}{2^n}\sum_{x\in V_n}\left[\forall\alpha^{'} \in \Delta(\alpha), \exists\beta^{'} \in \Delta(\beta) : S(x\oplus\alpha^{'}) = S(x)\oplus\beta^{'}\right] \tag{1.1}$$

З цього можемо знайти множину всіх вхідних текстів, що відповідають усіченому диференціалу (α, β):

$$TD^f(\alpha,\beta) = \{x \in V_n | \forall\alpha^{'} \in \Delta(\alpha), \exists\beta^{'} \in \Delta(\beta) : f(x\oplus\alpha^{'}) = f(x)\oplus\beta^{'}\}$$

Лема 1.1. [10] *Якщо* $\alpha = 0$ *і* $\beta \in \{0,*\}^n$, *то* $TDP^S(\alpha,\beta) = 1$. *Для усіх інших* β: $TDP^S(\alpha,\beta) = 0$.

Лема 1.2. [10] $\forall\alpha \in V_n^*$:

$$TDP^S(\alpha, ***...*) = 1.$$

Лема 1.3. [10] *Якщо* $\alpha \in \{0,1\}^n$ *та* $\beta \in \{0,1,*\}^n$, *то:*

$$TDP^S(\alpha,\beta) \geq \max_{\alpha'\in\Delta(\alpha),\beta'\in\Delta(\beta)} DP^S(\alpha^{'},\beta^{'}).$$

Означення 1.11. *Маска* α_1 *домінує над маскою* α_2, якщо $\forall i$ виконується одне з двох тверджень:

1) $\alpha_{1_i} = \alpha_{2_i}$,

2) $\alpha_{1_i} = *, \alpha_{2_i} \in \{0,1\}$.

Лема 1.4. [10] *Для будь-яких* $\beta_1, \beta_2 \in V_n^*$*, якщо* β_1 *домінує над* $\beta 2$*, то для будь-якої маски* α *виконується таке співвідношення:*

$$TDP^S(\alpha, \beta_1) \geq TDP^S(\alpha, \beta_2),$$

при чому $TD^S(\alpha, \beta_1) \supseteq TD^S(\alpha, \beta_2)$.

1.3 Специфікації шифрів

Для даної роботи були обрані шифри $PRESENT$ та $GIFT$-64, як представники легких блокових шифрів, що мають просту структуру шару перестановки, та $LBlock$, як представника легких блокових шифрів, що має у своїй структурі операцію побітового додавання за модулем два.

1.3.1 Шифр $PRESENT$

Шифр $PRESENT$ [2] — це 31 раундова 64-бітна SP-мережа, один раунд якої складається з 16-ти 4-бітових S-блоків, заданих таблицею 1.1, та лінійного перетворення, заданого таблицею 1.2.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Таблиця 1.1 – S-блок шифру $PRESENT$

1.3.2 Специфікація шифру $GIFT-64$

Шифр $GIFT$-64 [1] - це 28 раундова SP-мережа з розміром входу 64 біти. Кожен раунд складається з 16 S-блоків, заданих таблицею 1.3 та лінійного перетворення P заданого таблицею 1.4.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Таблиця 1.2 – Лінійне перетворення шифру $PRESENT$

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$GS(x)$	1	A	4	C	6	F	3	9	2	D	B	7	5	0	8	E

Таблиця 1.3 – S-блок шифру $GIFT$-64

1.3.3 Специфікація шифру *LBlock*

LBlock [7] — 32 раундовий шифр з довжиною вхідного тексту 64 біт, що має структуру на основі схеми Фейстеля, що зображено на Рис. 1.1, де функція F має структуру, зображену на рисунку 1.2. S-блоки шифру *LBlock* задаються таблицею 1.5.

1.4 Приклади успішних атак з використанням диференціального криптоаналізу на основі усічених диференціалів

Приклад 1.1. *Атака на 5 раундів шифру $MIDORI$-64.*

На рисунку 1.3 зображено урізану характеристику для 5-раундового шифру $MIDORI$-64 [5]. Ймовірність цієї характеристики

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	17	34	51	48	1	18	35	32	49	2	19	16	33	50	3
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	21	38	55	52	5	22	39	36	53	6	23	20	37	54	7
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	25	42	59	56	9	26	43	40	57	10	27	24	41	58	11
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	29	46	63	60	13	30	47	44	61	14	31	28	45	62	15

Таблиця 1.4 – Лінійне перетворення шифру $GIFT$-64

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0(x)$	E	9	F	0	D	4	A	B	1	2	8	3	7	6	C	5
$S_1(x)$	4	B	E	9	F	D	0	A	7	C	5	6	2	8	1	3
$S_2(x)$	1	E	7	C	F	D	0	6	B	5	9	3	2	4	8	A
$S_3(x)$	7	6	8	B	0	F	3	E	9	A	C	D	5	2	4	1
$S_4(x)$	E	5	F	0	7	2	C	D	1	8	4	9	B	A	6	3
$S_5(x)$	2	D	B	C	F	E	0	9	7	A	6	3	1	8	4	5
$S_6(x)$	B	9	4	E	0	F	A	D	6	C	5	7	3	8	1	2
$S_7(x)$	D	A	F	0	E	4	9	B	2	1	8	3	7	5	C	6

Таблиця 1.5 – S-блоки шифру $LBlock$

$EDP(\mathcal{U}_{\delta_0} \to \mathcal{V}_{\delta_4} | (\delta_0, ..., \delta_4)$ буде дорівнювати:

$$15^{-3} \cdot \sum (\mathbb{P}(a_0 \xrightarrow{S} x_0) \cdot \mathbb{P}(a_1 \xrightarrow{S} x_0) \cdot \mathbb{P}(a_2 \xrightarrow{S} x_0) \cdot \mathbb{P}(x_0 \xrightarrow{S} x_1) \cdot \mathbb{P}(x_1 \xrightarrow{S} x_2) \cdot$$

$$\mathbb{P}(x_1 \xrightarrow{S} x_3) \cdot \mathbb{P}(x_1 \xrightarrow{S} x_4) \cdot \mathbb{P}(x_4 \xrightarrow{S} x_5) \cdot \mathbb{P}(x_4 \xrightarrow{S} x_6) \cdot \mathbb{P}(x_4 \xrightarrow{S} x_7) \cdot$$

$$\mathbb{P}(x_2 \xrightarrow{S} x_8) \cdot \mathbb{P}(x_2 \xrightarrow{S} x_9) \cdot \mathbb{P}(x_2 \xrightarrow{S} x_6) \cdot \mathbb{P}(x_3 \xrightarrow{S} x_9) \cdot \mathbb{P}(x_3 \xrightarrow{S} x_6) \cdot$$

$$\mathbb{P}(x_3 \xrightarrow{S} x_5) \cdot \mathbb{P}(x_5 \xrightarrow{S} c_0) \cdot \mathbb{P}(x_5 \xrightarrow{S} c_1) \cdot \mathbb{P}(x_6 \xrightarrow{S} c_2) \cdot \mathbb{P}(x_9 \xrightarrow{S} c_3) \cdot$$

$$\mathbb{P}(x_9 \xrightarrow{S} c_4) \cdot \mathbb{P}(x_8 \xrightarrow{S} c_7) \cdot \mathbb{P}(x_7 \xrightarrow{S} c_8) \cdot \mathbb{P}(x_7 \oplus x_8 \xrightarrow{S} c_5) \cdot \mathbb{P}(x_7 \oplus x_8 \xrightarrow{S} c_6)),$$

де $a_0, a_1, a_2, x_0, ..., x_9, c_0, ..., c_8 \in \mathbb{F}_{2^4} \setminus \{0\}$.

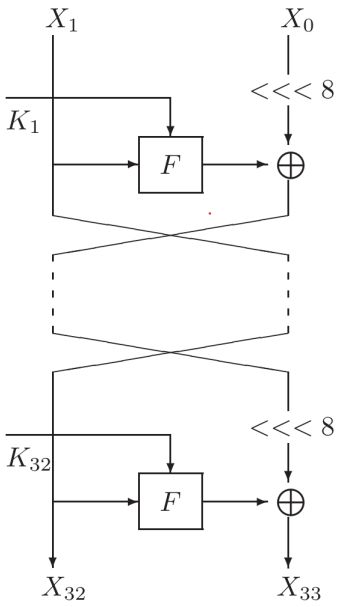

Рисунок 1.1 – Схема шифрування *LBlock*

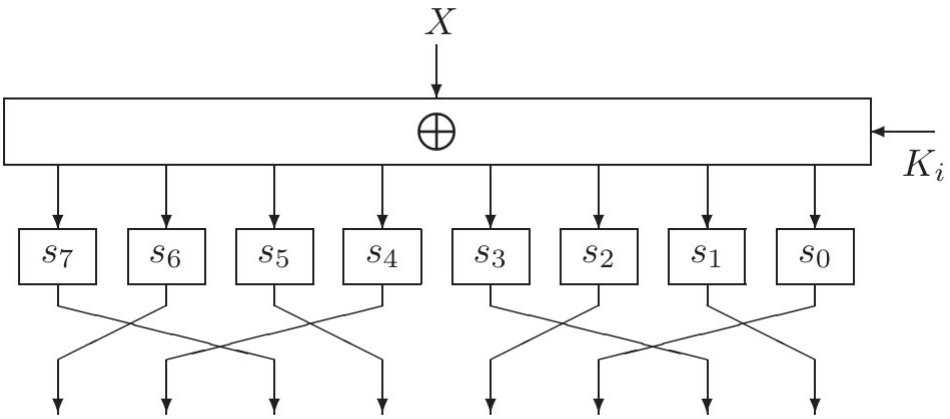

Рисунок 1.2 – Раундова функція F

Щоб обчислити точне значення EDP, потрібно зробити 25 пошуків по таблиці та арифметичних операцій для кожного 15^{22} значень $a_0, ..., c_8$, що приблизно $2^{90.60}$ операцій.

З іншої сторони, використовуючи метод Моріай та ін., достатньо перемножити $\mathbb{P}(\delta_0 \xrightarrow{\delta\mathcal{L}} \delta_1) = 15^{-2}$, $\mathbb{P}(\delta_1 \xrightarrow{\delta\mathcal{L}} \delta_2) = 1$, $\mathbb{P}(\delta_2 \xrightarrow{\delta\mathcal{L}} \delta_3) = 1$, $\mathbb{P}(\delta_3 \xrightarrow{\delta\mathcal{L}} \delta_4) = 14 \cdot 15^{-5}$, що дає в результаті $14 \cdot 15^{-7} \approx 2^{-23.54}$.

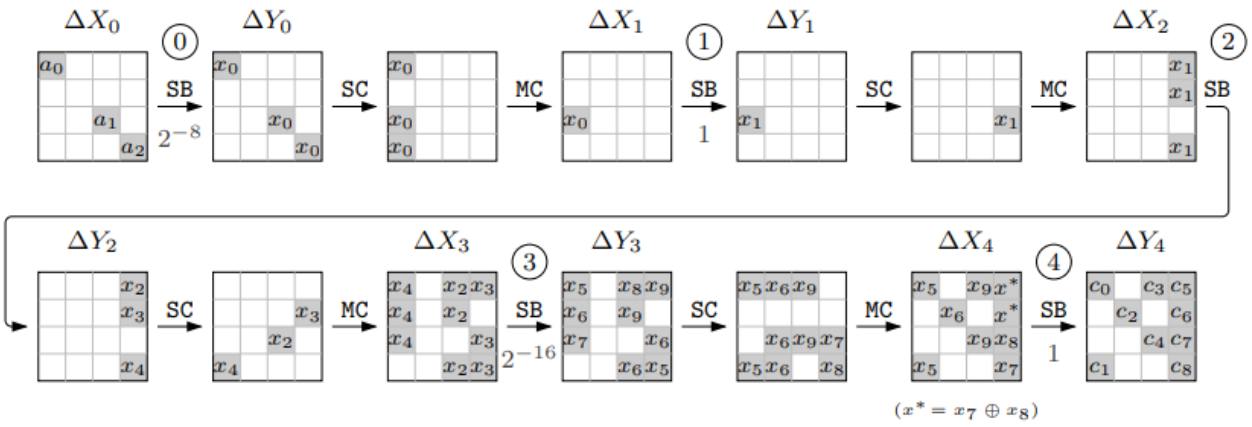

Рисунок 1.3 – Диференціальна характеристика для 5 раундів шифру $MIDORI$-64 [5]

Висновки до розділу 1

В цьому розділі було наведено необхідні теоретичні відомості для проведення подальшої роботи. Були розглянуті різні підходи до визначення усіченого диференціала. Наведено специфікації шифрів $PRESENT$, $GIFT-64$ та $LBlock$. Також було наведено приклад успішної атаки на 5 раундів шифру $MIDORI$-64 з використанням усічених диференціалів.

Підхід до визначення усічених диференціалів на основі масок спеціального, запропонований у роботі [10] є перспективним, та його можливо розширити і необхідно перевірити на реальних шифрах.

2 ПЕРЕВІРКА СТІЙКОСТІ ШИФРІВ *PRESENT* ТА *GIFT*-64 ДО ДИФЕРЕНЦІАЛЬНОГО КРИПТОАНАЛІЗУ НА ОСНОВІ УСІЧЕНИХ ДИФЕРЕНЦІАЛІВ СПЕЦІАЛЬНОГО ВИДУ

У цьому розділі буде перевірено застосовність методу гілок та границь для знаходження високоймовірних усічених диференціалів спеціального виду. Також буде проведено роботу по знаходженню високоймовірних диференціальних характеристик для шифрів *PRESENT* та *GIFT*-64.

2.1 Пошук високоймовірних усічених диференціалів методом гілок та границь

Часто для пошуку диференціалів з високою ймовірністю використовується метод гілок і границь. Його суть полягає в оптимізації повного перебору розв'язків дерева рішень до перебору лише гілок дерева, які відповідають певному критерію. В диференціальному криптоаналізі в методі гілок і границь коренем дерева є вхідна різниця, кожне ребро — це один раунд шифрування, а критерій, за яким відкидаються певні шляхи в дереві — це ймовірність диференціалу. Ймовірність диференціалу для одного раунду шифрування зазвичай легко обрахувати. Ймовірність диференціалу після кількох раундів шифрування для однієї гілки є добутком ймовірностей всіх диференціалів окремих раундів для цієї гілки дерева. У випадку, коли для одного не першого раунду шифрування різні вхідні різниці переходять в одну вихідну різницю, тоді ймовірність диференціалу з кореня дерева до цієї вихідної різниці буде сумуватися по двох різних гілках.

Візьмемо за перетворення f один раунд блокового шифру, який складається з двох S-блоків, заданого таблицею 2.1, та лінійного перетворення P_1, заданого таблицею 2.2

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Таблиця 2.1 – S-блок

i	0	1	2	3	4	5	6	7
$P_1(i)$	0	4	1	5	2	6	3	7

Таблиця 2.2 – 8-бітове лінійне перетворення

Нехай на вхід першого раунду подається маска 0^40001. Тоді:

1) $TDP^f(0^40001, 0^40100) = 0.25$;

2) $TDP^f(0^40001, 0^40101) = 0.125$;

3) $TDP^f(0^40001, 0^4010*) = 0.375$.

Тепер подамо кожну з цих вихідних масок на вхід другого раунду і обрахуємо TDP з однією і тією ж вихідною маскою $***1*^4$:

1) $TDP^f(0^40100, ***1*^4) = 0.75$ — за визначенням TDP, це означає, що $\frac{3}{4}$ всіх відкритих текстів $x \in V_8$ переводять вхідну маску 0^40100 у вихідну — $***1*^4$;

2) $TDP^f(0^40101, ***1*^4) = 0.75$ — це означає, що також $\frac{3}{4}$ всіх відкритих текстів $x \in V_n$ переводять вхідну маску 0^40101 у вихідну — $***1*^4$;

3) $TDP^f(0^4010*, ***1*^4) = 0.5$ — а це означає, що $\frac{1}{2}$ всіх відкритих текстів $x \in V_n$ одночасно переводять і вхідну маску 0^40100 і вхідну маску 0^40101 у вихідну маску $***1*^4$, бо за введеним означенням масок, множина $\Delta(0^4010*)$ складається з двох різниць 0^40100 і 0^40101.

За методом гілок і границь для обрахунку

$TDP^{\{f,f\}}(0^40001, * * *1*^4)$ (ймовірності переходу маски 0^40001 в маску $* * *1*^4$ після двох раундів шифрування) треба просумувати всі ймовірності усічених диференціальних характеристик нижче:

1) $EDP(0^40001 \xrightarrow{f} 0^40100 \xrightarrow{f} * * *1*^4) = 0.25 \times 0.75 = 0.1875$;
2) $EDP(0^40001 \xrightarrow{f} 0^40101 \xrightarrow{f} * * *1*^4) = 0.125 \times 0.75 = 0.09375$;
3) $EDP(0^40001 \xrightarrow{f} 0^4010* \xrightarrow{f} * * *1*^4) = 0.375 \times 0.5 = 0.1875$.

Але так робити не можна з нашим визначенням TDP, бо $\frac{1}{2}$ всіх відкритих текстів $x \in V_n$ на другому раунді буде врахована 3 рази в кожному з переходів. Така ситуація може призвести до значення $TDP > 1$ за достатньої кількості гілок для певної вихідної маски, що призводить до втрати сенсу параметра TDP як ймовірності.

Отже, при пошуку високоймовірних усічених диференціалів методом гілок та границь було виявлено, що таким методом користуватись не можна в контексті визначених масок усічених диференціалів і ймовірності переходу певної вхідної маски у вихідну.

2.2 Пошук високоймовірних диференціальних характеристик для шифру $PRESENT$

Для практичної перевірки дієздатності запропонованого підходу визначення усічених диференціалів та оцінки стійкості шифруючих перетворень до криптоаналізу на основі усічених диференціалів такого виду, було обрано шифр $PRESENT$.

Надалі будемо позначати f як раундове перетворення цього шифру, що було описано в розділі 1.3.1.

Оскільки метод гілок та границь неможливо застосувати до усічених диференціалів, будемо розглядати диференціальні характеристики побудовані на усічених диференціалах аналогічно до звичайних диференціалів.

2.2.1 Перевірка застосовності усічених диференціалів спеціального виду до знаходження високоймовірних диференціальних характеристик для шифру $PRESENT$

Використання усічених диференціалів дає перевагу над звичайними диференціалами у пошуку найбільш ймовірної вихідної маски для одного раунду шифрування.

Розглянемо вхідну маску $0^{63}1$ та дві вихідні маски $0^{15}10^{15} * 0^{31}1$ і $0^{15}10^{15}00^{31}1$, що відрізняються одним символом на 32-му місці. Далі обрахуємо значення TDP для одного раунду шифрування шифру $PRESENT$ для введених раніше вхідної і двох різних вихідних масок:

1) $TDP^f(0^{63}1, 0^{15}10^{15} * 0^{31}1) = 0.5$;

2) $TDP^f(0^{63}1, 0^{15}10^{15}00^{31}1) = 0.25$.

Як бачимо, значення TDP є більшим для усіченого диференціала, що має вихідну маску, що містить символ $*$. Якщо маска не містить символу $*$, то вона відіграє роль звичайної різниці двох текстів.

В той час, як на наступному раунді найбільш ймовірною вихідною маскою для вхідної маски $0^{15}10^{15} * 0^{31}1$ буде маска $\beta_1 = 0^3 10^{11} 10^3 10^{11} 10^7 * 0^{11} 10^3 * 0^7 1$:

$$TDP^f(0^{15}10^{15} * 0^{31}1, \beta_1) = 0.015625,$$

а для вхідної маски $0^{15}10^{15}00^{31}1$ найімовірнішою вихідною маскою буде $\beta_2 = 0^3 10^{11} 10^3 10^{11} * 0^{19} 10^{11} 1$:

$$TDP^f(0^{15}10^{47}1, \beta_2) = 0.125.$$

Тому, порахувавши ймовірність переходу масок для кожної характеристики, отримаємо наступне:

$$EDP(0^{63}1 \xrightarrow{f} 0^{15}10^{15} * 0^{31}1 \xrightarrow{f} \beta_1) = 0.0078125,$$

$$EDP(0^{63}1 \xrightarrow{f} 0^{15}10^{47}1 \xrightarrow{f} \beta_2) = 0.03125.$$

Як бачимо, значення ймовірності для характеристики, де маска в середині характеристики містить символ $*$, є значно меншим від іншої. Тобто можемо зробити висновок, що використання усіченого диференціала в диференціальній характеристиці дає перевагу над звичайним диференціалом лише якщо маска має символ $*$ в останньому раунді.

У роботі [6] наведено диференціальну характеристику для 4 раундів шифру $PRESENT$ з найбільшою ймовірністю — 2^{-18}, при чому ймовірність диференціалу останнього раунду дорівнює 2^{-6}. Замінивши диференціал останнього раунду на усічений диференціал з більшим значенням TDP, при цьому не міняючи диференціали попередніх раундів, можна досягти більшого значення EDP для диференціальної характеристики. Наприклад, можна використати такий усічений диференціал на останньому раунді:

$$TDP^f(0^5 1010^{45} 1010^8, 010^{11}10^3 10^{11}10^{31} * 0^2) = 0.09375 = 3 \times 2^{-3}.$$

В такому випадку значення ймовірності диференціальної характеристики

$$EDP(0^{49}10^{11}10^2 \xrightarrow{f} 0^{28}10^2 10^{28}10^2 1 \xrightarrow{f} 0^{23}10^7 10^{32} \xrightarrow{f} 0^5 1010^{45} 1010^8 \xrightarrow{f}$$

$$\xrightarrow{f} 010^{11}10^3 10^{11}10^{31} * 0^2) = 3 \times 2^{-15},$$

що значно більше, ніж значення 2^{-18}, яке досягається класичними диференціалами.

В таблиці 2.3 наведено приклади двох диференціальних характеристик C_2 та C_3, які використовують усічені диференціали, і порівняно значення TDP з диференціальною характеристикою наведеною в роботі [6] — C_1.

Таблиця 2.3 – Порівняння ймовірностей диференціальних характеристик для 4 раундів шифру $PRESENT$

Раунд	C_1 [6]	TDP	C_2	TDP	C_3	TDP
I	α	1	α	1	α	1
$R1$	β_1	2^{-4}	β_1	2^{-4}	β_1^*	3×2^{-3}
$R2$	β_2	2^{-4}	β_2	2^{-4}	β_2^*	2^{-6}
$R3$	β_3	2^{-4}	β_3	2^{-4}	β_3^*	2^{-16}
$R4$	β_4	2^{-6}	β_4^*	3×2^{-3}	–	–

$\alpha = 0^{49}10^{11}10^2$
$\beta_1 = 0^{28}10^210^{28}10^21$
$\beta_1^* = 0^{28}10^210^{15} * 0^{12}10^21$
$\beta_2 = 0^{23}10^710^{32}$
$\beta_2^* = 0^710^710^710^710^710^3 * 0^310^{11} * 0^4$
$\beta_3 = 0^51010^{45}1010^8$
$\beta_3^* = (01)^60^5(10)^60^{13} * 0^3 * 0^2(10)^41 * 10^2 * 0$
$\beta_4 = 0^{49}10^{11}10^2$
$\beta_4^* = 010^{11}10^310^{11}10^{31} * 0^2$

Диференціальна характеристика C_2 побудована так, щоб диференціал лише останнього раунду був змінений відносно наведеної характеристики C_1, що дає можливість характеристиці C_2 отримати більшу ймовірність диференціальної характеристики.

Диференціальна характеристика C_3 побудована інакшим методом: на кожному раунді обирається усічений диференціал з найбільшою ймовірністю. Це призводить до отримання кращої ймовірності лише в першому раунді, а далі ймовірності стають гіршими ніж в оригінальній диференціальній характеристиці запропонованій в [6].

2.2.2 Покращення диференціальної характеристики для шифру $PRESENT$

Також у роботі [6] була наведена характеристика для 14 раундів шифру $PRESENT$, яка має ймовірність якої дорівнює 2^{-62}. Використовуючи усічені диференціали, вдалось побудувати дві покращені характеристики.

Перша характеристика являє собою таку ж, що запропоновано в роботі [6], окрім останнього раунду — в ньому було використано усічений диференціал (Табл. 2.4).

Раунд	Маска	TDP
I	$0^5 1^3 0^{45} 1^3 0^8$	
$\vdots$	$\vdots$	$\vdots$
$R13$	$0^{49} 10^{11} 10^2$	2^{-6}
$R14$	$0^{28} 10^2 10^{12} * 0^2 * 0^{12} 10^2 1$	$2^{-2.83}$

Таблиця 2.4 – Диференціальна характеристика для 14 раундів шифру $PRESENT$

Ймовірність цієї характеристики дорівнює:

$$EDP(I \xrightarrow{f} R1 \xrightarrow{f} \ldots \xrightarrow{f} R14) \approx 2^{-60.83},$$

яка в свою чергу більша за 2^{-62} при використанні звичайних диференціалів, але при цьому дозволяє передбачити таку ж кількість змінених бітів.

Друга характеристика бере за основу ту ж саму характеристику для 14 раундів з [6] та доповнюється 15 раундом з використанням усічених диференціалів (Табл. 2.5).

Ймовірність такої характеристики дорівнює:

$$EDP(I \xrightarrow{f} R1 \xrightarrow{f} \ldots \xrightarrow{f} R15) \approx 2^{-62.415}.$$

Раунд	Маска	TDP
I	$0^5 1^3 0^{45} 1^3 0^8$	
⋮	⋮	⋮
$R14$	$0^{28} 10^2 10^{28} 10^2 1$	2^{-4}
$R15$	$0^7 * 0^7 * 0^7 10^7 * 0^7 * 0^7 * 0^{16}$	$2^{-0.415}$

Таблиця 2.5 – Диференціальна характеристика для 15 раундів шифру $PRESENT$

Ця характеристика дозволяє передбачити зміну одного біту після 15 раундів шифрування, чого неможливо досягти за допомогою звичайних диференціалів через те, що тоді ймовірність такої характеристики буде меншою за 2^{-64}, що дорівнює ймовірності випадкового вибору 64-бітного двійкового вектора.

2.3 Пошук високоймовірних диференціальних характеристик для шифру $GIFT$-64 з використанням усічених диференціалів спеціального виду

Як вже було показано у розділі 2.1, через неможливість застосування методу гілок та границь для знаходження багатораундового усіченого диференціала спеціального виду, будемо використовувати диференціальні характеристики, що побудовані на усічених диференціалах спеціального виду.

Також у розділі 2.2.1 було показано, що використання усічених диференціалів спеціального виду для побудови диференціальних характеристик має перевагу над звичайними диференціалами лише за умови використання перших для останнього раунду шифрування.

У цьому розділі будемо позначати f як раундове шифруюче перетворення шифру $GIFT$-64, описаного в розділі 1.3.2.

У роботі [8] наведено диференціальну характеристику для 12 раундів

шифру $GIFT-64$, яка має ймовірність 2^{-59} та дозволяє передбачити зміну 6 біт вхідного тексту на виході.

Використовуючи усічені диференціали, вдалось покращити дану характеристику до 13 раундів:

Раунд	Маска	TDP
I	$0^4 1^2 0^{35} 1^2 0^{21}$	
⋮	⋮	⋮
$R12$	$0^3 10^8 10^{33} 10^3 10^{13}$	2^{-5}
$R13$	$*^2 0^6 * 0^4 10^3 *^2 0^6 10^4 10^3 1 * 0^6 * 0^5 10^2 10^7 *^2 0^3$	$2^{-2.245}$

Таблиця 2.6 – Диференціальна характеристика для 13 раундів шифру $GIFT-64$

Ймовірність цієї характеристики дорівнює:

$$EDP(I \xrightarrow{f} R1 \xrightarrow{f} \ldots \xrightarrow{f} R13) \approx 2^{-61.245},$$

що більше за 2^{-64}, що дорівнює ймовірності випадкового вибору 64-бітового двійкового вектора. Також ця характеристика дозволяє передбачити зміну 6 бітів вхідного тексту на виході, як і диференціальна характеристика побудована на звичайних диференціалах в роботі [8], але має протяжність на більшу кількість раундів.

Висновки до розділу 2

В цьому розділі було розглянуто метод гілок та границь для пошуку високоймовірних диференціалів та показано його незастосовність для усічених диференціалів. Було проаналізовано існуючі підходи до побудови диференціальних характеристик на шифри $PRESENT$ та $GIFT$-64.

Для шифру $PRESENT$ було запропоновано 2 покращені характеристики з використанням усічених диференціалів:

1) Заміна диференціалу останнього раунду на усічений диференціал. Це дозволило збільшити ймовірність характеристики при збереженні такої ж кількості змінених біт.

2) Доповнення існуючої 14-раундової характеристики усіченим диференціалом для 15 раунду. Ця характеристика має перевагу над тією, що використовує звичайні диференціали за рахунок протяжності на більшу кількість раундів, але дозволяє передбачити зміну лише одного біту, на відміну від чотирьох, що дозволяє 14 раундова.

Для шифру $GIFT$-64 була запропонована доповнена диференціальна характеристика, що використовує усічені диференціали. Перевагою цієї характеристики є те, що вона дозволяє передбачити зміну такої ж кількості біт на більшій кількості раундів, ніж характеристика, що побудована з використанням звичайних диференціалів.

3 ПОШУК ВИСОКОЙМОВІРНИХ ДИФЕРЕНЦІАЛЬНИХ ХАРАКТЕРИСТИК З ВИКОРИСТАННЯМ УСІЧЕНИХ ДИФЕРЕНЦІАЛІВ СПЕЦІАЛЬНОГО ВИДУ ДЛЯ ШИФРУ *LBlock*

В цьому розділі буде визначено операцію побітового додавання за модулем два для масок спеціального виду, побудовано диференціальну характеристику для 10 раундів шифру *LBlock* з використанням усічених диференціалів.

3.1 Визначення операції $\oplus$ для масок спеціального виду

Оскільки у цій роботі використовуються маски спеціального виду, які в своєму складі мають $*$, то до них неможливо застосувати звичайну операцію побітового додавання за модулем два. Тому подальша робота потребує створення нових правил побітового додавання за модулем два.

За визначенням маски спеціального виду, якщо на певній позиції стоїть знак $*$, то це означає, що на тому місці може стояти як 0, так і 1. Розглянемо наступні випадки:

$$1)\ 0 \oplus * = \begin{cases} 0 \oplus 0 = 0 \\ 0 \oplus 1 = 1 \end{cases} \Rightarrow *$$

$$2)\ 1 \oplus * = \begin{cases} 1 \oplus 0 = 1 \\ 1 \oplus 1 = 0 \end{cases} \Rightarrow *$$

$$3)\ * \oplus * = \begin{cases} * \oplus 0 = * \\ * \oplus 1 = * \end{cases} \Rightarrow *$$

Бачимо, що будь-яка взаємодія з елементом $*$, при застосуванні операції побітового додавання за модулем два, в результаті дає елемент $*$.

Таким чином, операцію побітового додавання за модулем 2 можна задати табличкою 3.1.

x	y	$x \oplus y$
0	0	0
0	1	1
0	$*$	$*$
1	0	1
1	1	0
1	$*$	$*$
$*$	0	$*$
$*$	1	$*$
$*$	$*$	$*$

Таблиця 3.1 – Операція додавання за модулем два для масок спеціального виду

3.2 Пошук диференціальних характеристик для шифру *LBlock*

Як вже було показано в розділі 2.1 метод гілок та границь не застосовний до знаходження багатораундових усічених диференціалів спеціального виду.

У даному розділі будемо позначати f як раундове шифруюче перетворення шифру $LBlock$, описаного в розділі 1.3.3.

Для побудови усіченої диференціальної характеристики було обрано вхідну маску, що змінює лише один біт — $0^{63}1$. Далі, після кожного раунду обирався усічений диференціал з найбільшою TDP.

Результатом виконаної роботи є характеристика, задана таблицею 3.2.

Раунд	Маска	TDP
I	$0^{63}1$	
$R1$	$0^{23}10^{40}$	1
$R2$	$0^{16}101 * 0^{35}10^{8}$	2^{-2}
$R3$	$0^{15}10^{8}101 * 0^{20}101 * 0^{12}$	2^{-2}
$R4$	$0^{5} * 01^{2}01 * 0^{16}1^{2} *^{2} 0^{14}10^{8}101 * 0^{4}$	$2^{-2.678}$
$R5$	$1 *^{2} 10^{3}10^{4}101 * 101 *^{2} 1 *^{2} 0^{13} * 01^{2}01 * 0^{16}1^{2}*^{2}$	2^{-7}
$R6$	β_6	2^{-8}
$R7$	β_7	2^{-14}
$R8$	$0 *^{15} 01^{2} * 1 * 0 *^{9} 01 * (0101)^{2} * 0^{2} *^{9} 10 * 1^{2} *^{4} 1^{2} * 1$	2^{-12}
$R9$	$*^{24}1^{2} *^{6} 0 *^{15} 01^{2} * 1 * 0*^{9}$	2^{-7}
$R10$	$*^{56}1^{2}*^{6}$	2^{-2}

$\beta_6 = 1^{2}0 * 01^{2} * 1 *^{3} 0^{4} *^{4} 1^{2} *^{2} 101 * 0 * 01^{2} *^{2} 10^{3}10^{4}(101*)^{2} * 1 *^{2} 0^{8}$
$\beta_7 = 01 * (01)^{2} * 0^{2} *^{9} 10 * 1^{2} *^{4} 1^{2} * 1^{3}0 * 01^{2} * 1 *^{3} 0^{4} *^{4} 1^{2} *^{2} 101 * 0 * 01$

Таблиця 3.2 – 10 раундова диференціальна характеристика для шифру *LBlock*

Ймовірність цієї характеристики становить:

$$EDP(I \xrightarrow{f} R1 \xrightarrow{f} \ldots \xrightarrow{f} R10) \approx 2^{-56.678}$$

Висновки до розділу 3

У цьому розділі було розроблено правила побітового додавання за модулем два для масок спеціального виду.

Результатом пошуку високоймовірних усічених диференціалів для шифру *LBlock* стала диференціальна характеристика з ймовірністю $2^{-56.678}$, що дозволяє передбачити зміну 2 бітів після 10 раундів шифрування.

ВИСНОВКИ

При виконанні даної роботи було проведено огляд опублікованих досліджень з усічених диференціалів та диференціального криптоаналізу сучасних блокових шифрів. В результаті проведеного аналізу було встановлено, що наразі не існує уніфікованого визначення усіченого диференціалу. Тому було обрано формалізовану теорію, запропоновану в роботі О. П. Якимчука [10] для подальшого уточнення та перевірки її спроможностей.

Було перевірено можливість застосування усічених диференціалів на основі масок спеціального виду для побудови багатораундових диференціалів та диференціальних характеристик. Було встановлено, що метод гілок та границь не є застосовним до знаходження багатораундових усічених диференціалів.

В роботі продемонстровано застосовність усічених диференціалів на основі масок спеціального вигляду на біт-орієнтовані блокові шифри $PRESENT$ і $GIFT-64$. В результаті роботи було запропоновано дві покращені диференціальні характеристики з використанням усічених диференціалів спеціального виду для шифру $PRESENT$ і одну для шифру $GIFT-64$.

Також було розроблено правила застосування операції побітового додавання за модулем два до масок, що розглядаються в усічених диференціалах спеціального виду.

Уданій роботі також продемонстровано застосовність усічених диференціалів на основі масок спеціального виду на блоковому шифрі, що використовує операцію побітового додавання за модулем два у своїй структурі. В результаті було запропоновано диференціальну характеристику з використанням усічених диференціалів до 10 раундів шифру $LBlock$.

У подальшому розвитку дослідження планується розробка більш ефективного підходу до пошуку диференціальних характеристик та застосування його для інших шифрів.

ПЕРЕЛІК ПОСИЛАНЬ

[1] Subhadeep Banik та ін. *GIFT: A Small Present*. Cryptology ePrint Archive, Paper 2017/622. 2017. URL: https://eprint.iacr.org/2017/622.

[2] Andrey Bogdanov та ін. «PRESENT: An Ultra-Lightweight Block Cipher». В: *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Т. 4727. Lecture Notes in Computer Science. Springer, 2007, с. 450—466. DOI: 10.1007/978-3-540-74735-2_31.

[3] Maria Eichlseder, Gregor Leander та Shahram Rasoolzadeh. *Computing Expected Differential Probability of (Truncated) Differentials and Expected Linear Potential of (Multidimensional) Linear Hulls in SPN Block Ciphers*. Cryptology ePrint Archive, Paper 2020/1356. 2020. URL: https://eprint.iacr.org/2020/1356.

[4] Knudsen L. *Truncated and Higher Ordered Differentials*. 1008-е вид. 1994, С. 196–211. URL: https://link.springer.com/chapter/10.1007/3-540-60590-8_16.

[5] AmirHossein E. Moghaddam та Zahra Ahmadian. *New Automatic search method for Truncated-differential characteristics: Application to Midori, SKINNY and CRAFT*. Cryptology ePrint Archive, Paper 2019/126. 2019. URL: https://eprint.iacr.org/2019/126.

[6] Meiqin Wang. *Differential Cryptanalysis of PRESENT*. Cryptology ePrint Archive, Paper 2007/408. 2007. URL: https://eprint.iacr.org/2007/408.

[7] Wenling Wu та Lei Zhang. *LBlock: A Lightweight Block Cipher **. Cryptology ePrint Archive, Paper 2011/345. 2011. URL: https://eprint.iacr.org/2011/345.

[8] Baoyu Zhu, Xiaoyang Dong та Hongbo Yu. *MILP-based Differential Attack on Round-reduced GIFT*. Cryptology ePrint Archive, Paper 2018/390. 2018. URL: https://eprint.iacr.org/2018/390.

[9] К. А. Медведцький та О. П. Якимчук. «Аналіз підходів до пошуку високоймовірнісних усічених диференціалів спеціального вигляду». В: *XXII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» (Україна, м. Київ, 13-17 травня 2024 р.) : матеріали конференції*. Київ: КПІ ім. Ігоря Сікорського, Видавництво «Політехніка», 2024, с. 224—227. ISBN: 978-966-990-053-1.

[10] О. П. Якимчук. *Метод оцінювання стійкості блокових шифрів до криптоаналізу на основі усічених диференціалів : магістерська дис. : 113 Прикладна математика / Якимчук Олексій Петрович.* 2020. URL: https://ela.kpi.ua/handle/123456789/34327.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

А.1 Програма для знаходження диференціальних характеристик для шифру *PRESENT*

```
from s_box import Sbox
import side_functions as sf
from trunc_diff import truncated_differential as td
import time
import csv
import multiprocessing as mp
import constants as c
import textwrap
import copy

def one_round_with_permutation(input_chain: list[str], box_tdp_dict: dict[str, dict[str, float]]) ->

    input_masks = textwrap.wrap(input_chain[-1], 4)



    output_masks_for_concatination = {"": 1}
    con_count = 0
    for m in input_masks:
        temp = {}
        done = 0
        for key in output_masks_for_concatination:
            for output_mask in box_tdp_dict[m]:
                if key.count("*") + output_mask.count("*") - input_chain[-1].count("*") <= c.DIFF_OF
                    new_key = key + output_mask
                    new_value = output_masks_for_concatination[key] * box_tdp_dict[m][output_mask]
                    if new_value > c.LAST_ROUND_TRESHOLD:
                        temp[new_key] = new_value


            done += 1
            if done % 1000000 == 0:
                print(f"Done {done} masks out of {len(output_masks_for_concatination)}")
        output_masks_for_concatination = temp
        con_count += 1
        print(f"Concatenation done for {con_count} blocks, now {len(output_masks_for_concatination)}

    output_1_round = {}
    # perm_count = 0
    # check_count = 0
    for key in output_masks_for_concatination:
```

```
            perm = sf.permutation(key, 4)
            if perm.count("1") >= 4:
                output_1_round[perm] = output_masks_for_concatination[key]
                # print(f"{perm} = {output_masks_for_concatination[key]}")



    output_to_sort = {}
    for key in output_1_round:
        new_key = copy.deepcopy(input_chain[-1])
        new_key.append(key)
        output_to_sort[tuple(new_key)] = output_1_round[key]

    sorted_data = [(output_to_sort[key], key) for key in output_to_sort]
    sorted_data.sort(reverse=True)

    output = {}
    inside = 0
    for k in sorted_data:
        output[k[1]] = k[0]
        if inside < c.TOP_NUM:
            output[k[1]] = k[0]
            inside += 1
        else:
            break

    return output

if __name__ == "__main__":
    start_time = time.time()

    cpus = mp.cpu_count()

    s_1 = Sbox(transformation = [12, 5, 6, 11, 9, 0, 10, 13, 3, 14, 15, 8, 4, 7, 1, 2])

    masks = sf.generate_all_masks(4)

    tdps = {}

    for i in masks:
        d = {}
        for j in masks:
            t = td(i, j)
            tdp = t.tdp(s_1.get)
            if (j.count("*") - i.count("*") <= c.DIFF_OF_STARS):
```

```
                if tdp > 0:
                    d[j] = tdp

            sorted_data = [(d[key], key) for key in d]
            sorted_data.sort(reverse=True)

            inside = 0
            d_2 = {}
            for k in sorted_data:
                if inside < c.TOP_NUM:
                    d_2[k[1]] = k[0]
                    inside += 1
                else:
                    break
        tdps[i] = d_2

    input_file = "present_2_round.csv"
    output_file = "present_3_round.csv"

    tdps_m = {}

    with open(input_file, "r") as f:
        reader = csv.reader(f)
        for row in reader:
            value = float(row[-1])
            row.pop()
            tdps_m[tuple(row)] = value

    tdps_to_sort = {}

    len_input_masks = len(tdps_m)
    in_pool = 0
    done = 0
    to_pool = []
    for n, i in enumerate(tdps_m):
        if in_pool < cpus:
            to_pool.append((list(i), tdps))
            in_pool += 1
            if n != len_input_masks - 1:
                continue

        in_pool = 0
        pool = mp.Pool(cpus)
        results = pool.starmap(one_round_with_permutation, to_pool)
        pool.close()
```

```
        to_pool = []
        for result in results:
            done += 1
            for m in result:
                t = result[m] * tdps_m[i]
                if t > c.PROBABILITY_TRESHOLD:
                    tdps_to_sort[m] = t

            print(f"Done␣{done}␣masks␣out␣of␣{len_input_masks}")

    sorted_data = [(tdps_to_sort[key], key) for key in tdps_to_sort]
    sorted_data.sort(reverse=True)

    with open(output_file, "w", newline="") as f:
        writer = csv.writer(f)
        for i in sorted_data:
            line = [i[1][0]]
            for j in range(1, len(i[1]), 1):
                line.append(i[1][j])
            line.append(i[0])
            writer.writerow(line)

    end_time = time.time()
    execution_time = end_time - start_time
    print(f"Execution␣time:␣{execution_time:.6f}␣seconds")
```

A.2 Програма для знаходження диференціальних характеристик для шифру $GIFT$-64

```
from s_box import Sbox
import side_functions as sf
from trunc_diff import truncated_differential as td
import time
import csv
import multiprocessing as mp
import constants as c
import textwrap
import copy

def one_round_gift(input_mask: list, tdps: dict[str, dict[str, float]]) -> dict[str, float]:

    perm = [12, 1, 6, 11, 28, 17, 22, 27, 44, 33, 38, 43, 60, 49, 54, 59, 8, 13, 2, 7, 24, 29, 18, 23
```

```
input = input_mask[-1]
sbox_inputs = textwrap.wrap(input, 4)

output_for_concatination = {"": 1}

con_count = 0
for m in sbox_inputs:
    temp = {}
    done = 0
    for key in output_for_concatination:
        for output_mask in tdps[m]:
            if key.count("*") + output_mask.count("*") - input.count("*") <= c.DIFF_OF_STARS_CONC
                new_key = key + output_mask
                new_value = output_for_concatination[key] * tdps[m][output_mask]
                if new_value > c.LAST_ROUND_TRESHOLD:
                    temp[new_key] = new_value
        done += 1
        if done % 1000000 == 0:
            print(f"Done {done} masks out of {len(output_for_concatination)}")
    output_for_concatination = temp
    con_count += 1
    print(f"Concatenation done for {con_count} blocks, now {len(output_for_concatination)} masks

output = {}
for key in output_for_concatination:
    out = ""
    for k in perm:
        out += key[k]
    output[out] = output_for_concatination[key]

output_to_sort = {}
for key in output:
    new_key = copy.deepcopy(input_mask)
    new_key.append(key)
    output_to_sort[tuple(new_key)] = output[key]

sorted_data = [(output_to_sort[key], key) for key in output_to_sort]
sorted_data.sort(reverse=True)

output = {}
inside = 0
for k in sorted_data:
    if inside < c.TOP_NUM_MASKS:
        output[k[1]] = k[0]
```

```
            inside += 1
        else:
            break

    return output


if __name__ == "__main__":
    start_time = time.time()

    cpus = mp.cpu_count()

    s_gift = Sbox(transformation=[1, 10, 4, 12, 6, 15, 3, 9, 2, 13, 11, 7, 5, 0, 8, 14])

    masks = sf.generate_all_masks(4)

    tdps = {}

    for i in masks:
        d = {}
        for j in masks:
            t = td(i, j)
            tdp = t.tdp(s_gift.get)
            if (j.count("*") - i.count("*") <= c.DIFF_OF_STARS):
                if tdp > 0:
                    d[j] = tdp

            sorted_data = [(d[key], key) for key in d]
            sorted_data.sort(reverse=True)

            inside = 0
            d_2 = {}
            for k in sorted_data:
                if inside < c.TOP_NUM:
                    d_2[k[1]] = k[0]
                    inside += 1
                else:
                    break
        tdps[i] = d_2

    input_file = "gift_2_round.csv"
    output_file = "gift_3_round.csv"

    tdps_m = {}
```

```
    with open(input_file, "r") as f:
        reader = csv.reader(f)
        for row in reader:
            value = float(row[-1])
            row.pop()
            tdps_m[tuple(row)] = value

    tdps_to_sort = {}

    len_input_masks = len(tdps_m)
    in_pool = 0
    done = 0
    to_pool = []
    for n, i in enumerate(tdps_m):
        if in_pool < cpus:
            to_pool.append((list(i), tdps))
            in_pool += 1
            if n != len_input_masks - 1:
                continue

        in_pool = 0
        pool = mp.Pool(cpus)
        results = pool.starmap(one_round_gift, to_pool)
        pool.close()
        to_pool = []
        for result in results:
            done += 1
            for m in result:
                t = result[m] * tdps_m[i]
                if t > c.PROBABILITY_TRESHOLD:
                    tdps_to_sort[m] = t

            print(f"Done␣{done}␣masks␣out␣of␣{len_input_masks}")

    sorted_data = [(tdps_to_sort[key], key) for key in tdps_to_sort]
    sorted_data.sort(reverse=True)

    with open(output_file, "w", newline="") as f:
        writer = csv.writer(f)
        for i in sorted_data:
            line = [i[1][0]]
            for j in range(1, len(i[1]), 1):
                line.append(i[1][j])
            line.append(i[0])
            writer.writerow(line)
```

```
    end_time = time.time()
    execution_time = end_time - start_time
    print(f"Execution time: {execution_time:.6f} seconds")
```

A.3 Програма для пошуку диференціальних характеристик для шифру *LBlock*

```
from s_box import Sbox
import side_functions as sf
from trunc_diff import truncated_differential as td
import time
import csv
import multiprocessing as mp
import constants as c
import textwrap
import copy

def xor(a: str, b: str) -> str:
    if len(a) != len(b):
        raise ValueError("Length of a and b must be equal")

    output = ""
    for i in range(len(a)):
        if a[i] == "*" or b[i] == "*":
            output += "*"
            continue
        if a[i] == b[i]:
            output += "0"
        else:
            output += "1"

    return output

def one_round_feistel(input_mask: list, tdps: list[dict[str, dict[str, float]]]) -> dict[str, float]:
    if len(input_mask[-1]) != 64:
        raise ValueError("Input must be 64 bits long")

    reverse_transistion = [6, 4, 7, 5, 2, 0, 3, 1]
```

```
    def left_circle_shift(input: str, shift: int) -> str:
        if len(input) != 32:
            raise ValueError("Input must be 32 bits long")

        return input[shift:] + input[:shift]

    left = input_mask[-1][:32]
    right = input_mask[-1][32:]

    def F(input: str) -> dict[str, float]:
        if len(input) != 32:
            raise ValueError("Input must be 32 bits long")

        input_masks = textwrap.wrap(input, 4)
        input_masks.reverse()
        output_masks = {"": 1}

        for i in range(len(input_masks)):
            temp = {}
            for key in output_masks:
                for output_mask in tdps[reverse_transistion[i]][input_masks[reverse_transistion[i]]]:
                    if key.count("*") + output_mask.count("*") - input.count("*") <= c.DIFF_OF_STARS_
                        new_key = key + output_mask
                        new_value = output_masks[key] * tdps[reverse_transistion[i]][input_masks[reve
                        temp[new_key] = new_value
            output_masks = temp

        return output_masks

    left_output = F(left)

    round_output = {}

    right = left_circle_shift(right, 8)
    for key in left_output:
        new_key = copy.deepcopy(input_mask)
        new_key.append(xor(key, right) + left)
        round_output[tuple(new_key)] = left_output[key]

    sorted_data = [(round_output[key], key) for key in round_output]
    sorted_data.sort(reverse=True)

    output = {}
    inside = 0
    for k in sorted_data:
```

```
        if inside < c.TOP_NUM_MASKS:
            output[k[1]] = k[0]
            inside += 1
        else:
            break

    return output

if __name__ == "__main__":
    start_time = time.time()

    s_0 = Sbox(transformation= [14, 9, 15, 0, 13, 4, 10, 11, 1, 2, 8, 3, 7, 6, 12, 5])
    s_1 = Sbox(transformation= [4, 11, 14, 9, 15, 13, 0, 10, 7, 12, 5, 6, 2, 8, 1, 3])
    s_2 = Sbox(transformation= [1, 14, 7, 12, 15, 13, 0, 6, 11, 5, 9, 3, 2, 4, 8, 10])
    s_3 = Sbox(transformation= [7, 6, 8, 11, 0, 15, 3, 14, 9, 10, 12, 13, 5, 2, 4, 1])
    s_4 = Sbox(transformation= [14, 5, 15, 0, 7, 2, 12, 13, 1, 8, 4, 9, 11, 10, 6, 3])
    s_5 = Sbox(transformation= [2, 13, 11, 12, 15, 14, 0, 9, 7, 10, 6, 3, 1, 8, 4, 5])
    s_6 = Sbox(transformation= [11, 9, 4, 14, 0, 15, 10, 13, 6, 12, 5, 7, 3, 8, 1, 2])
    s_7 = Sbox(transformation= [13, 10, 15, 0, 14, 4, 9, 11, 2, 1, 8, 3, 7, 5, 12, 6])

    masks = sf.generate_all_masks(4)

    sbox_tdps = [{}, {}, {}, {}, {}, {}, {}, {}]

    for i in masks:
        d = [{}, {}, {}, {}, {}, {}, {}, {}]
        for j in masks:
            t = td(i, j)
            tdp_0 = t.tdp(s_0.get)
            tdp_1 = t.tdp(s_1.get)
            tdp_2 = t.tdp(s_2.get)
            tdp_3 = t.tdp(s_3.get)
            tdp_4 = t.tdp(s_4.get)
            tdp_5 = t.tdp(s_5.get)
            tdp_6 = t.tdp(s_6.get)
            tdp_7 = t.tdp(s_7.get)
            if (j.count("*") - i.count("*") <= c.DIFF_OF_STARS):
                if tdp_0 > 0:
                    d[0][j] = tdp_0
                if tdp_1 > 0:
                    d[1][j] = tdp_1
                if tdp_2 > 0:
                    d[2][j] = tdp_2
                if tdp_3 > 0:
                    d[3][j] = tdp_3
```

```
            if tdp_4 > 0:
                d[4][j] = tdp_4
            if tdp_5 > 0:
                d[5][j] = tdp_5
            if tdp_6 > 0:
                d[6][j] = tdp_6
            if tdp_7 > 0:
                d[7][j] = tdp_7

    for k in range(8):
        sorted_data = [(d[k][key], key) for key in d[k]]
        sorted_data.sort(reverse=True)
        inside = 0
        d_2 = {}
        if i == "0000":
            d_2["0000"] = 1.0
            sbox_tdps[k][i] = d_2
            continue
        for l in sorted_data:
            if inside < c.TOP_NUM:
                d_2[l[1]] = l[0]
                inside += 1
            else:
                break
        sbox_tdps[k][i] = d_2

input_file = "feistel_9_round.csv"
output_file = "feistel_10_round.csv"

tdps = {}

with open(input_file, "r") as f:
    reader = csv.reader(f)
    for row in reader:
        value = float(row[-1])
        row.pop()
        tdps[tuple(row)] = value

tdps_to_sort = {}

for i in tdps:
    round_tdps = one_round_feistel(list(i), sbox_tdps)

    for m in round_tdps:
        t = round_tdps[m] * tdps[i]
```

```
            if t > c.PROBABILITY_TRESHOLD:
                tdps_to_sort[m] = t

    sorted_data = [(tdps_to_sort[key], key) for key in tdps_to_sort]
    sorted_data.sort(reverse=True)

    with open(output_file, "w", newline="") as f:
        writer = csv.writer(f)
        for i in sorted_data:
            line = [i[1][0]]
            for j in range(1, len(i[1]), 1):
                line.append(i[1][j])
            line.append(i[0])
            writer.writerow(line)
```