МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«МИРЭА – Российский технологический университет»**
**РТУ МИРЭА**

---

Институт искусственного интеллекта
Базовая кафедра №252 – информационной безопасности

---

# КУРСОВАЯ РАБОТА

По дисциплине
«Криптографические методы защиты информации»

Тема курсовой работы
**«Эллиптические кривые»**

| | | |
|---|---|---|
| Студент группы ККСО-03-19 | Николенко В.О. | _____ *(подпись)* |
| Руководитель курсовой работы | Бондакова С.С. | _____ *(подпись)* |

Работа представлена к защите «__» _____ 2022 г.

Допущен к защите «__» _____ 2022 г.

Москва – 2023

# СОДЕРЖАНИЕ

# 1. ВВЕДЕНИЕ

В этой статье представлен улучшенный анализ прообраза на круглом уменьшенном Keccak-384/512. В отличие от версий малой емкости, Keccak-384/512 выводит данные из двух частей своего состояния: всей 320-битной плоскости и 64/192- битного усечения второй плоскости. Из-за отсутствия степеней свободы большинство существующих методов анализа прообразов могут управлять только первой 320-битной плоскостью и достигать ограниченных результатов. Тщательно проанализировав алгебраическую структуру Keccak, в этой статье предлагается технология под названием “дополнительная линейная зависимость”, которая может строить линейные соотношения между соответствующими битами с двух плоскостей. Чтобы применить технологию, эта статья наследует идеи пионеров в области атак, которые преобразуют выходные биты в линейные или квадратные уравнения входных переменных. При решении конечной системы уравнений эти линейные соотношения могут привести к дополнительным ограничивающим уравнениям вывода, превышающим предел ранга матрицы. В результате сложность атак с использованием прообразов на 2-раундовый и 3-раундовый Keccak-384/512 может быть уменьшен до $2^{39}/2^{204}$ и $2^{270}/2^{424}$ вызова Keccak соответственно, и это все наиболее известные результаты на данный момент. В поддержку теоретического анализа в данной статье приводится первый предварительный просмотр всего дайджеста "0"для 2-раундового Keccak-384, который можно получить за один день с помощью одноядерного процессора на обычном ПК.

Функция Keccak, разработанная Бертони и др. [1,2], была выбрана победителем конкурса SHA-3 в 2012 году и окончательно стандартизирована NIST в 2015 году. С Keccak был предложен в 2008 году, в общественном исследовательском сообществе были предложены различные виды криптоанализа безопасности , включая прообраз [3,4,5], столкновение [6,7,8], различение [9,10,11], режимы с ключом [12,13,14] и многие другие не упомянутые настройки безопасности. Эти передовые методы атаки хорошо работают даже с практическими в результате получается keccak малой вместимости: round-reduced Keccak-224/256. Тем не менее, для Keccak-384/512 с уменьшенным объемом памяти из-за отсутствия свободы в настройке блоков сообщений большинство методов не могут работать так эффективно, как в версиях с малой емкостью.

В этой статье мы в основном сосредоточимся на атаках с использованием прообразов на Keccak 384/512 с уменьшенным округлением - более конкретно, на линейном анализе. Наше исследование вдохновлено несколькими творческими работами, краткое описание которых приведено ниже. В

2016 году Го и др. [15] впервые применили стратегию под названием linear structure в атаках на прообразы в версиях Keccak с уменьшенным количеством раундов. Их идея состоит в том, чтобы линеаризовать все государство после нескольких раундов с частично оставленным пространством свободы. Однако для Keccak-384/512 их линейные конструкции могут проходить только 1 раунд, и поэтому им пришлось применить другие передовые технологии для достижения хорошие результаты, которые не требуются в данной работе. Затем, в 2019 году, Ли и Сун [16] улучшена линейная структура с помощью технологии, названной моделью распределения: первый блок сообщений направлен на генерацию ограниченного среднего состояния, удовлетворяющего определенным условиям, так что второй блок сообщений (XORed с ограниченным средним состоянием) может получить дополнительное пространство свободы при поиске прообраза. Они применили эту модель только на уменьшенном в размерах Keccak-224/256, в то время как она также может быть применена на 3-х раундовый Keccak-384 (мы приведем простую конструкцию в разделе 3.1). Раджасри [17] внесла улучшение с другой точки зрения. Он заметил, что количество оставшихся степеней свободы намного меньше, чем количество (линейных) выходных уравнений. Таким образом, он допустил существование нелинейных частей в структуре и просто построил выходные уравнения для линейных частей. Эта идея не увеличила бы пространство свободы, но увеличила бы пространство случайных констант, что также является заметной проблемой версий с высокой производительностью. В 2021 году он и др. [18] предложили технологию под названием zero coefficient. Это относится к некоторым линейно-зависимым парам битов в состоянии Keccak. С помощью с помощью этой технологии они успешно удовлетворили 173 уравнения всего с 162 степенями свободы, получив 11 линейно-зависимых пар битов. Это значение ограничено главным образом потому, что объектом их анализа является Keccak-224/256. Для Keccak-384/512 (с двумя выходными плоскостями) их количество может быть увеличено до уровня полосы движения. Последним анализом прообразов для Keccak-384/512 с уменьшенным раундом является исследование Liu et al [19], в основном из которого мы унаследовали фреймворки атак. Они также допускали существование нелинейных частей в структуре — в отличие от [17], они действительно могут увеличьте пространство свободы, но должны иметь дело с полностью квадратичным выходным состоянием. К счастью, они обнаружили, что, используя алгебраическую структуру Keccak, метод релинеаризации может быть применен к конечной системе уравнений. Этот метод подходит только для частного случая системы квадратных уравнений, где число уравнений намного больше, чем число различных квадратных членов. Используя этот метод, простран-

ство свободы может принести эквивалентный выигрыш при поиске по прообразу. Другими словами, n степеней свободы могут принести выигрыш в 2n при поиске по прообразу, идентично случаю системы линейных уравнений. Результаты анализа прообразов для Keccak-384/512 с круглым уменьшением, упомянутые выше3, обобщены в таблице 1.

Наш вклад. В этой статье предлагается новая технология под названием extra linear dependence для улучшения атак на прообразы на Keccak-384/512 с уменьшенным количеством раундов. Технология направлена на построение линейных соотношений между соответствующими битами из две выходные плоскости, так что конечная система уравнений может быть решена даже тогда, когда количество уравнений больше, чем количество переменных. В этом случае n степеней свободы могут принести выигрыш даже больший, чем 2n при поиске по прообразу. Чтобы применить эту технологию, мы наследуем (и слегка модифицируем) новейшие квадратичные структуры в [19]. В результате мы успешно построили 128 линейно-зависимых битовых пар во 2-м раунде Keccak-384/512 и 24 линейно-зависимых битовых пары в 3-м раунде Keccak-384, который может уменьшить сложность поиска (время угадывания) на 264/212. Что касается 3-раундового Keccak-512, то из-за отсутствия контролируемых сумм столбцов вряд ли может быть применена дополнительная линейная зависимость. Тем не менее, мы все еще добиваемся прогресса, исправляя упущение в применении метода релинеаризации и улучшая квадратичную структуру. Чтобы поддержать наш анализ, мы сначала предоставляем фактический прообраз (соответствующий правилу заполнения) всего дайджеста "0"для 2-раундового Keccak-384. Сравнения между нашими результатами и предыдущими результатами приведены в таблице 1.

| Variant | Guessing Times | Size[a] | Solving Time | Final Complexity | Reference |
|---|---|---|---|---|---|
| 2-round Keccak-384 | $2^{129}$ | \ | \ | \ | [15][b] |
| | $2^{113}$ | \ | \ | \ | [17][b] |
| | $2^{93}$ | 384 | $2^{11}$ | $2^{104}$ | [19][c] |
| | $2^{28}$ | 320 | $2^{11}$ | $2^{39}$ | Sect. 5.1 |
| 2-round Keccak-512 | $2^{384}$ | \ | \ | \ | [15][b] |
| | $2^{321}$ | \ | \ | \ | [17][b] |
| | $2^{258}$ | 448 | $2^{12}$ | $2^{270}$ | [19][c] |
| | $2^{193}$ | 384 | $2^{11}$ | $2^{204}$ | Sect. 5.2 |
| 3-round Keccak-384 | $2^{322}$ | \ | \ | \ | [15][b] |
| | $2^{321}$ | \ | \ | \ | [17][b] |
| | $2^{271}$ | 460 | $2^{12}$ | $2^{283}$ | [19][c] |
| | $2^{258}$ | 460 | $2^{12}$ | $2^{270}$ | Sect. 5.3 |
| 3-round Keccak-512 | $2^{482}$ | \ | \ | \ | [15][b] |
| | $2^{475}$ | \ | \ | \ | [17][b] |
| | $2^{440}$ | 494 | $2^{12}$ | $2^{452}$ | [19][c] |
| | $2^{412}$ | 448 | $2^{12}$ | $2^{424}$ | Sect. 5.4 |
| 4-round Keccak-384 | $2^{371}$ | \ | \ | \ | [17][b] |
| | $2^{366}$ | 175 | $2^{9}$ | $2^{375}$ | [19][c] |

Fig. 1. The sponge construction.

[a]"Размер" - это общее количество переменных в системе уравнений после применения метода линеаризации. Для лучшего сравнения мы будем использовать тот же способ для вычисления "размера", а затем "Время решения" будет оценено в соответствии с [19]. приведенные результаты относятся к времени угадывания вместо вызовов Keccak, которые не включают сложность решения конечной системы уравнений. Авторы допустили ошибку в соответствии с правилом заполнения Keccak. Их результаты по "Времени угадывания" и "Конечной сложности" должны быть уменьшены на 2,1.

Организация. Эта статья начинается с некоторых предварительных замечаний и примечаний о Кеккак в разделе 2. Обзор идей об атаке линейного анализа на Кеккак с круглым уменьшением приведен в разделе 3. Основная технология получения дополнительной линейной зависимости объясняется в разделе 4. Улучшены атаки прообразом во 2-раундовом и 3-раундовые Keccak-384/512 представлены в разделе 5. Выводы кратко изложены в разделе 6.

## ПРЕУМБУЛА

В этом разделе приведены описания конструкции sponge, перестановки Keccak-f, стандарта SHA-3, свойств инверсии S-box и значений обозначений, используемых в этой статье.

## Спонж конструкция

Функция Keccak использует новую итеративную конструкцию под названием sponge, которая включает в себя три параметра r, c, d и перестановку Keccak-f[b] с b = r + c (как показано на рис. 1). Эта конструкция обрабатывает сообщение в две фазы — фазу поглощения и фазу сжатия. На этапе поглощения сообщение M (после заполнения) разбивается на r-битные блоки. Начиная с b-бита all '0' IV, его первые r-биты преобразуются в x с первым блоком сообщений, за которым следует выполнение Keccak-f. После того, как все блоки сообщений будут обработаны аналогичным образом, наступает этап сжатия. На этапе сжатия конструкция выводит r-битный дайджест и смешивает его состояние , выполняя Keccak-f, повторяя до тех пор, пока длина дайджеста не достигнет d. Наконец, дайджест усекается до первых d-битов.

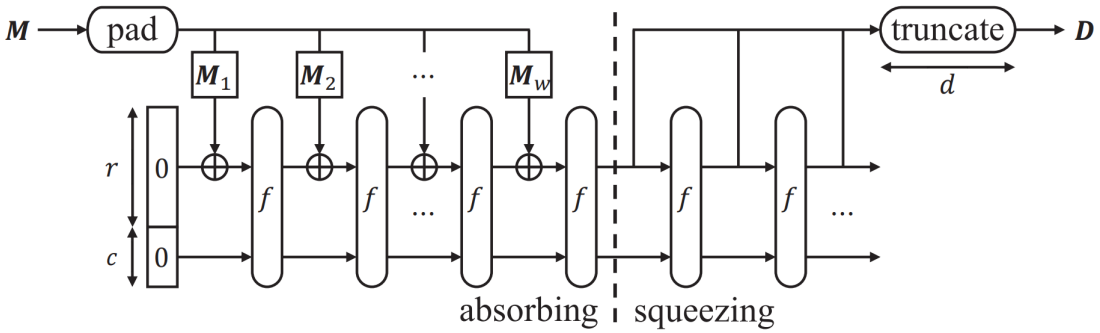


Fig. 1. The sponge construction.

## Перестановка Keccak-f

Ядром вычисления Keccak-f является его b-разрядное состояние. В [2] разработчики предусмотрели семь перестановок Keccak-f, где b ∈ 25, 50, 100, 200, 400, 800, 1600. В конце концов NIST выбрал b = 1600 в качестве стандарта SHA-3 [21]. В этой статье мы также рассмотрим только случай b = 1600.

В случае b = 1600 состояние Keccak-f можно рассматривать как 5×5 64-битных полос (как показано на рис. 2). Каждый бит обозначается как Ax,y,z, где x варьируется от от 0 до 4, y изменяется от 0 до 4, а z изменяется от 63 до 0 (считая от самого значащего бита), как указано стрелками на рис. 2. R-разрядная часть состояния складывается в порядке A0,0,0 = A0,0,63, A 1,0,0 = A1,0,63, . . . , A 4,0,0 = A4,0,63, A 0,1,0 = A0,1,63. . . Кроме того, разработчики определили некоторые компоненты состояния (также изображенные на рис. 2). Среди этих компонентов важным в данной статье является "плоскость"., который состоит из 5 полос или 64 рядов.

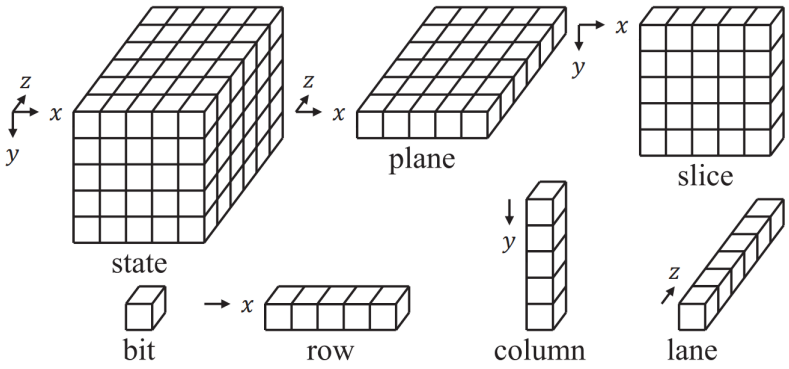


Fig. 2. The state and its components of Keccak-f.

Что касается вычисления Keccak-f, то оно состоит из 24 раундов функции R, и каждый R состоит из пяти шагов $R = i \cdot \chi \cdot \pi \cdot \rho \cdot \theta$, где:

$$\theta : A_{x,y,z} = A_{x,y,z} \oplus \bigoplus\nolimits_{j=0\sim4}(A_{x-1,j,z} \oplus A_{x+1,j,z-1})$$
$$\rho : A_{x,y,z} = A_{x,y,z-r_{x,y}}$$
$$\pi : A_{x,y,z} = A_{x+3y,x,z}$$
$$\chi : A_{x,y,z} = A_{x,y,z} \oplus (A_{x+1,y,z} \oplus 1) \cdot A_{x+2,y,z}$$
$$\iota : A_{0,0,z} = A_{0,0,z} \oplus RC_z$$

В приведенных выше формулах "$\oplus$" означает побитовое XOR, а "$\cdot$" означает побитовое AND. Индексы x и y вычисляются по модулю 5, а индекс z вычисляется по модулю 64. Кроме того, rx,y относится к константе поворота, зависящей от полосы движения, как показано в таблице 2. RC - это константа, зависящая от округления. Мы опускаем здесь детали RC, поскольку эти константы не влияют на наши методы атаки.

Table 2. The offsets of $\rho$.

| | $x=0$ | $x=1$ | $x=2$ | $x=3$ | $x=4$ |
|---|---|---|---|---|---|
| $y=0$ | 0 | 1 | 62 | 28 | 27 |
| $y=1$ | 36 | 44 | 6 | 55 | 20 |
| $y=2$ | 3 | 10 | 43 | 25 | 39 |
| $y=3$ | 41 | 45 | 15 | 21 | 8 |
| $y=4$ | 18 | 2 | 61 | 56 | 14 |

### Стандарт SHA-3

Любой вариант Keccak может быть обозначен как Keccak[r, c, d] с битрейтом r, емкостью c и длиной дайджеста d. В [21] NIST стандартизировал четыре версии SHA-3, которые имеют r = 1600 - 2d и c = 2d, где d ∈ 224, 256, 384, 512. Следовательно, мы можем использовать Keccak-d или SHA-3-d для краткости для обозначения версии SHA-3. Единственное различие между Keccak-d и SHA-3-d заключается в правиле заполнения: Keccak заполняет сообщение на 10 = 1, в то время как SHA-3 передает сообщение на 0110 =1. Это означает , что как для Keccak, так и для SHA-3 последний бит блока сообщений Mw должен быть "1"и только для SHA-3 предпоследнее “1” должно следовать за "01". Следовательно, соответствие правилу заполнения Keccak 4 или SHA-3 еще больше увеличит сложность поиска на 21 или 23.

### Свойства инверсии S-Box

Согласно вычислению Keccak-f, дайджест Keccak окончательно усекается из состояния после последнего шага i, который представляет собой всего лишь простую константу-XOR и может быть непосредственно инвертирован. На шаг назад состояние перед последним шагом $\chi$ также может быть частично восстановлено из дайджеста. Инвертирование последнего шага $\chi$ может эффективно уменьшить алгебраическую степень выходных уравнений и значительно упростить атаки на прообразы. Поскольку шаг $\chi$ обрабатывает разные строки, его можно рассматривать как 5-разрядный S-box, где входные данные a0 a1 a2 a3 a4 и выходные данные b0 b1 b2 b3 b4 вычисляются по (нижнему индексу вычисляется по модулю 5):

$$\begin{cases} b_i = a_i \oplus (a_{i+1} \oplus 1) \cdot a_{i+2} \\ a_i = b_i \oplus (b_{i+1} \oplus 1) \cdot (b_{i+2} \oplus (b_{i+3} \oplus 1) \cdot b_{i+4}) \end{cases}$$

Большинство свойств инверсии S-box были подробно обсуждались в предыдущих работах [15,16,17,18,19]. Для простоты здесь мы просто излагаем некоторые свойства, связанные с этой статьей, без каких-либо доказательств.

I. Сопоставление первой 320-битной плоскости со всеми известными b0b1b2b3b4. В этом случае каждый ai может быть восстановлен в соответствии с уравнениями (2). И любое ограничивающее уравнение для восстановленного искусственного интеллекта может принести выигрыш в 2 1 — прежде чем задавать ограничивающие уравнения, задействованные ai должны быть сначала линеаризованы.

II. Сопоставление 64-битного усечения второй плоскости с известным значением b0. В этом случае существует два способа задать ограничивающие уравнения. Если злоумышленники устанавливают a0 = b0, хотя вероятность совпадения составляет всего 3/4, одно ограничивающее уравнение все равно может принести выигрыш в 3/4÷1/2 ≈ 2 0.58. Если злоумышленники устанавливают a0 = b0 и при этом гарантируют, что a1 = 1 или a2 = 0, b0 должно быть согласовано, и два ограничивающих уравнения могут принести выигрыш в 21.

III. Сопоставление 192-битного усечения второй плоскости с известным b0b1b2. Этот случай гораздо сложнее. С одной стороны, с помощью тонких замен можно доказать, что:

Улучшены атаки по прообразу в раунде - Уменьшен Keccak-384/512

$$\begin{cases} a_0 \oplus (b_1 \oplus 1) \cdot a_2 = b_0 \\ a_1 \oplus (b_2 \oplus 1) \cdot a_3 = b_1 \end{cases}$$

Следовательно, ограничение уравнений на a0 или $a0 \oplus a2$ (в зависимости от b1) и a1 или $a1 \oplus a3$ (в зависимости от b2) всегда может привести к выигрышу в 2 по 1 штуке на каждого. С другой стороны, наши атаки могут столкнуться с особой ситуацией, когда могут быть ограничены только a0 и a4. В этом случае злоумышленники могут установить:

$$\begin{cases} a_0 = b_0 \oplus (b_1 \oplus 1) \cdot b_2 \\ a_4 = 0 \end{cases}$$

Тогда можно доказать, что до тех пор, пока b1 и b2 совпадают случайным образом, b0 должно совпадать одновременно. Следовательно, два ограничивающих уравнения на a0 и a4 всегда могут дать выигрыш в 21.

## 1.1. Условные Обозначения

Начиная с этого раздела, мы больше не будем использовать A для обозначения состояния Keccak-f, поскольку оно не может точно отображать процесс выполнения. Вместо этого мы будем использовать капитал Греческие буквы (в $\{\Theta, P, \Pi, X, I\}$) с надстрочным индексом (от 1 до 3) для обозначения состояния точно после выполнения соответствующего шага. Например, П2 обозначает состояние после второго $\pi$-шага, а X3 обозначает состояние после третьего $\chi$-шага. В частности, I0 обозначает начальное состояние одного Keccak-f (после XORing блока сообщений). Первые r бит I0 называются "входной частью" (XORed с помощью r-бита сообщение), а последние c-биты I0 называются "ограниченной частью" (неконтролируемой в следующем Keccak-f). Чтобы избежать двусмысленности, мы всегда будем использовать три индекса в нижнем индексе для обозначения компонента состояния. Однако мы можем использовать "*" для обозначения всех возможных значений. Для примеров, I1 *,y,z - 5-разрядная строка, I1 x,*,z - 5-разрядный столбец, I1 x,y,* - 64-разрядная полоса, I1 *,y,* - 320-разрядная плоскость, и I1 *,*,z - это срез размером $5 \cdot 5$. Если нижний индекс опущен, это указывает на 1600-битное целое состояние (подобно обозначениям выше). Настройка суммы столбцов является основной проблемой анализа прообразов при уменьшении округления Кеккак. В этой статье мы используем SA с двумя параметрами x, z для обозначения суммы определенного столбца из состояния A, которая равна:

$SA(x, z) = \oplus_{y=0\sim4} A_{x,y,z}$

Аналогично, x, z могут быть заменены на "*" для обозначения набора сумм столбцов.

# 2. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ

В данном разделе описываются некоторые существующие идеи атаки линейного анализа на Keccak-384/512 с уменьшенным кол-вом раундов, которые в значительной степени вдохновили наше исследование. Разработки фреймворка attack можно разделить на две части: линейную структуру с распределяющей моделью и квадратичную структуру с методом линеаризации.

### Линейная структура с моделью распределения

В 2016 году Го и соавт. [15] применили линейный анализ в Keccak с круглым уменьшением и предложили базовую линейную структуру. Их идея состоит в том, чтобы линеаризовать шаг $\chi$, который является единственным нелинейным шагом в функции R, чтобы они могли получить полностью линейное состояние после нескольких раундов. Возьмем в качестве примера их линейную конструкцию с 1 витком для Keccak-384 (как показано на рис. 3).

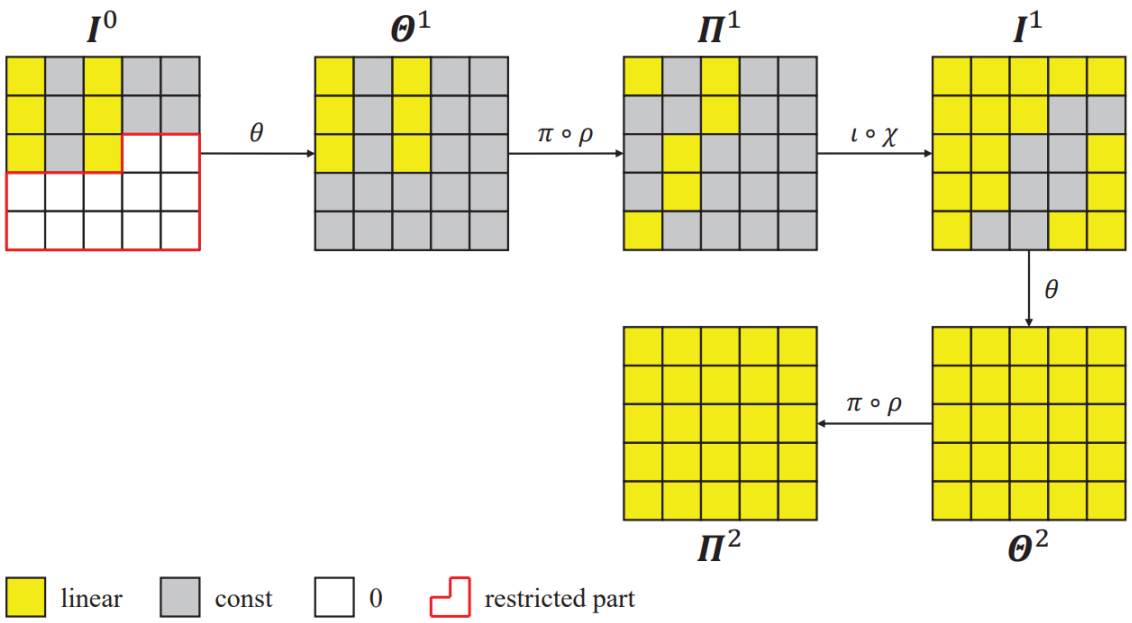


Рис. 3. 1-круговая линейная структура для Keccak-384 в [15].

Поскольку прямое вычисление S-box равно $b_i = a_i \oplus (a_i + 1 \oplus 1) \cdot a_i + 2$, чтобы линеаризовать шаг $\chi$, злоумышленники должны убедиться, что ни в одной строке перед шагом $\chi$ не существует последовательных линейных битов (в состоянии Фn).Однако из-за первого шага $\Theta$ начальные переменные биты будут беспорядочно рассеиваться в Ф1. Чтобы контролировать распространение, злоумышленники должны зафиксировать суммы в соответствующих столбцах, задав (линейные) уравнения. Для рис. 3 уравнения должны быть равны 5:

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{2,0,z} \oplus I^0_{2,1,z} \oplus I^0_{2,2,z} \oplus I^0_{2,3,z} \oplus I^0_{2,4,z} = S_{I^0}(2,z) \end{cases}$$

Затем, установив сумму столбцов, количество линейных полос остается 6 в Ф1, и первый шаг $\chi$ успешно линеаризуется. Тем не менее, по словам форварда вычисление S-box, переменные в Ф1 x,y,* все еще могут распространяться на I1 x-1,y,* или I1 x-2, y,* и, наконец, покройте все Ф2. Следовательно, базовая линейная структура Го и др. может пройти только 1 раунд для Keccak-384. Таким образом, установив 384 начальных бита переменной и зафиксировав суммы по 128 столбцам, Го и др. спроектировал 1-круговую линейную конструкцию для Keccak-384 с 256 градусами свободы не осталось. Эти степени свободы были дополнительно использованы для ограничения эквивалентных битов Ф2 *,0,*, которые могут быть восстановлены из дайджеста (см. раздел 2.4). Учитывая правило заполнения, сложность их поиска при атаке прообразом во 2-м раунде Keccak-384 - это 2384-256+1 = 2129.

Для обеспечения базовой линейной структуры естественной идеей является дальнейшее регулирование диффузии на первом этапе $\chi$, что требует дополнительных условий в ограниченной части I0. В 2019 году Ли и Сан [16] построили модель распределения и решили эту проблему. Применив такую модель, они просто улучшили атаки по прообразам на Keccak-224/256 с уменьшенным количеством раундов. Для лучшего сравнения здесь мы просто создаем двухраундовую линейную конструкцию для Keccak-384 в соответствии с их идеей (как показано на рис. 4).

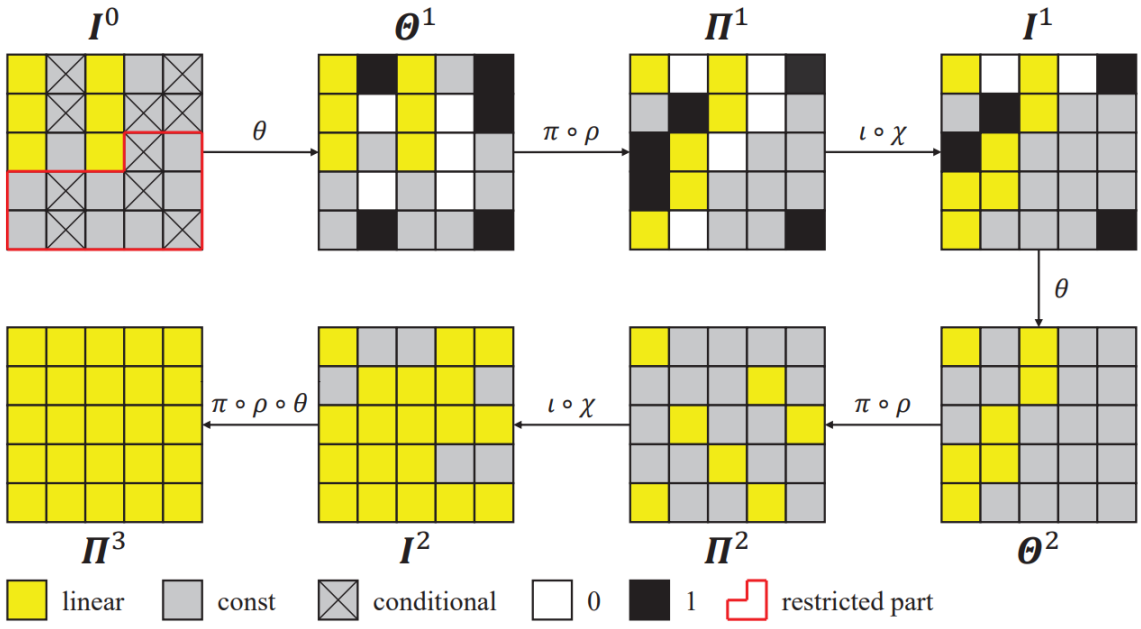


Рис. 4. Двухкруглая линейная структура для Keccak-384 с применением распределяющей модели.

По сравнению с рис. 3, эта структура начинается с идентичных начальных битов переменных (и идентичных уравнений суммы столбцов) в I0. Однако эта структура поддерживает несколько полос "0"и "1"в Ф1, так что диффузией на первой $\chi$ стадии можно эффективно управлять. Аналогично, после задания уравнений суммы столбцов 192 на втором шаге $\Theta$ (как показано ниже), эта структура может линеаризовать второй шаг $\chi$ с помощью 384 - 128 - 192 = осталось 64 степени свободы.

$$\begin{cases} I^1_{0,0,z} \oplus I^1_{0,1,z} \oplus I^1_{0,2,z} \oplus I^1_{0,3,z} \oplus I^1_{0,4,z} = S_{I^1}(0,z) \\ I^1_{1,0,z} \oplus I^1_{1,1,z} \oplus I^1_{1,2,z} \oplus I^1_{1,3,z} \oplus I^1_{1,4,z} = S_{I^1}(1,z) \\ I^1_{2,0,z} \oplus I^1_{2,1,z} \oplus I^1_{2,2,z} \oplus I^1_{2,3,z} \oplus I^1_{2,4,z} = S_{I^1}(2,z) \end{cases}$$

Эти полосы "0"и "1"изначально генерируются на первом шаге $\Theta$. Затем, в соответствии с расчетом шага, заранее требуются дополнительные условия:

$$\begin{cases} I^0_{1,0,z} = I^0_{1,1,z} \oplus 1 = I^0_{1,3,z} \oplus 1 = I^0_{1,4,z} \\ I^0_{3,1,z} = I^0_{3,2,z} = I^0_{3,3,z} \\ I^0_{4,0,z} = I^0_{4,1,z} = I^0_{4,4,z} \end{cases}$$

Среди вышеперечисленных условий $I01,3,z \oplus 1 = I01,4,z$ и $I03,2,z = I0,3,3,z$ относятся к ограниченной части, которую злоумышленники не могут контролировать. Мы называем такого рода условия “ограниченными условиями”. Эти ограниченные условия могут быть выполнены только выводом предыдущего Keccak-f и, следовательно, требуют модели распределения. В общих случаях, даже если злоумышленники удовлетворяют всем ограниченным условиям путем исчерпывающего поиска, сложность все равно далека от сложности поиска по прообразу (временно пренебрегая временем решения) — для рис. 4 первое равно $2^{128}$, в то время как последний является $2^{384} - 64 + 1 = 2321$. Следовательно, общая сложность поиска зависит только от пространства свободы, оставшегося в линейной структуре. Другой заметной проблемой при распределении модели является размер случайного пространства, который представляет собой общее количество различных систем уравнений, которые может использовать линейная структура. генерировать. Пусть d1 обозначает сложность поиска, удовлетворяющую всем ограниченным условиям (d1 может быть постоянного уровня), d2 обозначает сложность

поиска атаки на прообраз, а dr обозначает размер случайного пространства. Обычно dr < d2, в этом случае ожидается, что злоумышленники перезапустят линейную структуру (генерируя еще один квалифицированный I 0) [d2/dr] раз, чтобы найти прообраз. Тогда общая сложность поиска , удовлетворяющая всем ограниченным условиям, равна $d1 \cdot [d2/dr]$ — если dr < d1, общая сложность поиска становится [d1/dr] × d2, а не d2. Поэтому при применении распределяя модель, размер случайного пространства должен удовлетворять $dr \geq d1$. Что касается расчета dr, то это зависит от количества контролируемых сумм столбцов. На рис. 4, SI1 (0, *), SI1 (1, *) и SI1 (2, *) все являются управляемыми, в то время как SI0 должен удовлетворять следующим соотношениям 6, чтобы сгенерировать эти полосы из '0' и '1' в Θ1. Считая управляемый SI0 (1, *), размер случайного пространства равен dr = 264+192 = 2256.

$$\begin{cases} S_{I^0}(0,z) \oplus S_{I^0}(2,z-1) \oplus I^0_{1,3,z} = \Theta^1_{1,3,z} = 0 \\ S_{I^0}(2,z) \oplus S_{I^0}(4,z-1) \oplus I^0_{3,3,z} = \Theta^1_{3,3,z} = 0 \\ S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) \oplus I^0_{4,4,z} = \Theta^1_{4,4,z} = 1 \end{cases}$$

Таким образом, применяя модель распределения, базовую линейную структуру можно повысить до 2-раундовой для Keccak-384. При такой структуре общая сложность поиска атаки с использованием прообраза на 3-раундовом Keccak-384 составляет 2 384-64+1 = 2321. Параметрами модели являются d1 = 2128, d2 = 2321 и dr = 2256, удовлетворяющие $d1 \leq dr$.

### Квадратичная структура с использованием метода линеаризации

В предыдущем разделе все проекты линейной структуры были направлены на получение полностью линейного состояния, в котором в некоторых случаях нет необходимости. Как и в примере на рис. 4, полностью линейный Ф3 содержит 320 ограничивающих уравнений для Ф3 *,0,*, которые могут обеспечить выигрыш в 2 по 1 каждой, в то время как количество оставшихся степеней свободы составляет всего 64. В 2019 году, Раджасри [17] разработал двухкруглую частично линейную конструкцию для Keccak-384 (как показано на рис. 5) что позволило достичь баланса между этими двумя ценностями. В частично линейной структуре допускается существование квадратных полос движения, так что количество линейных ограничивающих уравнений на Ф3 *,0,* и количество оставшихся степеней свободы могут быть согласованы (оба равны 64). Расчеты параметров модели очень похожи на содержание разде-

ла 3.1 — для простоты здесь мы непосредственно заключаем, что параметры модели равны d1 = 264, d2 = 2321 и dr = 2320.

По сравнению с полностью линейной структурой (рис. 4), частично линейная структура Rajasree может просто увеличить размер случайного пространства dr, сохраняя сложность поиска d2 неизменной. Тем не менее, эта идея по-прежнему вдохновляет нас, потому что большее случайное пространство внесет значительный вклад в применение дополнительной линейной зависимости.

# 3. ЗАКЛЮЧЕНИЕ

В этой статье мы предлагаем улучшенный анализ прообразов на Keccak 384/512 с уменьшенным количеством раундов. Ядром наших атак на прообразы является линейный анализ. Мы наследуем существующие фреймворки атак из предыдущих исследований и улучшаем результаты анализа прообразов в двух аспектах:

I. Применяя новую технологию, получившую название extra linear dependence мы строим линейно-зависимые пары битов на уровне полосы пропускания между двумя выходными плоскостями, не расходуя степеней свободы (вместо этого сжимая случайное пространство).

II. Изменяя форму релинеаризации, мы разрабатываем улучшенную квадратичную структурируйте и уменьшите количество переменных в итоговой системе уравнений. В результате сложность прообразных атак на 2-раундовом Keccak-384/512 и 3-раундовый Keccak-384/512 уменьшен до 239/2204 и 2270/2424 вызова Keccak соответственно, и это все наиболее известные результаты на данный момент. Примечательно, что в нашей работе сначала выполняются практические атаки по прообразу на 2-раундовом Keccak-384. Отмечается, что наш алгоритм атаки все еще далек от того, чтобы угрожать безопасности полномасштабного Keccak. Однако идея построения дополнительной линейной зависимости может быть применимо к линейному анализу других криптографических функций.

# 4. СПИСОК ЛИТЕРАТУРЫ