Летняя школа-конференция "Криптография и информационная безопасность" 2021

Сборник тезисов

Новосибирск 2021

Содержание

О школе

Летняя школа-конференция "Криптография и информационная безопасность"памяти С.Ф. Кренелева для студентов и школьников – традиционное мероприятие, проходящее в стенах НГУ каждый год. Организаторами школы-конференции выступают Криптографический центр (Новосибирск), лаборатория криптографии JetBrains Research, факультет информационных технологий, Международный математический Центр в Академгородке, организаторы международной олимпиады NSUCRYPTO и Механико-математический факультет.

Даты проведения: 5 - 19 июля 2021.

Формат участия: очный.

Студенты принимали участие в лекциях, командной и индивидуальной работе в проектах, связанной с решением исследовательских задач в области криптографии и информационной безопасности, спортивных занятиях. Одно из важнейших событий школы-конференции – круглый стол по современным проблемам криптографии. Темы проектов связаны с различными вопросами современной криптографии и информационной безопасности: от разработки современных методов криптоанализа, построения шифров, квантовой криптографии до создания систем аналитической разведки с открытым кодом.

Участие в летней школе-конференции принимали студенты ВУЗов и школьники (11 класс).

Руководитель школы - к.ф.-м.н. Токарева Наталья Николаевна, доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН.

Лекторы и преподаватели школы:

- Nicky Mouha (USA) – PhD, научный сотрудник отдела компьютерной безопасности Национального института стандартов и технологий США (NIST);
- Nikolay Kaleyski (Болгария) – научный сотрудник Центр безопасности коммуникаций им. Селмера Бергенского университета (г. Берген, Норвегия);
- Агиевич Сергей Валерьевич (республика Беларусь) – к.ф.-м.н., заведующий НИЛ проблем безопасности информационных технологий НИИ прикладных проблем математики и информатики Белорусского государственного университета (г. Минск, Беларусь);
- Валиахметов Илья Вадимович – магистрант 2-го курса ФИТ НГУ;
- Высоцкая Виктория Владимировна -– аспирантка ВМК МГУ им. М.В. Ломоносова, специалист-исследователь лаборатории криптографии НПК "Криптонит"(г.Москва);
- Городилова Анастасия Александровна – к.ф.-м.н., старший преподаватель кафедры теоретической кибернетики ММФ НГУ, н.с. ИМ СО РАН;
- Гребнев Сергей Владимирович – ведущий криптограф-исследователь QApp, Российский квантовый центр (г. Москва);
- Идрисова Валерия Александровна – к.ф.-м.н., н.с. Института математики им. С.Л.Соболева СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ;
- Калгин Константин Викторович – к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН;
- Колегов Денис Николаевич – к.т.н., доцент кафедры компьютерной безопасности ТГУ, главный разработчик облачной платформы кибербезопасности компании Bi.Zone (г. Томск);
- Коломеец Николай Александрович – к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, н.с. ИМ СО РАН;
- Кондырев Дмитрий Олегович – аспирант ФИТ НГУ, ассистент кафедры систем информатики ФИТ НГУ, м.н.с. ИМ СО РАН;
- Косточка Светлана Владимировна – м.н.с. Института математики им. С.Л. Соболева СО РАН, тренер ММФ и ФИТ НГУ;
- Куценко Александр Владимирович – аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН;
- Кяжин Сергей Николаевич – к.ф.-м.н., руководитель проектов Лаборатории блокчейн, ПАО «Сбербанк» (г. Москва);
- Николаев Антон Анатольевич – студент кафедры компьютерной безопасности ТГУ, разработчик сервисов анализа защищенности Bi.Zone, главный разработчик фрэймворка Grinder (Томск);
- Максимлюк Юлия Павловна – аспирантка ММФ НГУ, м.н.с. Института математики им. С.С. Соболева СО РАН;

- Сазонова Полина Андреевна – аспирантка ФИТ НГУ, ассистент кафедры общей информатики ФИТ НГУ, м.н.с. ИМ СО РАН;
- Сутормин Иван Александрович – магистрант 1-го курса ММФ НГУ, м.н.с. Института математики им.С.Л.Соболева СО РАН;
- Токарева Наталья Николаевна – к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН.

Организационный комитет:

- Куценко Александр Владимирович;
- Коломеец Николай Александрович;
- Косточка Светлана Владимировна;
- Идрисова Валерия Александровна;
- Зюбина Дарья Александровна;
- Атутова Наталья Дмитриевна;
- Кондырев Дмитрий Олегович.

Лекции:

1. Криптография: быстрый старт – Токарева Н.Н.
2. Особенности структуры современных криптографических хэш-функций – Коломеец Н.А.
3. Non-Stop University CRYPTO Olympiad: ideas, problems, results – Городилова А.А.
4. Введение в технологию блокчейн – Сазонова П.А.
5. Теория кодирования для криптографии: введение – Высоцкая В.В.
6. Шифрование на основе кодов, исправляющих ошибки – Высоцкая В.В.
7. Криптография: история, приложения, современные вопросы – Токарева Н.Н.
8. Схемы подписи на основе кодов, исправляющих ошибки – Высоцкая В.В.
9. Атаки на машинное обучение – Николаев А.А.
10. Уязвимости интернета вещей – Николаев А.А.
11. Основы алгебраического криптоанализа – Куценко А.В.
12. SAT-решаели и криптография – Калгин К.В.
13. Публичное интервью по компьютерной безопасности (часть 1) – Колегов Д.Н.
14. Публичное интервью по компьютерной безопасности (часть 2) – Колегов Д.Н.
15. Квантовый криптоанализ: первое приближение – Куценко А.В.
16. О протоколах доказательства с нулевым разглашением на примере Bulletproof (часть 1) – Кяжин С.Н.
17. О протоколах доказательства с нулевым разглашением на примере Bulletproof (часть 2) – Кяжин С.Н.
18. Архитектура блокчейн-платформ: составные компоненты и принципы работы – Кондырев Д.О.
19. Introduction to ARX constructions – Nicky Mouha
20. Differential cryptanalysis of ARX constructions – Nicky Mouha
21. Crash course on APN functions – Nikolay Kaleyski
22. Testing equivalence between APN functions – Nikolay Kaleyski
23. Подходы к построению криптографических функций – Идрисова В.А.
24. Введение в эллиптические кривые – Гребнев С.В.
25. Протоколы выработки общего ключа – Гребнев С.В.
26. Эффективная реализация криптосистем на эллиптических кривых – Гребнев С.В.

27. Схемы цифровой подписи, в том числе ГОСТ Р 34.10-2012, и методы их анализа – Гребнев С.В.
28. Криптография как реализация полезных интерфейсов – Агиевич С.В.
29. On the distance between APN functions – Nikolay Kaleyski
30. Обзор направлений постквантовой криптографии – Гребнев С.В.
31. Введение в криптографию на изогениях для самых маленьких – Гребнев С.В.

Программа конференции

19 июля 2021г., начало в 14:00:

1. Т.А. Бонич, Д.А. Быков, М.А. Панферов, И.А. Сутормин **"О разностных характеристиках композиции операций циклического сдвига битов и побитового XOR"** (куратор – Н.А. Коломеец)

2. А.С. Мокроусов **"N максимальных разностных характеристик операции XOR по модулю 2^N"** (куратор – Н.А. Коломеец)

3. Н.Д. Атутова, А.О. Бахарев, Д.П. Парфенов **"Разработка алгоритма поиска гарантированного числа активаций (GNA) в криптографических схемах и его использование в задачах построения шифров"** (куратор – А.В. Куценко)

4. Д.А. Зюбина, И.С. Хильчук, А.П. Корнилов, N.S. Malanda **"О корреляционно-иммунных функциях с максимальной алгебраической иммунностью"** (кураторы – Н.Н. Токарева, Ю.П. Максимлюк, К.В. Калгин)

5. А.А. Соколенко, А.Ю. Гилязов, В.В. Гринчуков, В.С. Шапаренко **"Анализ подкодов кода Рида-Маллера для построения постквантовой криптосистемы"** (кураторы – А.В. Куценко, В.В. Высоцкая)

6. Е.А. Волкова, А.А. Григорьевская, М.С. Ерценкин, П.А. Кайдаш, А.Е. Киреева, С.О. Кирсанов, Д.А. Пешков, В.В. Скудина, А.Ю. Чанчиков, А.В. Черкашин. **"Реализация алгоритмов обеспечения приватности в смарт-контрактах Ethereum"** (куратор – Д.О. Кондырев)

7. В.С. Никифоров, А.П. Евсюков, А.В. Головнина, Т.Д. Калмыков, С.Д. Атконов **"Система децентрализованного взаимодействия между компьютерами на основе протокола Kademlia"** (куратор – И.В. Валиахметов)

8. И.А. Матеюк **"Исследование методов source-to-source обфускации. Разработка обфускатора с использованием расширяемой инфраструктуры."** (куратор – И.В. Валиахметов)

ДИЗАЙН ШИФРОВ

Разработка алгоритма поиска гарантированного числа активаций (GNA) в криптографических схемах и его использование в задачах построения шифров

Н.Д. Атутова[1], А.О. Бахарев[1], Д.Р. Парфенов[1]
[1]Новосибирский государственный университет
E-mail: n.atutova@g.nsu.ru, a.bakharev@g.nsu.ru, d.parfenov@g.nsu.ru

Аннотация

Гарантированное число активаций является важной характеристикой, позволяющей получить оценку стойкости шифра к разностному криптоанализу. В данной работе исследован один из алгоритмов (Агиевич, 2020) поиска числа гарантированных активаций XS-схем. Предложен подход к оптимизации существующего решения с помощью метода ветвей и границ, а также анализа специальных матриц, характеризующих XS-схему. Для шифра BeltWBL-2 были проведены вычислительные эксперименты, которые демонстрируют существенное ускорение вычисления гарантированного числа активаций по сравнению с известными подходами.

Ключевые слова: *гарантированное число активаций, XS-схема, разностный криптоанализ, метод ветвей и границ.*

На сегодняшний день разностный (дифференциальный) криптоализ является одним из наиболее эффективных статистических методов анализа симметричных блочных шифров. Данный подход основывается на анализе разностей между парами открытых текстов и соответствующих им пар шифртекстов.

XS-схемы представляют собой конструкции блочных шифров, основанные на использовании двух операций: X (поразрядное сложение двоичных слов по модулю 2) и S (подстановка слов с помощью нелинейных взаимно-однозначных отображений). Известно, что модели XS-схем покрывают достаточно широкий спектр блочных шифров, включая MARS3, SMS4, Skipjack, схему Фейстеля.

В данной работе рассматривается задача оптимизации поиска гарантированного числа активаций в XS-схемах, что позволит получить оценку эффективности разностного криптоанализа таких шифров.

$$(a, B, c)[S] : \mathbb{F}^n \to \mathbb{F}^n, (x_1, x_2, ..., x_n) \longmapsto (x_2, x_3, ..., x_n, x_{n+1}),$$
$$x_{n+1} = (x_1, x_2, ..., x_n)b + S((x_1, x_2, ..., x_n)a).$$

XS-схему можно представить в виде расширенной матрицы

$$\begin{pmatrix} B & a \\ c & 0 \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} & a_1 \\ b_{21} & b_{22} & \dots & b_{2n} & a_2 \\ \vdots & \vdots & \ddots & \vdots & \\ b_{n1} & b_{n2} & \dots & b_{nn} & a_n \\ c_1 & c_2 & \dots & c_n & 0 \end{pmatrix}$$

Для каждого шифра из класса XS-схем (схема находится в первой канонической форме) строится матрица G размерности $(t + n) \times 2t$ (подробнее в [1])

$$G = G(n, a, b, t)$$

столбцы которой имеют следующий вид:

$$\begin{matrix} \tau - 1 \\ n + 1 \\ t - \tau \end{matrix} \begin{cases} \begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix} \\ \begin{matrix} a & b \\ 0 & 1 \end{matrix} \\ \begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix} \end{cases}$$

где b - последний столбец матрицы B.

Для каждого шифра можно найти число активаций — количество ненулевых разностей, попавших на вход S-блоков. Гарантированное число активаций шифра — наименьшее из всех минимальных чисел активаций. Данная характеристика позволяет получить нижнюю оценку сложности разностного криптоанализа шифра. Отметим, что поиск связан с исследованием линейного кода определенного вида, который строится на основе рассматриваемого шифра.

Одним из подходов к поиску гарантированного числа активаций является алгоритм GNA[2]. Основная идея алгоритма - полный перебор разбиений матрицы G на G_0 и G_1, так чтобы:

1. матрица G_0 содержала $k + 1$ (k изначально известно) столбцов из G,
2. $rankG_0 < t + n - 1$, где t-число раундов, n-длина a и b,
3. в матрице G_1 не было ни одного столбца линейно зависимого от столбцов в G_0.

Определение 1. Будем называть разбиение пар столбцов матрицы G на матрицы G_0 и G_1 **"плохим"** если в G_1 содержится столбец, линейно зависящий от столбцов матрицы G_0.

Существующий алгоритм при увеличении t и n работает довольно долго. В работе предложен способ оптимизации разбиения матрицы G, который существенно уменьшает время работы алгоритма.Его основная идея строится на методе ветвей и границ, который является развитием метода полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Предлагается рассмотреть следущую интерпретацию полного перебора. Рассмотрим двоичное дерево, в котором каждый лист соответствует некоторому разбиению пар столбцов матрицы G на матрицы G_0 и G_1. Корень дерева соответствует пустому множеству пар столбцов матрицы G. При переходе с i-го уровня на $(i + 1)$-ый, переход к правому потомку соответстует добавлению $(i+1)$-ой пары столбцов в матрицу G_0, а переход влево – в G_1. Таким образом, каджый узел дерева соответствует некоторому разбиению подмножества пар столбцов матрицы G на матрицы G_0 и G_1.

Предложение 1. *При обходе дерева, в случае, если обрабатываемый узел дерева соответствует "плохому"разбиению, то все потомки данного узла также будут содержать "плохие"разбиения и поэтому их можно не обрабатывать.*

Данный подход позволяет существенно ускорить перебор, однако он может быть улучшен.

Лемма 1. *Рассмотрим n подряд идущие пары столбцов из G, где n – размерность XS-схемы. Тогда первый столбец из $(n+1)$-ой пары будет линейно зависим от столбцов из предыдущих n пар.*

Доказательство. По Лемме 1 из статьи [2] матрица G является полноранговой и имеет ранг равный числу строк $(n+t)$ при $t \geq n$. Строки, в которых есть ненулевые координаты, назовём *значимыми*. Рассмотрим n подряд идущие пары столбцов и оставим в них только значимые строки. Получившаяся матрица совпадает с матрицей G при $t = n$ и имеет ранг равный $2n$. Добавив к такой матрице первый столбец $n+1$-ой пары, мы получим матрицу размерности $(2n) \times (2n+1)$, но ранг матрицы не может превышать число значимых строк, следовательно такая матрица будет иметь ранг равный $2n$. Тогда прибавленный столбец будет линейно выражаться через остальные. □

Следствие 1. *Если в G_0 содержатся n подряд идущие пары столбцов и существует пара с большим порядковым номером, лежащая в G_1, то такое разбиение является "плохим".*

Доказательство. Рассмотрим пару из G_1, порядковый номер которой больше порядковых номеров n подряд идущих пар столбцов, попавших в G_0, и является минимальным. Если така пара столбцов является следующей за n подряд идущими из G_0, то по Лемме 1 первый из её столбцов линейно зависим от столбцов из G_0. По определению такое разбиение является "плохим". Если порядковый номер такой пары не совпадает с следующим за n подряд идущими из G_0, то, в силу его минимальности, в G_0 существуют такие n подряд идущие пары столбцов, что их максимальный порядковый номер будет на единицу меньше порядкового номера пары из G_1. Тогда по рассуждениям выше такое разбиение является "плохим". □

Лемма 2. *Пусть a_1 и b_1 не равны одновременно 1 и выполняются условия Леммы 1. Тогда один из столбцов предшествующей пары линейно зависим от столбцов из следующих n пар.*

Доказательство. Заметим, что a_1 и b_1 не могут одновременно равнятся 0, иначе раунд такой XS-схемы не будет обратимым. Проведя рассуждения, аналогичные доказательству Леммы 1, получим, что столбец из предшествующей пары, у которого первая координата равна 0, линейно выражаться через остальные. □

Следствие 2. *Если в G_0 содержатся n подряд идущие пары столбцов и a_1, b_1 не равны одновременно 1, то такое разбиение является "плохим".*

Доказательство. Из построения матрицы G_0 и Лемм 1, 2 следует, что если в G_0 содержатся n подряд идущие пары столбцов и a_1, b_1 не равны одновременно 1, то в G_1 существует столбец, который линейно зависим от столбцов матрицы G_1. Тогда, по определению, такое разбиение является "плохим". □

Следствия из Лемм 1 и 2 позволяют добавить критерии, отсекающие заведомо неоптимальные ветви перебора. Также, дополнительные критерии могут быть разработаны на их основе.

Предложение 2. *Пусть a_1 и b_1 не равны одновременно 1, тогда если для разбиения, заданного обрабатываемым узлом дерева, невозможно дополнить матрицу G_0 до $k+1$ пары столбцов без появления n подряд идущих в G пар столбцов, то такое разбиение является "плохим".*

Для проверки предложенного решения на практике было произведено несколько тестовых запусков. Референсная реализация [3] GNA выполнена на Python. Вариант обозначенный GNA-alt отличается от референсного GNA только добавлением дерева с отсечением заведомо неоптимальных ветвей согласно предложению 1. Вариант обозначенный GNA-alt-v2 дополнительно содержит

в качестве критериев отсечения следствия из Лемм 1 и 2, а также предложение 2. Реализации GNA-alt и GNA-alt-v2 для корректности сравнения также выполнены на Python и используют те же библиотеки, что и референсная реализация GNA.

На рисунке 1 и в таблице 1 приводится время работы реализаций алгоритма для $15 \leq t \leq 20$ раундов BeltWBL-2.

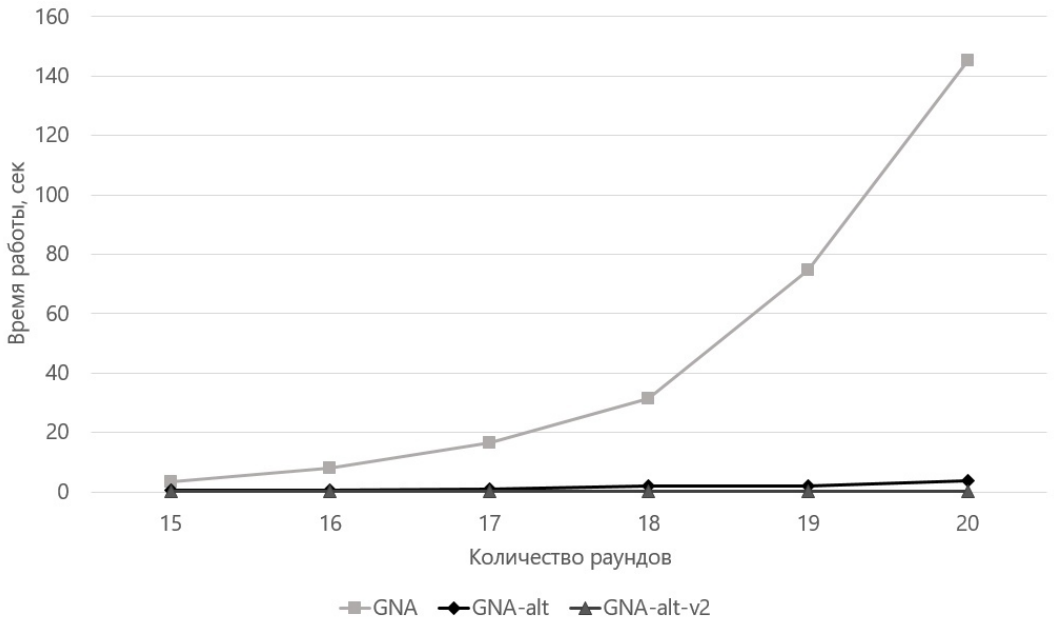

Рис. 1: Время работы различных версий алгоритма для BeltWBL-2

	15	16	17	18	19	20
GNA	3,3678	8,0145	16,627	31,57	74,749	145,39
GNA-alt	0,48	0,4909	1,0074	1,9	1,95	3,9033
GNA-alt-v2	0,0581	0,0226	0,0638	0,1715	0,0698	0,194

Таблица 1: Время работы (в секундах) различных версий алгоритма для BeltWBL-2

В рамках работы был проанализирован алгоритм GNA[2],а также предложен подход к оптимизации существующего решения. Предлагаемые изменения позволяют сущетсвенно ускорить вычисление гарантированного числа активаций, а также вычислять гарантированное число активаций для большего количества раундов, чем референсная реализация.

В дальнейшем планируется продолжить работу по улучшению алгоритма, поиску новых подходов к оптимизации и исследование алгоритма в рамках линейного криптоанализа.

ЛИТЕРАТУРА

[1] S. Agievich – XS-circuits in block ciphers//Mat. Vopr. Kriptogr. 2019, №10 (2).pp.7–30.

[2] S. Agievich – On the Guaranteed Number of Activations in XS-circuits // In: Preproceedings of the 9th Workshop on Current Trends in Cryptology (CTCrypt-2020, September 15–17, 2020), 2020, pp. 73–85.

[3] XS-circuits [Электронный ресурс] URL: https://github.com/agievich/xs (дата обращения 17.07.2021).

Куратор исследования – аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН Куценко Александр Владимирович

О корреляционно-иммунных функциях с максимальной алгебраической иммунностью

Д. А. Зюбина[1], И. С. Хильчук[1], А. П. Корнилов[2], N. S. Malanda[1]
[1]Новосибирский государственный университет
[2]Южный федеральный университет
E-mail: d.zyubina@g.nsu.ru, i.khilchuk@g.nsu.ru, akornilov@sfedu.ru, n.malanda@g.nsu.ru

Аннотация

В работе рассматривается геометрическое представление корреляционно-иммунных булевых функций с максимальной алгебраической иммунностью. Найдено пересечение классов функций с максимальной алгебраической иммунностью и функций, обладающих корреляционной иммунностью, от малого числа переменных. Для $n = 3, 4, 5$ произведена классификация таких функций.

Ключевые слова: *булевы функции, алгебраическая иммунность, корреляционная иммуность, булев куб.*

Обозначим через $\mathbb{Z}_2$ множество $\{0, 1\}$, тогда $\mathbb{Z}_2^n$ — векторное пространство двоичных векторов длины n. Пусть $\oplus$ обозначает сложение по модулю 2. Определим *скалярное произведение* $\langle x, y\rangle$ двух векторов из $\mathbb{Z}_2^n$ как число $x_1y_1 \oplus \ldots \oplus x_ny_n$. *Булева функция* f — это произвольное отображение из $\mathbb{Z}_2^n$ в $\mathbb{Z}_2$. Любую булеву функцию можно единственным образом записать в *алгебраической нормальной форме (АНФ, полином Жегалкина)* :

$$f(x_1, \ldots, x_n) = \left(\bigoplus_{k=1}^{n} \bigoplus_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k}\, x_{i_1} \cdot \ldots \cdot x_{i_k} \right) \oplus a_0,$$

где при каждом k все индексы $i_1, \ldots, i_k$ различны и параметры $a_0, a_{i_1}, \ldots, a_{i_k}$ принимают значения 0 или 1. *Степенью* булевой функции называется число переменных в самом длинном ненулевом слагаемом АНФ. Для стойкости шифра необходимо, чтобы функции обладали высокой алгебраической степенью. Функции, степень которых не превосходит 1, называются *аффинными*.

Носитель булевой функции — множество всех векторов, на которых функция принимает значение 1:

$$supp(f) = \{x \in \mathbb{Z}_2^n : f(x) = 1\}.$$

Весом Хэмминга $wt(f)$ булевой функции f от n переменных называется число ненулевых координат ее вектора значений. *Расстояние Хэмминга* $dist(f, g)$ между двумя булевыми функциями f и g от n переменных — число векторов $x \in \mathbb{Z}_2^n$, на которых функции принимают различные значения, или, что эквивалентно, $dist(f, g) = wt(f \oplus g)$.

Преобразование Уолша-Адамара булевой функции f от n переменных — целочисленная функция

$$W_f(y) = \sum_{x \in \mathbb{Z}_2^n} (-1)^{\langle x,y\rangle \oplus f(x)}, \text{ где } y \in \mathbb{Z}_2^n.$$

Булев куб — граф $\mathbb{E}^n$, вершинами которого являются все двоичные векторы длины n, т. е. $V = \{(x_1, \ldots, x_n) : x_i \in \mathbb{Z}_2\}$, а ребрами соединяются только те векторы, расстояние Хэмминга между которыми равно единице. Число n называется размерностью булева куба.

Гранью размерности k в булевом кубе $\mathbb{E}^n$ называется множество

$$\Gamma^{a_1,\ldots,a_{n-k}}_{i_1,\ldots,i_{n-k}} = \{x_{i_1} = a_1, \ldots, x_{i_{n-k}} = a_{n-k}\}.$$

Множество $\{i_1, \ldots, i_{n-k}\}$ называется направлением грани.

Булева функция f от n переменных называется *корреляционно-иммунной порядка* r, $1 \leq r \leq n$, если для любой её подфункции $f^{a_1,\ldots,a_r}_{i_1,\ldots,i_r}$, полученной фиксацией r переменных, выполняется равенство

$$wt(f^{a_1,\ldots,a_r}_{i_1,\ldots,i_r}) = \frac{wt(f)}{2^r}.$$

Имеет место эквивалентное определение. Булева функция f от n переменных является корреляционно-иммунной порядка $n-k$, $1 \leq k \leq n$, если любой грани $\Gamma^{a_1,\ldots,a_{n-k}}_{i_1,\ldots,i_{n-k}}$ булева куба $\mathbb{E}^n$ размерности k принадлежит одинаковое число точек носителя функции f, а именно $wt(\Gamma^{a_1,\ldots,a_{n-k}}_{i_1,\ldots,i_{n-k}}) = wt(f) \cdot 2^{-(n-k)}$.

Функция g называется *аннулятором* функции f, если g не равен тождественно нулю и $f(x)g(x) \equiv 0$.

Алгебраическая иммунность $AI(f)$ функции f — минимальная из степеней аннуляторов функций f и $f \oplus 1$.

Теорема 1. *Функция f от n переменных является корреляционно-иммунной порядка $n-2$, если она является функцией одного из следующих типов:*

- *константа,*
- *счётчик чётности или его отрицание:* $x_1 \oplus x_2 \oplus \ldots \oplus x_n, x_1 \oplus x_2 \oplus \ldots \oplus x_n \oplus 1$,
- *существует $i \in \{1, \ldots, n\}$ такой, что f не зависит от переменной x_i, а от всех остальных переменных зависит существенно.*

Рассмотрим функции от четырёх переменных веса $wt(f) = 6$ с максимальной алгебраической иммунностью и корреляционной иммунностью $CI(f) = 1$. Таких функций 36. Заметим, что в алгебраической нормальной форме каждой из 36 функций присутствуют все возможные мономы степени 3. Покажем возможную классификацию данных функций:

Первому типу (рис.2) принадлежат 12 функций, в АНФ которых по три монома второй степени и всевозможные мономы степени один:

$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_2x_4 \oplus x_3x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_2x_4 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_2x_3 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_1x_4 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_1x_3 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_1x_4 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_1x_3 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_1x_2 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$

От каждой из этих функций можно перейти к другой функции того же типа с помощью транспозиции переменных или последовательного применения нескольких транспозиций.

Второму типу (рис.3) принадлежат 12 функций:

$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_2x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_2x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_1x_4 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$

$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_1x_4 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_1x_2 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_1x_2 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_1x_3 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_1x_2 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_1x_2 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$

В АНФ данных функций по два монома второй степени и три монома первой степени. Переход также осуществляется перестановкой переменных.

Третьему типу (рис.4) принадлежат 12 функций, в АНФ которых по одному моному второй степени и три монома первой степени:

$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_3x_4 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_4 \oplus x_3 \oplus x_2 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_4 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_2 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_2 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$
$x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_3 \oplus x_2 \oplus x_1 \oplus 1$

Переход осуществляется перестановкой переменных, являющейся транспозицией или комбинацией пары транспозиций.

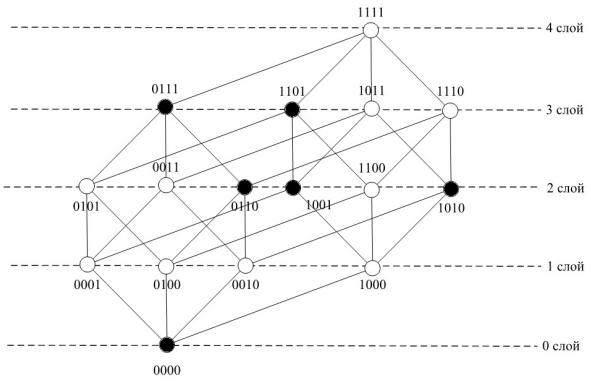

Рис. 2: Тип №1 для $wt(f) = 6$

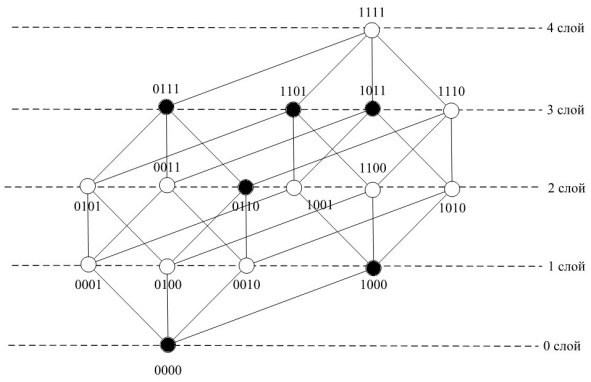

Рис. 3: Тип №2 для $wt(f) = 6$

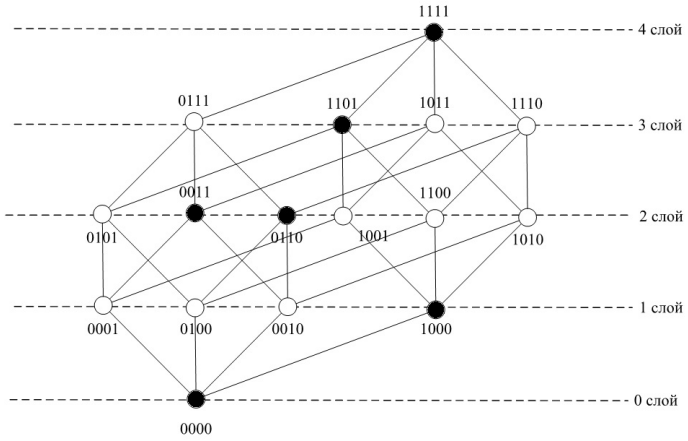

Рис. 4: Тип №3 для $wt(f) = 6$

Для $n = 3$ была получена полная классификация булевых функций с корреляционной иммунностью порядка 3, 2, 1. Всего существует 2 функции порядка 3 — функции-константы; 4 функции порядка 2 — функции-константы и функции-счётчики чётности; 18 функций порядка 1. Из всех 18 булевых функций от трёх переменных ни одна не имеет максимально возможное значение алгебраической иммунности (т.е. $AI(f) < 2$).

Для $n = 4$ также была полученна полная классификация булевых функций с корреляционной иммунностью порядка 4, 3, 2, 1. Всего существует 2 функции порядка 4 и 4 функции порядка 3 (аналогично ситуации для функции от 3 переменных). Есть 12 функций с корреляционной иммуностью порядка 2 и 648 функций порядка 1. При пересечении множества функций с корреляционной иммунностью $CI(f) \geq 1$ и максимальной алгебраической иммунностью было обнаружено, что существует 392 функции с максимальной алгебраической иммунностью ($AI(f) = 2$) и корреляционной иммунностью порядка 1 ($CI(f) = 1$). Функций с корреляционной иммунностью порядка 2 среди функций с максимальной алгебраической иммунностью не существует.

При $n = 5$ был взят полный список булевых функций с максимальной алгебраической иммунностью ($AI(f) = 3$) - всего их 197 765 122. Из них 48 384 функций имеют корреляционную иммунность порядка 1 ($CI(f) = 3$). Функций с более высоким порядком корреляционной иммунности среди функций с максимальной алгебраической иммунностью не существует.

Рассмотрим все функции от четырех переменных с весом Хэмминга $wt(f) = 8$, максимальной для данного числа переменных алгебраической имунностью $AI(f) = 2$ и корреляционной имунностью $CI(f) = 1$. Таких функций в данной размерности существует 200. Рассмотрим функции вида f без учета функций вида $f \oplus 1$. Произведем типизацию данных функций с учетом того, что при вращении куба в трехмерном пространстве и при смене координатного базиса составного трехмерного куба данный объект (представление носителя f в гиперкубе) не меняет свой вид. Корреляционная иммунность функции от 4 переменных с весом 8, равная единице, означает, что каждой грани размерности 3 булева куба принадлежат ровно четыре точки из носителя функции. Существует четыре различных варианта топологий куба, что показано на рисунках 5–8. Закрашеные вершины принадлежат носителю функции, те что не являются закрашеными — не принадлежат.

Среди 100 рассматриваемых функций f с весом 8 определим, к какому из приведенных типов относится каждая.В данной классификации наблюдается следующее соотношение между типом и количеством функций с заданным весом:

- Тип №1 – 16 функций;

- Тип №2 – 24 функций;
- Тип №3 – 48 функций;
- Тип №4 – 12 функций.

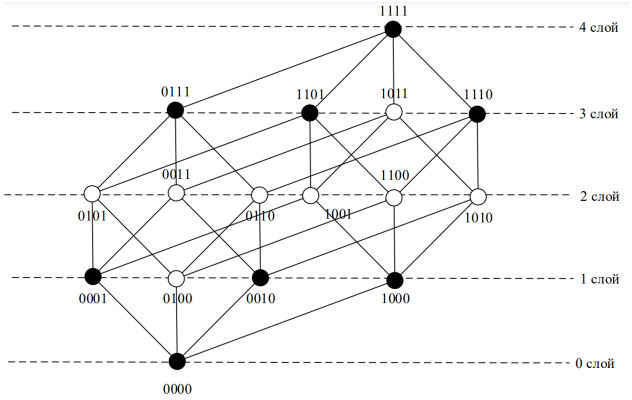

Рис. 5: Тип №1 для $wt(f) = 8$

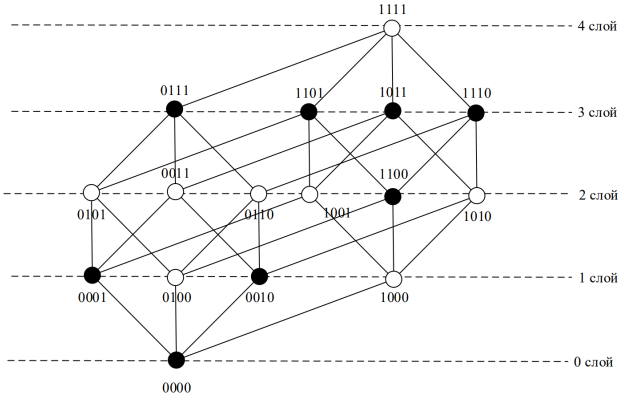

Рис. 6: Тип №2 для $wt(f) = 8$

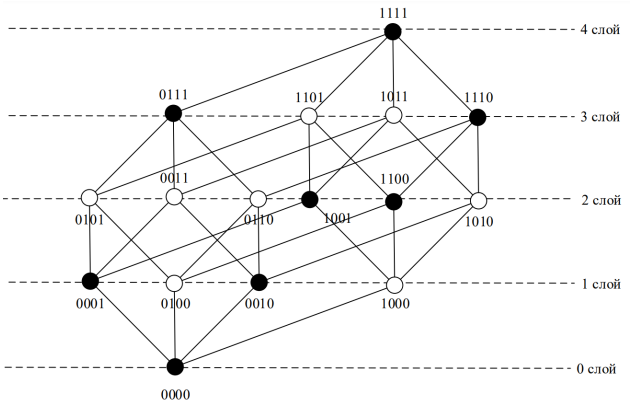

Рис. 7: Тип №3 для $wt(f) = 8$

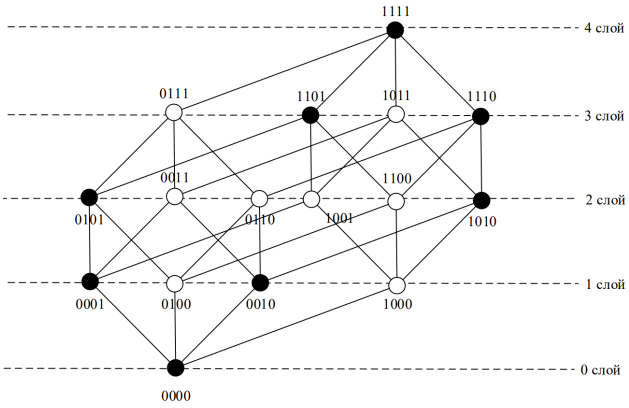

Рис. 8: Тип №4 для $wt(f) = 8$

Легко заметить, что первый, второй и третий типы получаются фиксацией одного трехмерной грани в точках (векторах), которые соответствуют 0000, 0001, 0010, 0111 векторам пространства $\mathbb{Z}_2^4$, а другая трехмерная грань получается при помощи добавления ($\oplus$) в функцию монома, содержащего новую для данной грани координату и с применением транспозиции переменных или последовательного применения нескольких транспозиций. Ситуации с левым и правым вращением изоморфны друг другу. Тип №4 представляет из себя замкнутый цикл на половине вершин четырехмерного булева куба, у каждой вершины, принадлежащей носителю функции есть два ребра, соединяющие ее с другими точками из носителя.

ЛИТЕРАТУРА

[1] Carlet, C. Boolean Functions for Cryptography and Error-Correcting Codes. – Text : unmediated // Boolean Models and Methods in Mathematics, Computer Science and Engineering / Y. Crama and P. L. Hammer. – Cambridge : Cambridge University Press, 2010. – P. 257–397. – ISBN 978-0-521-84752-0.

[2] Токарева, Н. Н. Симметричная криптография. Краткий курс : учебное пособие. -– Новосибирск : Изд-во НГУ , 2012. – 234 с. – ISBN 978-5-4437-0067-0.

[3] Панкратова, И. А. Булевы функции в криптографии : учебное пособие. – Томск : Издательский дом ТГУ, 2014. – 88 с.

Кураторы исследования –

к.ф.-м.н., с.н.с. Института математики им. С. Л. Соболева СО РАН, Наталья Николаевна Токарева; аспирантка ММФ НГУ, м.н.с. Института математики им.С.Л.Соболева СО РАН, Юлия Павловна Максимлюк;

к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, Константин Викторович Калгин.

КРИПТОАНАЛИЗ СИММЕТРИЧНЫХ ШИФРОВ

О разностных характеристиках композиции операций циклического сдвига битов и побитового XOR

Т.А. Бонич, Д.А. Быков, М.А. Панферов, И.А. Сутормин
Новосибирский государственный университет
E-mail: t.bonich@g.nsu.ru, d.bykov1@g.nsu.ru ,m.panferov@g.nsu.ru, i.sutormin@g.nsu.ru

Аннотация

В работе исследуется разностная характеристика композиции операций циклического сдвига битов и побитового XOR. Эта величина широко используется при криптоанализе симметричных шифров архитектуры ARX. В данной работе изучены свойства рассматриваемой характеристики, такие как симметрии относительно аргументов и случаи, когда она не равна нулю. Получены реккурентные формулы для её вычисления.

Ключевые слова: *ARX, дифференциальный криптоанализ, XOR, сложение по модулю*

ARX – одна из современных архитектур для примитивов симметричной криптографии. В компонентах шифров ARX архитектуры используются три операции: сложение по модулю 2^n ($+$, A), циклический сдвиг на r влево ($\lll r$, R) и покомпонентное сложение по модулю 2 ($\oplus$, X). В качестве примеров можно привести шифры FEAL, Threefish, Salsa20, а также хэш-функции BLAKE и Skein. Их преимуществами являются скорость и простота программной реализации, а также устойчивость к атакам по времени. Одним из недостатков является сложность их разностного криптоанализа.

Разностный криптоанализ основан на изучении преобразования разностей открытых текстов в разности шифртекстов. Обычно при изучении прохождения разностей через несколько операций, в частности, последовательное применение операций $+$, $\lll r$ и $\oplus$, предполагают независимость выходных разностей предыдущей операции и входных разностей следующей и перемножают вероятности для каждой операции. На практике это предположение зачастую неверно, и поэтому имеет смысл рассматривать $+$, $\lll r$ и $\oplus$ как единое целое.

Если использовать разность по модулю 2^n, то разности проходят через + с вероятностью 1. Значит, имеет смысл рассматривать лишь последовательные $\lll r$ и $\oplus$. Введем разностную характеристику RX относительно сложения по модулю 2^n

$$\mathrm{adp}^{\mathrm{RX}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{P}[x, y \in \mathbb{Z}_2^n \,:\, ((x+\alpha) \lll r) \oplus (y+\beta) = ((x \lll r) \oplus y) + \gamma] =$$
$$= \frac{|x, y \in \mathbb{Z}_2^n \,:\, ((x+\alpha) \lll r) \oplus (y+\beta) = ((x \lll r) \oplus y) + \gamma|}{2^{2n}}.$$

Аналогично определяются вероятности для XR:

$$\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \frac{|x, y \in \mathbb{Z}_2^n \,:\, ((x+\alpha) \oplus (y+\beta)) \lll r = (x \oplus y) \lll r + \gamma|}{2^{2n}}.$$

Заметим, что с вектором $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{Z}_2^n$ мы ассоциируем целое число $x_0 + x_1 2^1 + \ldots + x_{n-1} 2^{n-1}$. Операции $+$ и $-$ в контексте элементов $\mathbb{Z}_2^n$ означают операции над этими числами по модулю 2^n. В работах [1, 2] можно найти информацию об $\mathrm{adp}^{\mathrm{X}} = \mathrm{adp}^{\oplus}$.

Известно, что $\mathrm{adp}^{\mathrm{RX}}(\alpha, \beta \xrightarrow{r} \gamma)$ при $\alpha, \beta, \gamma \in \mathbb{Z}_2^n$ представимо в виде суммы четырех произведений матриц 8×8, что позволяет вычислять $\mathrm{adp}^{\mathrm{RX}}$ за линейное по n время, подробнее см. [3].

Нетрудно показать, что $\mathrm{adp}^{\mathrm{RX}}(a, b \xrightarrow{r} c) = \mathrm{adp}^{\mathrm{XR}}(c, b \xrightarrow{n-r} a)$. Поэтому можно рассматривать только одну из величин.

Также были получены симметрии для $\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma)$.

Теорема 2. *Для любых α, β, γ следующие равенства верны*

1. $\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{adp}^{\mathrm{XR}}(\beta, \alpha \xrightarrow{r} \gamma)$
2. $\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{adp}^{\mathrm{XR}}(-\alpha, -\beta \xrightarrow{r} -\gamma)$
3. $\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{adp}^{\mathrm{XR}}(-\alpha, -\beta \xrightarrow{r} \gamma)$
4. $\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{adp}^{\mathrm{XR}}(\alpha + 2^{n-1}, \beta + 2^{n-1} \xrightarrow{r} \gamma)$

Введем вспомогательные понятия padp и cadp. Пусть $\alpha, \beta, \gamma \in \mathbb{Z}_2^n$, и $\alpha = (\alpha_R, \alpha_L)$, $\beta = (\beta_R, \beta_L)$, $\gamma = (\gamma_R, \gamma_L)$, где $\alpha_R, \beta_R, \gamma_L \in \mathbb{Z}_2^{n-r}$, $\alpha_L, \beta_L, \gamma_R \in \mathbb{Z}_2^r$. Отметим, что α_R содержит младшие биты, т.е. $\alpha_R = (\alpha_0, \ldots, \alpha_{n-r-1})$.

Тогда padp и cadp определим следующим образом

$$\mathrm{cadp}^c(\alpha, \beta \to \gamma) = \frac{1}{4^n} | x, y \in \mathbb{Z}_2^n : (x + \alpha) \oplus (y + \beta) = \gamma + (x \oplus y) \\ \text{и } c \cdot 2^n \le \gamma + (x \oplus y) < (c+1) \cdot 2^n |, \text{ где } c \in \mathbb{Z}_2,$$

$$\mathrm{padp}^{a,b}(\alpha, \beta \to \gamma) = \frac{1}{4^n} | x, y \in \mathbb{Z}_2^n : (x + \alpha) \oplus (y + \beta) = \gamma + (x \oplus y) \\ \text{и } a \cdot 2^n \le \alpha + x < (a+1) \cdot 2^n, \\ b \cdot 2^n \le \beta + y < (b+1) \cdot 2^n |, \text{ где } a, b \in \mathbb{Z}_2$$

Теорема 3. *Для любых α, β, γ верно следующее равенство*

$$\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \sum_{a,b,c \in \mathbb{Z}_2} \mathrm{padp}^{a,b}(\alpha_R, \beta_R \to \gamma_L + c) \mathrm{cadp}^c(\alpha_L + a, \beta_L + b \to \gamma_R).$$

Следствие 3. *Пусть $c = \alpha_{R0} \oplus \beta_{R0} \oplus \gamma_{L0}$, $a = \alpha_{L0} \oplus \beta_{L0} \oplus \gamma_{R0}$. Тогда*

$$\mathrm{adp}^{\mathrm{XR}}(\alpha, \beta \xrightarrow{r} \gamma) = \mathrm{padp}^{a,0}(\alpha_R, \beta_R \to \gamma_L + c) \mathrm{cadp}^c(\alpha_L + a, \beta_L \to \gamma_R) \\ + \mathrm{padp}^{a \oplus 1, 1}(\alpha_R, \beta_R \to \gamma_L + c) \mathrm{cadp}^c(\alpha_L + (a \oplus 1), \beta_L \oplus 1 \to \gamma_R).$$

Пусть $P = (p_0, p_1, p_2), Q = (q_0, q_1, q_2) \in \mathbb{Z}_2^3$ и $\alpha p_0 = (\alpha_0, \ldots, \alpha_n, p_0)$, $1^n = (1, \ldots, 1)$ – единичный вектор длины n и $q_0 1^n = (q_0, \ldots, q_0)$. За $Q \preceq P$ обозначим, что вектора Q, P удовлетворяют условию $q_i \le p_i$ (i=0,1,2). Тогда можно вывести рекуррентные формулы для padp и cadp.

Теорема 4. *(Рекуррентная формула для* padp*)* $\mathrm{padp}^{a,b}(\alpha p_0, \beta p_1 \to \gamma p_2) =$

$$\begin{cases} 0, & \textit{если } \mathrm{wt}(P) \textit{ нечетно,} \\ \mathrm{padp}^{a,b}(\alpha, \beta \to \gamma), & \textit{если } \mathrm{wt}(P) = 0, \\ \frac{1}{4} \sum\limits_{Q \preceq P} \mathrm{padp}^{a \oplus q_0, b \oplus q_1}(\alpha \oplus q_0 1^n, \beta \oplus q_1 1^n \to \gamma \oplus q_2 1^n), & \textit{иначе.} \end{cases}$$

Теорема 5. *(Рекуррентная формула для* cadp*)* $\mathrm{cadp}^c(\alpha p_0, \beta p_1 \to \gamma p_2) =$

$$\begin{cases} 0, & \textit{если } \mathrm{wt}(P) \textit{ нечетно,} \\ \mathrm{cadp}^c(\alpha, \beta \to \gamma), & \textit{если } \mathrm{wt}(P) = 0, \\ \frac{1}{4} \sum\limits_{Q \preceq P} \mathrm{cadp}^{c \oplus q_2}(\alpha \oplus q_0 1^n, \beta \oplus q_1 1^n \to \gamma \oplus q_2 1^n), & \textit{иначе.} \end{cases}$$

Для cadp был найден простой критерий на выбор аргументов $\alpha, \beta, \gamma \in \mathbb{Z}_2^n$, таких что $\mathrm{cadp}^c(\alpha, \beta \to \gamma) > 0$.

Предложение 3. *Пусть* $\alpha, \beta, \gamma \in \mathbb{Z}_2^n$. *Тогда*

- $\mathrm{cadp}^0(\alpha, \beta \to \gamma) = 0$ *тогда и только тогда, когда* $adp^{\oplus}(\alpha, \beta \to \gamma) > 0$,
- *если* $\gamma \neq 0$, *то* $\mathrm{cadp}^1(\alpha, \beta \to \gamma) = 0$ *если и только если* $adp^{\oplus}(\alpha, \beta \to \gamma) > 0$,
- $\mathrm{cadp}^1(\alpha, \beta \to 0) = 0$.

Для padp в случае $\alpha, \beta, \gamma \neq 0$ и $\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) > 0$ простые условия на аргументы, когда

$$\mathrm{padp}^{a,b}(\alpha, \beta \to \gamma) = 0$$

найти пока не удалось. Остальные случаи равенства характеристики нулю были разобраны, и при $\alpha, \beta, \gamma \neq 0$ доказанны полезные соотношения для padp в стиле Предложения 3.

ЛИТЕРАТУРА

[1] Lipmaa, H., Wallén, J., Dumas, P. – On the Additive Differential Probability of Exclusive-Or. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 317–331. Springer, Heidelberg (2004)

[2] Mouha, N., Velichkov, V., De Cannière, C., Preneel, B. – The Differential Analysis of S-Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 36–56. Springer, Heidelberg (2011)

[3] Velichkov, V., Mouha, N., De Cannière, C., Preneel, B. – The Additive Differential Probability of ARX. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 342–358. Springer, Heidelberg (2011)

Куратор исследования – к.ф.-м.н., н.с. Института математики им. С.Л.Соболева СО РАН, Н.А. Коломеец.

N максимальных разностных характеристик операции XOR по модулю 2^N

А.С. Мокроусов[1]
[1]Новосибирский государственный университет
E-mail: a.mokrousov@g.nsu.ru

Аннотация

В работе рассмотрены свойства величины $\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma), \alpha, \beta, \gamma \in \mathbb{Z}_2^n$, которая является вероятностью выходной разности γ у операции XOR при входных разностях α и β, разности рассматриваются по модулю 2^n. Она используется при проведении разностного криптоанализа шифров архитектуры ARX. Были получены явные выражения для n максимальных значений вероятности и троек (α, β, γ), на которых данные значения достигаются.

Ключевые слова: *ARX, разностный криптоанализ*

Одной из современных конструкций, используемых в симметричной криптографии, является архитектура ARX. В ней шифры строятся с использованием трех операций: сложение $-$ $(x + y) \bmod 2^n$, циклический сдвиг $-$ $(x_{n-1}, x_{n-2}, ..x_0) \to (x_0, x_{n-1}, ...x_1)$ и XOR $-$ $x \oplus y$.

Разностный криптоанализ $-$ один из наиболее используемых методов криптоанализа. Он основан на рассмотрении того, как разности открытых текстов переходят в разности шифротекстов. Чем выше вероятность какого-то перехода (разностная характеристика), тем проще осуществлять данный вид криптоанализа. Поэтому максимальные разностные характеристики представляют наибольший интерес. Такой вид криптоанализа применим и к ARX, однако для практически используемых $n = 32, 64$ перебрать все возможные разности затруднительно. В связи с этим необходимы аналитические способы находить максимальные разностные характеристики.

В данной работе были исследованы разностные характеристики преобразования $f(x, y) = x \oplus y$, где в качестве разности берется вычитание по модулю 2^n. По определению они вычисляются следующим образом:

$$\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) = \frac{\underset{x,y \in \mathbb{Z}_2^n}{\mathrm{Cnt}} [((x + \alpha) \oplus (y + \beta)) - (x \oplus y) = \gamma]}{2^n \cdot 2^n}$$

В рамках работы получены явные выражения для n максимальных значений вероятности и троек (α, β, γ), на которых данные значения достигаются.

Рассмотрим следующие группы преобразований:

- $G_1 = \{(\alpha, \beta, \gamma) \to (\alpha, \beta, \gamma), (\alpha, \beta, \gamma) \to (\alpha + 2^{n-1}, \beta + 2^{n-1}, \gamma), (\alpha, \beta, \gamma) \to (\alpha, \beta + 2^{n-1}, \gamma + 2^{n-1}), (\alpha, \beta, \gamma) \to (\alpha + 2^{n-1}, \beta, \gamma + 2^{n-1})\}$ (4 элемента)
- $G_2 = \{(\alpha, \beta, \gamma) \to (\pm\alpha, \pm\beta, \pm\gamma)\}$ (8 элементов)
- $G_3 = S_3$ — группа перестановок трёхэлементного множества (6 элементов)

Из [1] известно, что все они сохраняют значение $\mathrm{adp}^{\oplus}$.

Определим, что $(\alpha, \beta, \gamma) \sim (\alpha', \beta', \gamma')$, если существует цепочка преобразований $T_1, ..., T_n, T_i \in (G_1 \cup G_2 \cup G_3), (\alpha, \beta, \gamma) \circ T_1 \circ ... \circ T_n = (\alpha', \beta', \gamma')$

С использованием введенного отношения эквивалентности возможно определить все тройки (α, β, γ), имеющие n максимальных разностных характеристик.

Теорема 6. *Пусть* $P = \{\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) \,|\, \alpha, \beta, \gamma \in \mathbb{Z}_2^n\}$. *Обозначим за* $(p_1, p_2, ...p_n, p_{n+1})$ *первые* $n + 1$ *элементов из* P *в порядке убывания. Тогда*

1. $p_1 = 1,\ \forall i\ \ 2 \le i \le n,\ p_i = p_{i-1} - \frac{1}{2 \cdot 4^{i-2}}$

2. $\forall i \le n\ \ \mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) = p_i \Leftrightarrow (\alpha, \beta, \gamma) \sim (0, 2^{n-i}, 2^{n-i}))$

3. $p_{n+1} \le \frac{1}{4}$

Теорема 7. *Пусть* $C_i = \{(\alpha, \beta, \gamma) \,|\, (\alpha, \beta, \gamma) \sim (0, 2^{n-i}, 2^{n-i}), \alpha, \beta, \gamma \in \mathbb{Z}_2^n\}$. *Тогда*

$$|C_i| = \begin{cases} 4, & i = 1 \\ 24, & i = 2 \\ 48, & 3 \le i \le n \end{cases}$$

ЛИТЕРАТУРА

[1] Mouha N. et al. Maximums of the Additive Differential Probability of Exclusive-Or // IACR Transactions on Symmetric Cryptology. – 2021. – С. 292-313.

Куратор исследования – к.ф.-м.н., н.с. Института математики им. С.Л. Соболева СО РАН, Н.А. Коломеец.

ПОСТКВАНТОВАЯ КРИПТОГРАФИЯ

Анализ подкодов кода Рида-Маллера для построения постквантовой криптосистемы

А.А. Соколенко[1], А.Ю. Гилязов[2], В.В. Гринчуков[1], В.С. Шапаренко[2]
[1]Балтийский федеральный университет имени Иммануила Канта
[2]Новосибирский государственный университет
E-mail: В.С. Шапаренко: v.shaparenko@g.nsu.ru,
В.В. Гринчуков: brazetb@gmail.com

Аннотация

Разработка и анализ постквантовых криптосистем является актуальным направлением современной криптографии. Одним из основных направлений является развитие криптосистем, основанных на использовании кодов, исправляющих ошибки. Классические двоичные коды Рида–Маллера не являются подходящими для использования в таких криптосистемах. Тем не менее, эти коды допускают достаточно эффективную реализацию, в связи с чем возникает задача поиска подходящих подкодов кодов Рида-Маллера с целью построения стойких криптосистем. Также рассматривается задача изучения «слабых» подкодов кода Рида-Маллера порядка 2, использование которых допускает сведение задачи криптоанализа к анализу криптосистемы на полном коде. В настоящей работе были проанализированы порождающие матрицы таких подкодов, а также описаны некоторые графы, соответствующие слабым подкодам.
Ключевые слова: *постквантовая криптография, подкоды кодов Рида-Маллера, произведение Адамара*

Человечество стоит на пороге изобретения полноценного квантового компьютера, обладающего достаточной мощностью для реализации известных квантовых алгоритмов факторизации и дискретного логарифмирования [1], что представляет угрозу современным ассиметричным криптосистемам. В связи с этим ведётся активная разработка новых стандартов постквантовой криптографии. Национальным институтом стандартов и технологий США в настоящее время проводится конкурс на стандарт постквантовой криптографии. Одним из финалистов 3-го раунда данного конкурса стала криптосистема Мак-Элиса, основанная на кодах, исправляющих ошибки.

Известным семейством кодов, которое можно применять в этой и многих других кодовых криптосистемах, являются коды Рида-Маллера. Однако существует ряд эффективных атак на криптосистемы на основе этих кодов (например, атаки Миндера-Шокроллахи [2] и Чижова-Бородина [3]). Одной из актуальных задач является поиск подкодов данного кода для построения стойкой криптосистемы.

Пусть $\mathbb{F}_2^n$ обозначает n-мерное метрическое пространство всех двоичных векторов длины n с метрикой Хэмминга.

Определение 2. Произвольное подмножество C пространства $\mathbb{F}_2^n$ называется *двоичным кодом* длины n, элементы кода называются *кодовыми словами*.

Двоичный линейный код, кодовые слова которого образуют линейное подпространство, называется *линейным*. Порождающая матрица линейного кода — это матрица, строки которой задают базис линейного кода.

Определение 3. *Произведением Адамара* двух векторов $x, y \in \mathbb{F}_2^n$ называется вектор, полученный в результате покомпонентного произведения координат этих векторов:

$$(x_1, \ldots, x_n) \circ (y_1, \ldots, y_n) = (x_1 y_1, \ldots, x_n y_n),$$

а произведение Адамара двух кодов C и C' есть линейная оболочка множества всех попарных произведений вида $x \circ y$, где $x \in C$, $y \in C'$.

Пусть $f : \mathbb{F}_2^n \to \mathbb{F}_2$ — булева функция от n переменных. *Алгебраической нормальной формой* (АНФ, полином Жегалкина) функции f называется многочлен вида

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{(i_1,i_2,\ldots,i_n)\in\mathbb{F}_2^n} a_{i_1 i_2 \ldots i_n} x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n},$$

где $a_z \in \mathbb{F}_2$ для любого $z \in \mathbb{F}_2^n$ (с соглашением $0^0 = 1$).

Алгебраической степенью $\deg(f)$ булевой функции f называется максимальная из степеней мономов, которые появляются в её АНФ с ненулевыми коэффициентами. Функция называется *квадратичной*, если её степень равна 2.

Определение 4. Код Рида-Маллера $RM(r, m)$ — множество всех векторов значений булевых функций от m переменных степени не выше r.

Для удобства исследования кодов Рида-Маллера порядка 2 можно представлять их в виде графов. Мономы степени $r = 1$ — это вершины графа, а мономам степени $r = 2$ соответствуют рёбра. Для описание мономов больших степеней потребуется рассмотрение гипер-рёбер.

Для кодов Рида-Маллера справедливо следующее свойство: [1]

$$(RM(2, m))^2 = RM(4, m). \tag{1}$$

Чтобы это условие выполнялось для подкода, в котором присутствуют не все возможные мономы степени $r = 2$, необходимо, чтобы в соответствующем ему графе присутствовали все такие рёбра (мономы степени 2), с помощью которых было возможно получить все кодовые слова кода $RM(4, m)$, соответствующие функциям степени $3, 4$.

Определение 5. Граф, такой, что индуцированный подграф на любых четырёх его вершинах включает совершенное паросочетание, будем называть *покрытым*.

В контексте рассматриваемой задачи такие граф представляют интерес, так как если в подкоде присутствуют мономы степени два такие, что порождаемый ими граф является покрытым, квадрат такого подкода содержит векторы значений всех мономов степени 4.

Свойства минимальных покрытых графов:

1) Степень регулярного минимального покрытого графа равна $m - 3$.

   Доказательство. Для проверки графа $G = (V, E)$ на покрытость нужно убирать $m - 4$ вершин различными способами, где m — количество вершин. Оставшиеся 4 вершины должны быть соединены двумя непересекающимися рёбрами [4]. Если у какой-либо вершины $v \in V$ будет степень $r_v \leqslant m - 4$, то при проверке на покрытость можно будет убрать все вершины, с которыми связана v. Тогда это будет изолированная вершина, следовательно, граф покрыт не будет. Значит, минимальная степень регулярного покрытого графа равна $m - 3$. □

2) Количество рёбер в минимальном покрытом графе равно $m(m - 3)/2$.

Доказательство. Так как степень каждой вершины в графе с m вершинами равна $m-3$, для поиска числа рёбер умножим эту степень на число вершин. Из-за того, что каждое ребро мы считаем дважды, поделим результат пополам и получим: $\frac{m(m-3)}{2}$. □

Первое свойство было представлено в работе [4], однако мы привели формальное доказательство данной формулы, потому что она использовалась нами в дальнейшем исследовании покрытых графов.
Второе свойство в отношении кодов Рида-Маллера определяет минимальное возможное количество мономов $\omega(2,m)$ в следующей формуле:

$$(RM(1,m) \cup \{f_1, ..., f_{\omega(2,m)}\})^2 = RM(4,m), \tag{2}$$

где f_i — мономы степени 2. Т.е. $\omega(2,m)$ — это минимальное число мономов степени больше 1, которые можно оставить в подкоде Рида-Маллера с параметром $r=2$, исключив все остальные, так, чтобы сохранилось свойство из формулы (1).

Теорема 8. [4] *Пусть выполняются условия:*

1) $\exists\, G=(V,E) : \forall\, v \in V : deg(v) = m-3$

2) $(a,b) \notin E$ *и* $(a,c) \notin E \Rightarrow (b,c) \in E$

Тогда граф G покрыт.

Доказательство. Пусть $G=(V,E)$. Рассмотрим условие (2): оно эквивалентно тому, что любой индуцированный подграф графа G с 4 вершинами имеет подграф, изоморфный покрытому графу, состоящему из четырёх вершин и двух рёбер.

Также стоит обратить внимание на то, что для выполнения условия (1) необходимо доказать, что любой моном 4-й степени можно получить как произведение двух мономов степени 2. Тогда то же самое верно и для мономов 3 степени. Покажем это: для любых мономов 3-й степени $u_1u_2u_3$ по крайней мере один из мономов u_1u_2, u_1u_3 или u_2u_3 лежит в коде. Иначе после возведения в квадрат мы бы не получили моном $u_1u_2u_3$.

Для доказательства необходимости зафиксируем любую вершину v. Тогда если бы какие-либо три ребра вида $\{v,u_i\}$ для $i=1,2,3$ отсутствовали бы, то индуцированный подграф на вершинах v,u_1,u_2,u_3 не имел бы требуемого нам подграфа. Это показывает, что $deg(v) \geq m-3$. Однако если $deg(v)=m-3$ и $\{v,u_1\} \notin E$, $\{v,u_2\} \notin E$, то $\{u_1,u_2\} \in E$, иначе ни один из индуцированных подграфов, содержащих вершины v,u_1,u_2, не будет иметь вид требуемого нам подграфа. Таким образом выполнены свойства (1) и (2).

Для доказательства достаточности зафиксируем любой индуцированный подграф с 4 вершинами (обозначим как v,u_1,u_2,u_3). Отметим, что он обладает свойствами (1) и (2) при $m=4$. Если вершина v имеет степень 1, то есть $\{v,u_1\} \in E$, но $\{v,u_2\} \notin E$ и $\{v,u_3\} \notin E$, то из условия (2) следует, что $\{u_1,u_2\} \in E$. Таким образом у нас есть рёбра $\{v,u_1\}$ и $\{u_1,u_2\}$, необходимые для соответствующего подграфа. □

Примеры минимальных покрытых графы для произвольного m можно построить по общему алгоритму $\mathbb{A}$. Он состоит в слдедующем: необходимо построить полный граф с последовательно пронумерованными вершинами, исключив из него контур.

Теорема 9. *Алгоритм* $\mathbb{A}$ *корректен.*

Доказательство. Минимальность графа, построенного по данному методу, легко объясняется тем, что степени всех вершин в минимальном покрытом графе должны быть равны $m-3$. Очевидно, что данное условие выполняется, если каждую вершину соединить рёбрами со всеми, кроме двух соседних. Для доказательства покрытости рассмотрим дополнительный граф к графу, построенному по приведённому методу. Это будет контур. Для графа, являющегося дополнительным к контуру, выполняется инвертированное условие (2) из теоремы 8, то есть в таком графе отсутствуют треугольники.

Получается, что выполняются оба условия теоремы 8. Следовательно, граф, построенный приведённым методом, является минимальным покрытым.

□

По данному алгоритму была написана программа на языке Python. В ней реализованы построение минимальных покрытых графов для произвольных m и проверка графа на покрытость. К примеру, для $m=5$, $m=10$, $m=20$ (рис. 9).

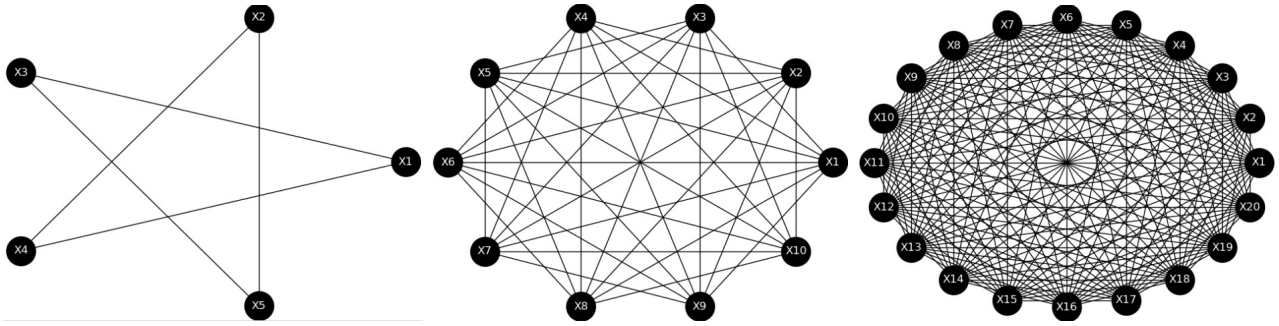

Рис. 9: Графы с $m=5$, $m=10$, $m=20$

При умножении произвольной обратимой матрицы S на попрождающую матрицу G кода Рида-Маллера $RM(r,m)$ получаемая матрица также будет порождающей матрицей этого кода. При этом ее строки будут соответствовать другому базису кода Рида-Маллера. Расссматривалась задача нахождения максимального числа строк p, которые можно вычеркнуть из обратимой матрицы S так, чтобы ранг квадрата Адамара кода, порождающей матрицей которого является произведение $S'G$, равнялся 16.

Для ответа на этот вопрос была построена программа, которая генерирует произвольную невырожденную матрицу S, затем строит невырожденную матрицу S' путем вычёркивания из S заданного числа p строк. Затем рассматривается линейный код C, являющийся подкодом кода Рида-Маллера $RM(2,m)$, порождающая матрица которого есть $S'G$, и проверяется условие $C^2 = RM(4,m)$.

Проведённые эксперименты показали, что для кода $RM(2,4)$ существуют такие матрицы S, что при вычёркивании $p = 1,2,3,4,5$ числа произвольных строк матриц S ранг квадрата кода с порождающей матрицей $S'G$ равен 16. Таким образом, квадрат полученного подкода является кодом $RM(4,4)$. Результаты вычислений и примеры таких матрицы для параметров $(r,m)=(2,4)$ представлены ниже (рис. 10).

Для $p \geqslant 6$ не удалось получить матрицу S'.

Также вызывают интерес вопросы, связанные с изучением редуцированных базисов кода Рида-Маллера таких, что квадрат соответствующего подкода совпадает с кодом Рида-Маллера. Он обу-

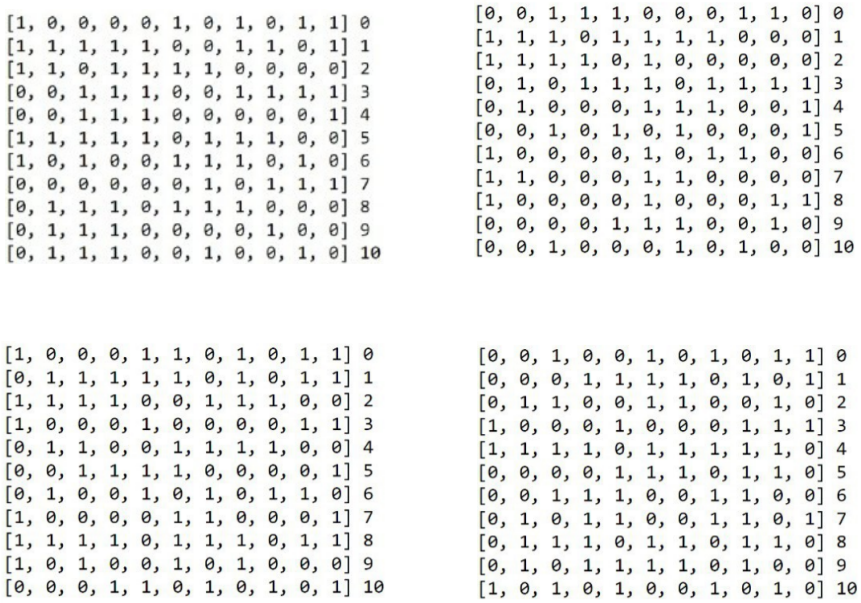

Рис. 10: Примеры матриц с параметрами $P = 2$, $P = 3$, $P = 4$, $P = 5$

словлен тем, что для криптосистем, основанных на использовании «слабых» подкодов кода Рида-Маллера, задача криптоанализа сводится к анализу криптосистемы на полном коде, для которой, в свою очередь, известен ряд эффективных атак.

Возникает проблема определения минимальной размерности подкода, который может обладать тем свойством, что его квадрат совпадает с кодом Рида-Маллера.

Всюду далее для множества полиномов M через $L(M)$ обозначается линейная оболочка данного множества.

Теорема 10. *Пусть*

$$K = L(RM(r, m) \cup \{a_1, a_2, \ldots, a_{\omega(r+1,m)}\}), \tag{3}$$

где a_i — моном степени $r + 1$, $i = 1, 2, \ldots, \omega(r + 1, m)$ и выполняется $K^2 = RM(2r + 2, m)$. Тогда для

$$K_1 = L(RM(r, m) \cup \{f_1, f_2, \ldots, f_{w(r+1,m)}\}),$$

где $f_i = a_i \oplus P_i$, а P_i — произвольный полином степени не выше r, справедливо $K_1^2 = RM(2r+2, m)$.

Доказательство. Стандартный базис кода $RM(r, m)$ образован векторами значений мономов $\{1, x_1, x_2, \ldots, x_m, x_1x_2, x_1x_3, \ldots, x_1x_m, \ldots, x_1 \ldots x_m\}$. Квадрат кода K_1 состоит из векторов значений следующего объединения полиномов:

$$L\left(RM(2r, m) \cup \bigcup_{i=1}^{\omega(r+1,m)} RM(r, m) \cdot f_i \cup \{f_i f_j | i, j = 1, 2, \ldots, \omega(r + 1, m)\}\right).$$

В силу того, что код $RM(2r, m)$ является подкодом K_1^2, приведенную выше линейную обочку можно представить в следующем виде:

$$L\left(RM(2r, m) \cup \bigcup_{i=1}^{\omega(r+1,m)} RM(r, m) \cdot a_i \cup \{f_i f_j | i, j = 1, 2, \ldots, \omega(r + 1, m)\}\right).$$

Так как K_1 получено из K, а по условию в K^2 есть векторы значений всех мономов степени $2r+1$, то в K_1^2 также есть векторы значений всех мономов степени $2r + 1$. В свою очередь, в K^2 есть векторы

значений всех мономов степени $2r+2$ и в K_1^2 есть векторы значений всех мономов степени $2r+1$, в K_1^2 есть векторы значений всех мономов $\{a_i a_j | i,j = 1,2,\ldots,\omega(r+1,m)\}$. Следовательно, в K_1^2 есть векторы значений всех мономов степени $2r+2$. □

Обозначим через B линейный код, порождаемый векторами значений мономов вида

$$\{1, x_1, x_2, x_3, x_4\} \cup \{x_1x_2, x_3x_4\}.$$

Предложение 4. *Для линейного кода C, порождаемого векторами значений полиномов вида*

$$\{1, l_1(x), l_2(x), l_3(x), l_4(x)\} \cup \{x_1x_2, x_3x_4\},$$

где $A = (a_{ij})$ — произвольная обратимая двоичная матрица порядка 4×4 и $l_i(x) = \bigoplus_{j=1}^{4} a_{ij}x_j$, $i = 1,2,3,4$, справедливо $C^2 = RM(4,4)$.

Доказательство. Заметим, что $B^2 = RM(4,4)$, следовательно, в C^2 также есть вектор значений монома $x_1x_2x_3x_4$, так как из порождающего множества меняются только полиномы 1-й степени. Также нетрудно видеть, что код $RM(2,4)$ является подкодом кода C^2. В силу этого факта, для доказательства утверждения достаточно изучить наличие в коде C^2 всех возможных полиномов степени не выше 3.

Рассмотрим 4 полинома, полученных произведением полиномов l_i степени 1 и одного из мономов степени 2 (для второго рассуждения будут аналогичными). Из этих полиномов хотелось бы выразить 2 монома степени 3 (если моном степени 2 был x_1x_2, то пытаемся получить $x_1x_2x_3$ и $x_1x_2x_4$). В представленных полиномах эти мономы входят с некоторыми коэффициентами, которые заданы соответствующими столбцами матрицы A (т.е. они могут входить в полином вместе, если соответствующие элементы равны единицам). Но известно, что матрица обратимая, то есть ее столбцы линейно независимы. А это значит, что из них можно выделить обратимую подматрицу 2×2. Тогда линейная комбинация строк этих подматриц может породить любые значения, в том числе искомые: $1,0$ и $0,1$ (тогда в результате умножения матрицы на мономы степени 2 получатся полиномы степени 3, в которых будет только один моном степени 3, а остальные могут быть исключены линейными комбинациями с полиномами младших степеней).

Таким образом, получаем, что $C^2 = RM(4,4)$. □

Заключение

В рамках настоящей работы были исследованы подкоды кодов Рида-Маллера $RM(2,m)$ и сделаны некоторые выводы о структуре слабых подкодов. Дальнейшие планы включают продолжение работы над их описанием и изучением свойств, что в перспективе позволит использовать для криптосистемы Мак-Элиса попрождающие матрицы на основе подкодов кодов Рида-Маллера.

ЛИТЕРАТУРА

[1] Shor P. V., Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing, **26**:5 (1997), 1484–1509.

[2] Minder L., Shokrollahi A., Cryptanalysis of the Sidelnikov Cryptosystem, Advances in Cryptology — EUROCRYPT 2007, EUROCRYPT 2007, Lecture Notes in Computer Science, **4515**, eds. Naor M., Springer, Berlin, Heidelberg, 2007, 347–360.

[3] Borodin M., Chizhov I., Effective attack on the McEliece cryptosystem based on Reed-Muller codes, Discrete Mathematics and Applications, 24:5 (2014), 10–20.

[4] Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. Теория кодов, исправляющих ошибки. М.: Связь, 1979

[5] Блейхут Р. Теория и практика кодов, контролирующих ошибки. М.: Мир, 1986

[6] Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. М.: Мир, 1976

[7] Vysotskaya V. V. Characteristics of Hadamard Square of Reed — Muller Subcodes of Special Type. https://eprint.iacr.org/2020/507.

Кураторы исследования:
аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН Куценко Александр Владимирович;
аспирантка ВМК МГУ им. М.В. Ломоносова, специалист-исследователь лаборатории криптографии НПК "Криптонит" (г. Москва) Высоцкая Виктория Владимировна.

БЛОКЧЕЙН-ТЕХНОЛОГИИ

Реализация алгоритмов обеспечения приватности в смарт-контрактах Ethereum

Е.А. Волкова, А.А. Григорьевская, М.С. Ерценкин, П.А. Кайдаш, А.Е. Киреева,
С.О. Кирсанов, Д.А. Пешков, В.В. Скудина, А.Ю. Чанчиков, А.В. Черкашин.
Новосибирский государственный университет
E-mail: **volkova1@g.nsu.ru, a.grigorevskaya@g.nsu.ru, misha_ertcenkin@mail.ru, p.kaidash@g.nsu.ru, a.kireeva@g.nsu.ru, s.kirsanov@g.nsu.ru, pyatyorki@yandex.ru, v.skudina@g.nsu.ru, a.chanchikov@g.nsu.ru, a.cherkashin1@g.nsu.ru**

Аннотация

В рамках данного проекта был исследован принцип работы децентрализованных блокчейн-систем, в особенности, платформы Ethereum, а также устройство смарт-контрактов и язык Solidity для их написания. Для создания смарт-контрактов с использованием протокола Zero-Knowledge Proof также необходим высокоуровневый язык ZoKrates, который был изучен нами с нуля. Было изучено устройство и работа его библиотек. В результате проекта в стандартную библиотеку была добавлена реализация функции конъюнкции для массива значений.

Ключевые слова: *блокчейн, распределенные системы, Ethereum, конфиденциальность, доказательство с нулевым разглашением, zk-SNARK, ZoKrates*

Блокчейн — это разновидность децентрализованных систем, которая занимается сбором данных, их хранением и управлением.

Существуют три основные проблемы у современного блокчейна — это проблемы конфиденциальности, эффективности и масштабируемости . Проблема масштабируемости возникает из-за того, что концепция требует обработки транзакций на каждом узле системы. Отсюда следует неэффективность системы блокчейна, так как если каждый узел сети делает одно и то же, то очевидно, что пропускная способность всей сети равна пропускной способности одного узла. Проблема отсутствия конфиденциальности заключается в том, что данные блокчейна являются общедоступными, ведь они должны обрабатываться на каждом узле.

Именно эти три проблемы считаются основным препятствием для внедрения блокчейна в разные области науки и бизнеса. Основная работа велась с блокчейном на платформе Ethereum.

Ethereum — криптовалюта и платформа для создания децентрализованных онлайн-сервисов на базе блокчейна (децентрализованных приложений), работающих на базе смарт-контрактов. Реализована как единая децентрализованная виртуальная машина. Концепт был предложен основателем журнала Виталиком Бутериным в конце 2013 года, сеть была запущена в 2015 году.

Ethereum изначально создавался не столько как платёжная система, сколько как база для доступного внедрения технологии блокчейн в сторонние проекты. К нему проявили интерес не только новые стартапы, но и крупные разработчики (Microsoft, IBM). Также Ethereum получило прозвище bitcoin 2.0.

В отличие от Bitcoin у Ethereum скорость совершения транзакции выше в 8 раз. Преимуществом Ethereum также является возможность создания смарт-контрактов — это такой электронный алгоритм или условие, при выполнении которого стороны могут обмениваться деньгами или другими активами. Изначально протокол биткоина не предполагался как протокол смарт-контрактов – а лишь для передачи самых простых данных (входов и выходов транзакций). Однако, на блокчейне биткоина все же можно исполнять простейшие логические операции и фактически создать несколько вариантов смарт-контрактов, не обладавших полнотой по Тьюрингу.

Ethereum оснащен встроенным Тьюринг-полным языком программирования, позволяющим любому желающему писать смарт-контракты и децентрализованные приложения, где каждый сможет создавать собственные произвольные правила владения, форматы транзакций и произвольные функции изменения состояния. Смарт-контракты, криптографические «коробки», содержащие значение и открывающиеся только при определённых условиях, также могут быть построены поверх платформы Ethereum, причем со значительно большей функциональностью, чем та, что предложена в языке сценариев Bitcoin.

Проблема отсутствия конфиденциальности на платформе Ethereum решается с помощью алгоритмов сокрытия данных в распределенных системах.

ZoKrates - это набор инструментов для zk-SNARK на платформе Ethereum. Так как Ethereum выполняет вычисления на всех узлах сети, что приводит к высоким затратам, ограничениям сложности и низкой конфиденциальности, а zk-SNARK позволяет проверять вычисления в цепочке только за небольшую часть стоимости их выполнения, то работа с такими вычислениями становится чрезвычайно трудоемкой. Но данную проблему решает ZoKrates. Он помогает связывать с блокчейном Ethereum программы, которые так же с его помощью создаются вне сети. Это позволяет расширить возможности децентрализованных приложений.

На данный момент у ZoKrates существует возможность создавать простые схемы. Но проблема заключается в том, что реализация алгоритмов не автоматизирована. То есть необходимо генерировать новые параметры при изменении смарт-контракта. К тому же реализация сложных схем пока невозможна.

Определение zk-SNARK:
Generator (C circuit, λ is):
(pk, vk) = G(λ, C)
Prover (x pub inp, w sec inp):
π = P(pk, x, w)
Verifier:
V(vk, x, π) == ($\exists$ w s.t. C(x, w))
zk-SNARK включает в себя три алгоритма: G, P, V:

1. Генератор ключей G, принимающий секретный параметр lambda и программу C, генерирует доказательный ключ pk и ключ проверки vk. Данные ключи являются общедоступными параметрами и создаются только один раз для программы C.

2. Доказывающий алгоритм P, принимающий ключ pk, публичное значение x и секретное значение w. Алгоритм генерирует доказательство prf = P(pk, x, w) того, что доказывающий знает секретное значение w, удовлетворяющее программе.

3. Проверяющий алгоритм V вычисляет V(vk, x, prf) результатом которого будет “true” или “false”. То есть, если секретное значение w удовлетворяет C(x,w), то алгоритм возвращает “true” и в противном случае - “false”.

На реальных приложениях затрудняет использование zk-SNARK секретный параметр lambda. Любой, кто знает этот параметр, может генерировать поддельные доказательства. В частности, при любой программе C и публичном значении x человек, который знает lambda, но не знает секретного w, может создать доказательство fake_prf, на которое алгоритм V(vk, x, fake_prf) вернет true.

zk-SNARK — это захватывающая технология, которая позволит достичь доверия в вычислительных системах, не достигнутое ранее. Эта идея особенно важна, поскольку организации, которые имеют возможности фальсификации информации, не смогут сделать это при использовании данной

системы защиты. Для внедрения этого типа технологии потребуется время, однако, преимущества будут значительными и создадут нечто действительно уникальное: проверяемое доверие к системе.

zk-SNARK в Ethereum выглядит следующим образом: блоки алгоритма проверки добавляются в Ethereum в виде предварительно скомпилированных контрактов. Используется следующее: генератор запускается вне блокчейна, чтобы создать доказательный ключ и ключ проверки. Любой доказывающий может затем использовать доказательный ключ, чтобы создать доказательство, также вне сети. Затем общий алгоритм проверки может выполняться внутри смарт-контракта, используя в качестве входных параметров доказательство, ключ проверки и публичное значение. Результат алгоритма проверки затем может быть использован для запуска других действий на блокчейне.

Нашей целью было научиться писать на языке ZoKrates и дополнить его библиотеку новым функционалом, так как текущая реализация алгоритмов доказательства с нулевым разглашением в ядре Ethereum предоставляет разработчикам смарт-контрактов довольно ограниченные возможности по работе с приватными данными.

Задачи:

1. Познакомиться с внутренним устройством блокчейн-платформы Ethereum.
2. Разобрать механизмы работы смарт-контрактов и язык Solidity.
3. Изучить криптографический протокол обеспечения приватности zk-SNARK.
4. Написать функцию для стандартной библиотеки ZoKrates.
5. Собрать итоговый проект и протестировать работу.

Создание и добавление новой функции в библиотеку языка ZoKrates. Задача: дан одномерный массив заданной длины типа “bool”. Необходимо произвести конъюнкцию, аргументами которой являются значения всех элементов данного массива.

Оптимальная функция, которая бы выполняла поставленную задачу для больших массивов, не была реализована в Zokrates. Наша команда реализовала алгоритм, который перемножает все элементы массива и выводит “true”, если все элементы равны “true”, или “false” в противном случае.

Функция работает по законам алгебры логики - логическое «И» заменяется умножением, а значения false и true - на нуль и единицу соответсвенно и производится перемножение всех элементов массива. Если массив полностью состоит из переменных со значением true - произведение единиц дает единицу (=true), иначе - нуль (=false).

Особенность заключается в том, что до этого в языке был функционал для проверки логического умножения только двух переменных (логическое «И»).

Реализованная функция работает корректно. Чтобы убедиться в корректности, члены нашей команды разработали тесты.

Алгоритм был написан в виде программы на языке ZoKrates, файл с данной функцией был добавлен в стандартную библиотеку stdlib проекта ZoKrates.

В процессе обучения участники проекта занимались следующим списком задач:

1. Изучение теории работы блокчейн-технологии и историю их появления.
2. Для работы с ZoKrates был изучен протокол доказательства знания с нулевым разглашением zk-SNARK.

3. Далее мы взялись за изучение IDE Remix и языка Solidity, которые предназначены для платформе Ethereum. Прошли курсы по синтаксису Solidity и написания смарт-контрактов, компилилировали и развертывали их (тестировали правильность работы). Были изучены онлайн-уроки по синтаксису Solidity, а полученные знания применены на практике с использованием тестового эфира и тестовой сети Ropsten.

4. После всего этого перешли к изучению уже самого языка ZoKrates, а именно - синтаксиса, процесса сборки и компиляции проекта через консоль и способов добавления функции, которую написали, в библиотеку проекта. Для этого мы использовали IDE Clion и язык программирования Rust.

Но не обошлось и без трудностей в работе, таких как:

1. Малая аудитория ZoKrates, а значит - не всегда возможно найти ответы на то, почему код или сборка проекта может выдавать ошибки и как это можно устранить.

2. Сбои в работе сервисов по выдаче тестового эфира, который нужен для того чтобы проверять корректность работы написанных смарт-контрактов.

3. Большинство инструкций по работе написаны для ОС Linux, в то время как у большинства участников проекта - Windows.

В итоге мы реализовали функцию, которая проверяет истинность всех элементов массива. Это поможет разработчикам, которые часто используют в своем коде такие проверки и вынуждены писать их каждый раз с нуля.

Дальнейшие планы:

1. Реализация других логических операций для упрощения написания программ.

2. Оптимизация имеющихся алгоритмов.

ЛИТЕРАТУРА

[1] Vitalik Buterin — Ethereum Whitepaper [Электронный ресурс] // URL: https://ethereum.org/en/whitepaper/

[2] Christian Lundkvist — Introduction to zk-SNARKs with Examples [Электронный ресурс] // URL: https://media.consensys.net/introduction-to-zksnarks-..

[3] Ariel Gabizon — Explaining SNARKs Part I: Homomorphic Hidings [Электронный ресурс] // URL: https://electriccoin.co/blog/snark-explain/

[4] Source codes of Zokrates [Электронный ресурс] // URL: https://github.com/Zokrates/ZoKrates

[5] Official documentation of Zokrates [Электронный ресурс] // URL: https://zokrates.github.io/introduction.html

[6] Edi Sinovcic — ZoKrates — zkSNARKs On Ethereum (made easy) [Электронный ресурс] // URL: https://medium.com/coinmonks/zokrates-zksnarks-on-eth..

[7] Cornell Blockchain — A Brief Dive Into zk-SNARKs and the ZoKrates Toolbox on the Ethereum Blockchain [Электронный ресурс] // URL: https://medium.com/cornellblockchain/a-brief-dive-int..

[8] Official documentation of Solidity [Электронный ресурс] // URL: https://docs.soliditylang.org/en/v0.8.6/index.html

Куратор исследования – Кондырев Дмитрий Олегович, аспирант ФИТ НГУ, ассистент кафедры систем информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ

Система децентрализованного взаимодействия между компьютерами на основе протокола Kademlia

В.С. Никифоров[1], А.П. Евсюков[1], А.В. Головнина[1], Т.Д. Калмыков[2], С.Д. Атконов[3].
[1]Новосибирский государственный университет
[2]Томский государственный университет
[3]Лицей Информационных Технологий г. Новосибирск

Аннотация

В настоящее время P2P-сети используются для разделения файлов, параллельного программирования, распределенного кэширования ресурсов для разгрузки серверов, рассылки уведомлений и статей, поддержки системы доменных имен, индексирования распределенных ресурсов и их поиска, резервного копирования и создания устойчивых распределенных хранилищ данных, обмена сообщениями, создания систем, устойчивых к атакам типа “отказ в обслуживании”, распространения программных модулей. Имеется большое число клиентских программ для работы с P2P-сетями, как коммерческих, так и с открытым кодом. Постоянно идет работа по усовершенствованию протоколов и увеличению функциональности систем. В рамках проекта был реализован протокол для работы с децентрализованными сетями Kademlia, на основе которого сделана широковещательная маршрутизация.
Ключевые слова: *Peer-to-Peer network, Kademlia, DHT, Broadcasting.*

Определение 6. P2P сеть - это бессерверная сетевая технология, которая позволяет нескольким устройствам совместно использовать ресурсы и общаться напрямую друг с другом без посредника.

Определение 7. Распределенная хэш-таблица (DHT) - это распределенная система, предоставляющая службу поиска, аналогичную хэш-таблице: пары ключ-значение хранятся в DHT, и любой участвующий узел может эффективно извлекать значение, связанное с данным ключом.
Распределенные хэш-таблицы (DHT) предназначены для поиска объектов в распределенных средах, таких как одноранговые (P2P) системы, без необходимости централизованного сервера.

В основе работы протокола и построения таблиц лежит определение расстояния между узлами через XOR-метрику: $d(x, y) = x \oplus y$.

XOR-метрика удовлетворяет всем аксиомам метрики:

1. $d(x, y) >= 0$
2. $d(x, y) == 0 <=> x == y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) <= d(x, z) + d(z, y)$

На вычислении расстояний между узлами (посредством применения метрики к их идентификаторам) базируется алгоритм построения таблиц маршрутизации. Каждый *i*-ый столбец таблицы ответственен за хранение информации об узлах на расстоянии от 2^{i+1} до 2^i. *K* – ограничение на число хранимых узлов на каждом уровне, столбцы таблицы, ограниченные таким *K*, называют *K*-buckets.

Протокол Kademlia содержит 4 типа сообщений:

- **PING** – запрос, использующийся для проверки существования конкретного узла в сети;
- **STORE** – запрос, позволяющий разместить информацию на заданном узле;
- **FIND_VALUE** – запрос, позволяющий найти значение по ключу;
- **FIND_NODE** – запрос, используемый для поиска ближайших K к заданному идентификатору.

В рамках летней школы при реализации проекта использовались языки программирования Kotlin для описания P2P сети и Java-Script с библиотекой React для разработки пользовательского интерфейса.

ЛИТЕРАТУРА

[1] Maymounkov, P., & Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. Peer-to-Peer Systems, 53–65. https://doi.org/10.1007/3-540-45748-8_5

[2] Pita, I., & Fernández-Camacho, M.-I. (2013). Formal Specification of the Kademlia and the Kad Routing Tables in Maude. Recent Trends in Algebraic Development Techniques, 231–247. https://doi.org/10.1007/978-3-642-37635-1_14

[3] Czirkos, Z., & Hosszú, G. (2013). Solution for the broadcasting in the Kademlia peer-to-peer overlay. Computer Networks, 57(8), 1853–1862. https://doi.org/10.1016/j.comnet.2013.02.021

Куратор исследования – И.В. Валиахметов

Исследование методов source-to-source обфускации. Разработка обфускатора с использованием расширяемой инфраструктуры.

И.А. Матеюк
Новосибирский государственный университет
E-mail: i.mateyuk@g.nsu.ru

Аннотация

В рамках данной темы было проведено исследование различных методов source-to-source обфускации и последующее внедрение части из них при разработке расширяемого C/C++ обфускатора. Так как фундаментальным конкурентным преимуществом являлась разработка на базе расширяемой инфраструктуры, то большая часть работы пришлась на транслятор Clang. В качестве инструмента для работы с Clang использовалась библиотека LibTooling и непосредственно C++ 14 версии. В результате работы удалось применить преобразования переменных, имен функций и строковых констант, тем самым затрудняя анализ исходного кода для злоумышленника.

Ключевые слова: *обфускация, source-to-source, Clang, LibTooling*

Введение.

Проблема безопасности в IT, а также нарушения авторских прав до сих пор стоит очень остро. Доступ к исходному коду проприетарного программного обеспечения может повлечь убытки, связанные с пиратством и кражей интеллектуальной собственности, например, алгоритмов, бизнес-логики. Также страдают и конечные пользователи, они могут подвергаться атакам злоумышленников. Так, например, злоумышленник может найти и использовать уязвимости в коде, которые могут повлечь за собой значительный ущерб, или исследовать выпущенные критические обновления для последующей атаки на еще не обновившихся пользователей. Для борьбы с описанными выше явлениями разработчики оставляют исходный код закрытым и стараются не допустить его утечек, в мире информационной безопасности это называют security through obscurity — обеспечение безопасности через сокрытие внутреннего устройства системы. Эффективность такого метода защиты довольно не высока, поскольку, имея конечный продукт, специалисты могут провести его реверс-инжиниринг с целью обхода защиты, создания аналога программного обеспечения или поиска уязвимостей для их дальнейшего эксплуатирования. При проведении реверс-инжиниринга используют такие методики как дизассемблирование или декомпиляцию машинного кода, в результате которых злоумышленники могут попытаться восстановить исходный код. Для того чтобы избежать этого, исходный код можно подвергнуть обфускации. Обфускация или запутывание кода — метод приведение исходного или исполняемого кода программы к виду, который позволяет сохранить изначальную функциональность, но сильно затрудняющему анализ и понимание алгоритмов работы программного обеспечения при декомпиляции. Обфускация не делает анализ и взлом программы невозможной, но делает этот процесс слишком трудоемким и ресурсозатратным, чего хватает в большинстве случаев для избежания каких-либо проблем.

Цель: Изучение классических методов source-to-source обфускации, разработка обфускатора C/C++ кода на базе расширяемой архитектуры.

Задачи: Изучить теоретическую информацию о методах обфускации программного кода; Исследовать алгоритмы, позволяющие препятствовать реверс инжинирингу; Поиск и анализ уже существующих инструментов для обфускации и деобфускации программного кода; Выбор расширяемой инфраструктуры как основы; Непосредственно разработка.

Выбор архитектуры.

Доступность качественных средств анализа кода и большой выбор подключаемых модулей, в автоматическом режиме обходящих многие приемы противодействия анализу, понижают планку требований к квалификации аналитика, что ведет к повышению требований к защите программ. Необходимо использовать либо методы противодействия анализу, неизвестные широкому кругу лиц, либо использовать трудоемкие для анализа преобразования. Оптимальным выбором, позволяющим реализовать стойкие варианты запутывания программ, является создание обфускатора на базе одной из существующих компиляторных инфраструктур. С одной стороны, это позволит производить запутывание программы, имея полную информацию о ней на всех этапах компиляции, а с другой позволит сосредоточиться на разработке защиты, а не на создании требуемой инфраструктуры. Из-за сложности синтаксического анализа кода C и C ++ разработка полнофункционального парсера заняла бы много времени. Его можно сократить, реализовав простой синтаксический анализатор, который выбирает только желаемые элементы синтаксиса в исходном файле, но в нем всегда будут отсутствовать важные функции для некоторых пользователей. У такого парсера правила всегда были бы грубыми и без учета полной грамматики языка C++ и с самого начала были бы обречены. Инфраструктура, на базе которой будет разрабатываться запутывающий компилятор, должна удовлетворять следующему набору требований:

1. обеспечивать компиляцию исходных кодов на C/C++ под Windows и Linux;
2. иметь открытые исходные коды;
3. иметь документацию и поддержку сообщества;
4. иметь возможность расширяемости;
5. возможность влиять на генерируемый код на любом этапе компиляции, от препроцессора до генерации кода;
6. возможность получить различную информацию об обрабатываемой программе на любой стадии компиляции, требуемую для реализации алгоритмов запутывания кода.

В качестве такой инфраструктуры выбор пал на комбинацию Clang с использованием libTooling. Такой выбор позволяет обеспечить основную цель работы - открытость и расширяемость обфускатора.

Методы обфускации.

Перенос локальных переменных в глобальную область видимости.

Перенос локальных переменных в глобальную область видимости с последующим их использованием в разных функциях производится с целью затруднить точный анализ потоков данных в программе. В общем случае нельзя изменять значения переменных, вынесенных из других функций в произвольном месте программы, так как это может привести к неправильному выполнению компилируемой программы. Поэтому строится граф вызовов для всех функций в модуле, затем для каждой функции вычисляется множество переменных, модификация которых не нарушит работоспособность программы. Такими переменными будут переменные, вынесенные из функций, расположенных на разных путях в дереве вызовов. После формирования множеств подходящих переменных осуществляется добавление мусорного кода, использующего для вычислений «безопасные» переменные. Также найденные переменные используются в предикатах. Функции, передаваемые по адресу в другие функции, не обрабатываются, так как они могут использоваться в многопоточном коде. При восстановлении алгоритма работы программы используется построение

слайсов программы. Выполняется отбор тех операторов программы, выполнение которых влияет на выходные данные или на выполнение которых повлияли входные данные. Во время статического анализа для переменных, расположенных в глобальной области памяти, требуется проводить межпроцедурный анализ.

Шифрование строк

Во время статического анализа строковые константы, хранящиеся в открытом виде, могут дать аналитику дополнительную информацию о функционировании программы или помочь найти интересующий код по строкам, выводимым во время интересующего его события. Преобразование, маскирующее строковые константы, предназначено для сокрытия информации о строках во время статического анализа программы. Шифрование строк выполняется следующим образом: вначале все константные строки, кроме тех, что содержатся в агрегатных типах (массивы, контейнеры из стандартной библиотеки), шифруются, в модуль добавляются шифрующая и дешифрующая функции. Перед каждым использованием той или иной строки вставляется вызов функции дешифратора, а после – шифрующей функции. Это справедливо для строк, для которых не выполняются операции с указателями. Если же такие операции имеют место, то для корректной работы запутывающего алгоритма необходим анализ указателей. В таких случаях обратного шифрования строки не производится. Шифрование строк после использования требуется для того, чтобы во время работы программы все строки не находились в памяти расшифрованными. Шифрование строк производится с помощью операции XOR со случайным ключом.

Вставка фиктивных циклов

Фиктивный цикл – цикл, в котором никогда не происходит более одной итерации. В коде запутываемой программы происходит поиск участков кода, по структуре напоминающих одну итерацию цикла. Далее в начало участка или в его конец (в зависимости от типа фиктивного цикла) вставляется базовый блок с условным переходом в противоположный конец участка. Условный переход содержит в себе непрозрачный предикат, который и маскирует лишь одно исполнение цикла. В качестве подходящего участка рассматривается участок с одним входом и выходом.

Переплетение функций

Классический подход к переплетению функций обладает малой стойкостью. Он предполагает объединение сигнатур функций и наличие параметра, по которому происходит диспетчеризация. Восстановить исходный код переплетенных таким образом функций не составляет особого труда. Предложена модификация упомянутого алгоритма таким образом, чтобы, помимо диспетчеризующего условия, переплетаемые функции имели точки пересечения потоков управления и потоков данных. Тогда применение алгоритма обратного слайсинга не позволяет найти единственную точку, в которой производится выбор рабочей функции. Переплетение происходит следующим образом: 1) Объединяются сигнатуры двух функций, генерируется дополнительный параметр, по которому в процессе выполнения будет производиться выбор функции.

2) Если функции возвращают целочисленное значение, то для реального возврата значения из переплетенной функции используются глобальные перемененные, а сама функция возвращает неиспользуемое значение. Если функции возвращают указатели, то тип возвращаемого значения переплетенной функции становится указателем на void.

3) В новой функции, полученной на основе переплетения двух функций, произвольно выбираются по одному блоку из каждой функции, затем над ними производится преобразование зацепления дуг. В генерируемом общем базовом блоке производятся вычисления с глобальными переменными. Для затруднения анализа потоков данных эти переменные используются для вычислений в и других функциях модуля. Таким образом, у двух переплетенных функций всегда будут общие вычисления. Результат вычислений используется в качестве возвращаемого значения, а также

записывается в глобальную переменную, что не позволит исключить добавленные вычисления как мертвый код, результат которого нигде не используется.
4) После генерации переплетенной функции все места вызова оригинальных функций заменяются на вызов переплетенной функции с генерацией дополнительных параметров и изменением обработки возвращаемого значения.

Заключение.

В ходе летней школы были успешно выполнены все поставленные задачи. Была изучена и описана теоретическая информация по способам реверс-инжиниринга и методам обфускации противостоящим им. Также были выделены ключевые особенности и трудности, с которыми можно столкнуться в ходе разработки, была выбрана инфраструктура для разработки, реализован прототип программы с минимальным функционалом – с преобразованием переменных, имен функций и строковых констант. В дальнейшем планируется развитие и расширение обфускатора путем добавления в него новых способов для борьбы с реверс-инжинирингом.

ЛИТЕРАТУРА

[1] Dominik Picheta – Code Obfuscation for the C/C++ Language, 2018.

[2] Лебедев Р. К. – Практическая информационная безопасность, курс лекций, 2019.

[3] N. Fujieda, T. Tanaka, and S. Ichikawa – Design and implementation of instruction indirection for embedded software obfuscation, vol. 45, pp. 115–128, 2016.

[4] V.Ivannikov, S.Kurmangaleev, A.Belevantsev – Implementing Obfuscating Transformations in the LLVM Compiler Infrastructure, 2014.

Куратор исследования – б/с, И.В. Валиахметов.

Список авторов:

1. **Malanda Nomonde Sharon** – Механико-математический факультет, Новосибирский государственный университет, 1 курс аспирантуры
2. **Атконов Сергей Дмитриевич** – Лицей Информационных Технологий, 10 класс
3. **Атутова Наталья Дмитриевна** – Механико-математический факультет, Новосибирский государственный университет, 3 курс бакалавриата
4. **Бахарев Александр Олегович** – Механико-математический факультет, Новосибирский государственный университет, 3 курс бакалавриата
5. **Бонич Татьяна Андреевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс магистратуры
6. **Быков Денис Александрович** – Механико-математический факультет, Новосибирский государственный университет, 3 курс бакалавриата
7. **Волкова Екатерина Алексеевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
8. **Гилязов Артур Юрисович** – Факльтет информационных технологий, Новосибирский государственный университет, 3 курс бакалавриата
9. **Головнина Алина Владимировна** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
10. **Григорьевская Анжелика Анатольевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
11. **Гринчуков Владимир Владиславович** – Компьютерная безопасность, Балтийский федеральный университет имени Иммануила Канта, 3 курс бакалавриата
12. **Евсюков Анатолий Павлович** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
13. **Ерценкин Михаил Сергеевич** – МАОУ ОЦ "Горностай 11 класс
14. **Зюбина Дарья Александровна** – Факультет информационных технологий, Новосибирский государственный университет, 4 курс бакалавриата
15. **Кайдаш Полина Андреевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
16. **Калмыков Тимур Денисович** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 1 курс бакалавриата
17. **Киреева Александра Евгеньевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
18. **Кирсанов Семён Олегович** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата

19. **Корнилов Александр Павлович** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет, 3 курс бакалавриата

20. **Матеюк Илья Анатольевич** – Факультет информационных технологий, Новосибирский государственный университет, 3 курс бакалавриата

21. **Мокроусов Антон Сергеевич** – Факультет информационных технологий, Новосибирский государственный университет, 4 курс бакалавриата

22. **Никифоров Владислав Сергеевич** – Факльтет информационных технологий, Новосибирский государственный университет, 3 курс бакалавриата

23. **Панферов Матвей Андреевич** – Механико-математический факультет, Новосибирский государственный университет, 2 курс магистратуры

24. **Парфенов Денис Романович** – Факультет информационных технологий, Новосибирский государственный университет, 4 курс бакалавриата

25. **Пешков Данила Александрович** – Лицей №6 города Бердск, 11 класс

26. **Скудина Виктория Викторовна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата

27. **Соколенко Анастасия Алексеевна** – Компьютерная безопасность, Балтийский федеральный университет имени Иммануила Канта, 3 курс бакалавриата

28. **Сутормин Иван Александрович** – Механико-математический факультет, Новосибирский государственный университет, 1 курс магистратуры

29. **Хильчук Ирина Сергеевна** – Механико-математический факультет, Новосибирский государственный университет, 1 курс магистратуры

30. **Чанчиков Антон Юрьевич** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата

31. **Черкашин Антон Вадимович** – Физический факультет, Новосибирский государственный университет, 2 курс бакалавриата

32. **Шапаренко Владислав Сергеевич** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата