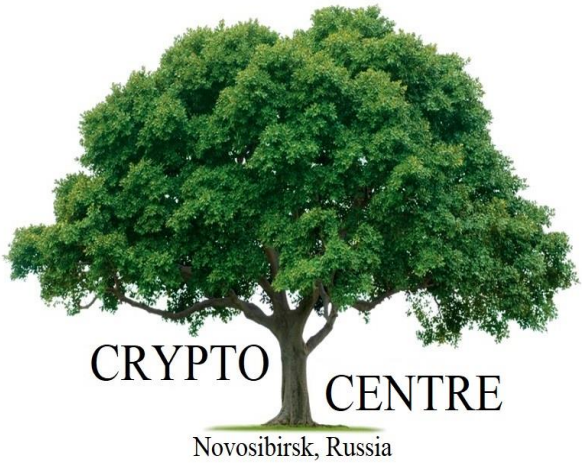
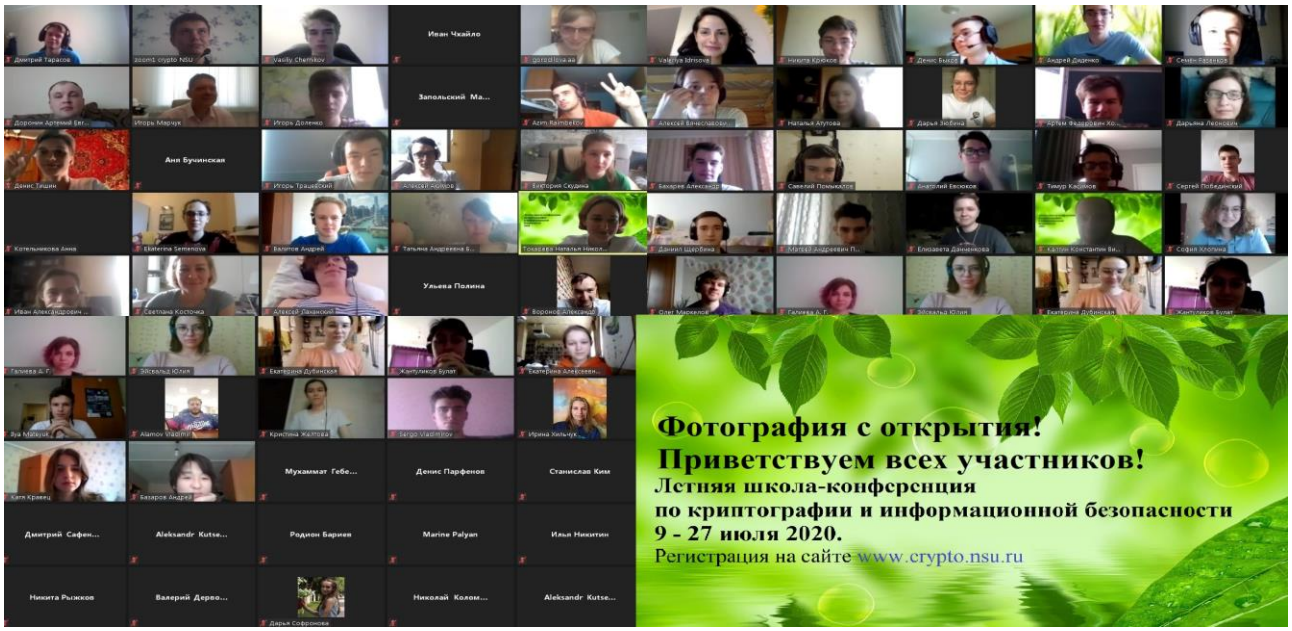


Летняя школа-конференция «Криптография и информационная безопасность» 2020

СБОРНИК ТЕЗИСОВ

Новосибирск 2020

Оглавление



Летняя школа-конференция «Криптография и информационная безопасность» — традиционное мероприятие, проходящее в стенах НГУ каждый год. Организаторами школы-конференции выступают Криптографический центр (Новосибирск), лаборатория криптографии JetBrains Research, Международный математический Центр в Академгородке, организаторы международной олимпиады NSUCRYPTO, Факультет информационных технологий и Механико-математический факультет. Основатель летней школы по криптографии - Сергей Федорович Кренделев.

Участие в школе-конференции принимали студенты, выпускники школ и школьники 11 классов. Школа проходила в дистанционном формате.

В течение трех недель со студентами работали около 15 преподавателей. Студенты принимали участие в лекциях, командной и индивидуальной работе в проектах, связанной с решением исследовательских задач в области криптографии и информационной безопасности, в спортивных занятиях. Одно из важнейших событий школы-конференции – круглый стол по современным проблемам криптографии. Темы проектов были 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. В 2020 году в рамках летней школы-конференции со студентами работали преподаватели из России, Европы и США, в том числе авторы международных стандартов в области криптографии. Часть школы-конференции проходила на английском языке.

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

Лекторы и преподаватели школы:

- Nicky Mouha (USA) - PhD, научный сотрудник отдела компьютерной безопасности Национального института стандартов и технологий США (NIST);
- 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);
- Городилова Анастасия Александровна - к.ф.-м.н., старший преподаватель кафедры теоретической кибернетики ММФ НГУ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Калгин Константин Викторович - к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Колегов Денис Николаевич - к.т.н., доцент кафедры компьютерной безопасности ТГУ, главный разработчик облачной платформы кибербезопасности компании Bi.Zone (Томск);
- Коломеец Николай Александрович - к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, кафедры параллельного программирования ФИТ НГУ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Кондырев Дмитрий Олегович - аспирант ФИТ НГУ, ассистент кафедры систем информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Куценко Александр Владимирович - аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Малыгина Екатерина Сергеевна - к.ф.-м.н., доцент Балтийского федерального университета им. Иммануила Канта (Калининград);
- Николаев Антон Анатольевич - студент кафедры компьютерной безопасности ТГУ, разработчик сервисов анализа защищенности Bi.Zone, главный разработчик фрэймворка Grinder (Томск);
- Облаухов Алексей Константинович - аспирант ИМ СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Пудовкина Марина Александровна - д.ф.-м.н., профессор МГТУ им. Баумана (Москва);
- Сазонова Полина Андреевна - аспирантка ФИТ НГУ, ассистент кафедры общей информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Токарева Наталья Николаевна - к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research.
- Завалишина Елена Владимировна - магистрантка ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;

Организационный комитет:

- Куценко Александр Владимирович - аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Коломеец Николай Александрович - к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, кафедры параллельного программирования ФИТ НГУ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Косточка Светлана Владимировна - м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Идрисова Валерия Александровна - к.ф.-м.н., н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Белоусова Алина Александровна – м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Зюбина Дарья Александровна - студентка ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;
- Максимлюк Юлия Павловна - м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research;

Лекции:

1. Криптография: быстрый старт - Токарева Н.Н.
2. Постквантовая криптография: вызов брошен? - Куценко А.В.
3. Nonstop University CRYPTO: как совместить игру и науку в формате олимпиады? - Городилова А.А.
4. Криптография и криптоанализ с открытым ключом - Токарева Н.Н.
5. Практические аспекты компьютерной безопасности - Колегов Д.Н.
6. On proving security against differential cryptanalysis 1 - Nicky Mouha
7. On proving security against differential cryptanalysis 2 - Nicky Mouha
8. Основы алгебраического криптоанализа - Куценко А.В.
9. Основы технологии блокчейн - Сазонова П.А.
10. ARX-based cryptography 1 - Nicky Mouha
11. ARX-based cryptography 2 - Nicky Mouha
12. Сетевое сканирование и анализ угроз - Колегов Д.Н., Николаев А.А.
13. Методы и техники анализа веб-приложений - Колегов Д.Н., Николаев А.А.
14. Блокчейн изнутри - Кондырев Д.О
15. SAT-решатели и их приложения в криптографии - Калгин К.В.
16. Безопасность систем машинного обучения - Колегов Д.Н., Николаев А.А.
17. Введение в поиск пропавших людей с использованием методов и техник разведки на основе открытых источников - Колегов Д.Н., Николаев А.А.
18. Хэш-функции: построение и анализ - Коломеец Н.А.
19. Алгебраический анализ LRX-шифров - Куценко А.В.
20. Методы машинного обучения в системах разведки на основе открытых источников - Колегов Д.Н., Николаев А.А.
21. Разработка инструментов с применением методов разведки на основе открытых источников - Колегов Д.Н., Николаев А.А.
22. Квантовый криптоанализ: первое приближение - Куценко А.В.
23. Как стать специалистом по компьютерной безопасности - Колегов Д.Н.
24. Введение в теорию эллиптических кривых - Малыгина Е.С.
25. Введение в криптографию на эллиптических кривых - Малыгина Е.С.
26. Базисы Грёбнера и алгоритм Бухбергера в криптографии 1 - Агиевич С.В.
27. Базисы Грёбнера и алгоритм Бухбергера в криптографии 2 - Агиевич С.В.
28. Группы подстановок в криптографии 1 - Пудовкина М.А.
29. Группы подстановок в криптографии 2 - Пудовкина М.А.
30. Группы подстановок в криптографии 3 - Пудовкина М.А.
31. Группы подстановок в криптографии 4 - Пудовкина М.А.

Программа конференции 27 июля 2020г, начало в 14:00:

1. Атутова Н.Д., Зюбина Д.А., Разенков С.И. **Построение алгебраического описания шифров Simon и Speck** (кураторы - Агиевич С.В., Куценко А.В.)
2. Леонович Д.А., Маро Е.А., Филиппов С.Д. **Алгебраические атаки на шифры Simon и Speck** (кураторы - Агиевич С.В., Куценко А.В.)
3. Ходзицкий А.Ф., Сергеев А.В. **Регистры сдвига с нелинейной обратной связью** (куратор - Токарева Н.Н.)
4. Желтова К.А. **Поиск и анализ векторных булевых функций с максимальной компонентной алгебраической иммунностью** (кураторы - Токарева Н.Н., Облаухов А.К.)
5. Зюбина Д.А. **Алгебраическая иммунность S-блока построенного на основе булевой функции от малого числа переменных и перестановки** (кураторы - Токарева Н.Н., Облаухов А.К.)
6. Запольский М.М., Хильчук И.С., Чхайло И.Д. **О необходимых условиях сбалансированности S-блока и его компонентной алгебраической иммунности** (кураторы - Токарева Н.Н., Облаухов А.К)
7. Ахтямов Д.А., Бонич Т.А., Жантуликов Б.Ф., Ищукова Е.А., Панферов М.А., Сутормин И.А., Титова К.М. **Разностные характеристики побитового сложения по модулю 2 относительно сложения по модулю 2^n** (кураторы - Nicky Mouha, Коломеец Н.А.)
8. Валитов А.А., Сафенрейтер Д.А., Лаханский А.А. **Интеграция алгоритмов доказательства с нулевым разглашением в смарт-контракты Ethereum** (куратор - Д.О.Кондырев)
9. Аламов В.А., Быков Д.А. **Разработка смарт-контракта для сервиса купли-продажи ипотечных закладных** (куратор - Сазонова П.А.)
10. Матеюк И.А., Базаров А.А. **Внедрение протокола доказательства с нулевым разглашением для сервиса купли-продажи ипотечной закладной** (куратор - Сазонова П.А.)
11. Щербина Д.А., Раимбеков А.Р. **Разработка веб-приложения для сервиса купли-продажи ипотечных закладных** (куратор - Сазонова П.А.)
12. Побединский С.Ю. **Разработка SAT-решателя с нуля** (куратор - Калгин К.В.)
13. Ким С.Е. **Применение ROBDD для решения криптографических задач** (куратор - Калгин К.В.)Разработка фреймворка для поиска информации на основе открытых источников
14. Софронова Д.А. **Применение SAT-решателей для криптоанализа потоковых шифров** (куратор - Калгин К.В.)
15. Доронин А.Е. **Решение криптографических задач с помощью SAT-решателей** (куратор - Калгин К.В.)
16. Евсюков А.П., Скудина В.В., Карнаухова В.О., Ляпич Н.С., Эйсвальд Ю.И., Котельникова А.А., Помыкалов С.В., Диденко А.А. **Разработка секретного чата TGmini** (Куратор - Завалишина Е.В.)

17. Натарова К.В. **Оптимизация параметров алгоритма Полларда p-1** (кураторы - Токарева Н.Н., Облаухов А.К.)
18. Завалишина Е.В., Парфёнов Д.Р. **Анализ базовой версии криптосистемы с открытым ключом, основанной на сложности решения систем уравнений в конечных полях** (куратор - Токарева Н.Н.)
19. Бахарев А.О., Горяйнова А.П. **Постквантовая криптография: NTRU** (куратор - Куценко А.В.)
20. Семенова Е.В., Трацевский И.Д. **Постквантовая криптография: NTS-KEM** (куратор - Куценко А.В.)
21. Палян М.Г., Кравец Е.А., Сергеев М.И. **Разработка генераторов и мутаторов данных для поиска информации на основе открытых источников** (Кураторы - Колегов Д.Н., Николаев А.А.)
22. Дубинская Е.К., Хлопина С.С., Маркелов О.С., Данченкова Е.Р. **Разработка модулей для сбора информации о людях из открытых источников** (Кураторы - Колегов Д.Н., Николаев А.А.)
23. Крюков Н.Д., Касимов Т.Р., Проскурников Н.А., Черников В.В., Никифоров В.С., Шапоренко А.С. **Разработка фреймворка для поиска информации на основе открытых источников** (Кураторы - Колегов Д.Н., Николаев А.А.)

S-БЛОКИ: ПОСТРОЕНИЕ И СВОЙСТВА

Поиск и анализ векторных булевых функций с максимальной компонентной алгебраической иммунностью

К.А. Желтова

Сибирский государственный университет науки и технологий им. академика М. Ф. Решетнева

E-mail: **masterkristall@gmail.com**

Исследуются методы нахождения булевых функций с максимальной алгебраической иммунностью, векторных булевых функций с максимальной компонентной иммунностью вида $F_\pi(x) = (f(x), f(\pi(x)), f(\pi^2(x)), \ldots, f(\pi^{n-1}(x)))$. Реализован генетический алгоритм для их генерации. Рассмотрено графическое представление таких функций, выявлены ограничения на слои булева куба при малых n.

Ключевые слова: S-блоки, векторная булева функция, генетический алгоритм, алгебраическая иммунность, компонентная алгебраическая иммунность, булев куб.

ВВЕДЕНИЕ

S-блоки – это основные нелинейные преобразования симметричных шифров. В общем случае S-блоки представимы в виде векторных булевых функций $F\colon \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^m$. Векторную булеву функцию F от n переменных всегда можно представить в виде $F(x) = (f^1(x), \ldots, f^m(x))$, где f^j — обычная булева функция от n переменных [1].

При проектировании S-блоков особое внимание уделяется их свойствам, от которых существенно зависит стойкость к криптоанализу. В числе таких свойств можно выделить компонентную алгебраическую иммунность.

Алгебраической иммунностью булевой функции f называется минимальное число d такое, что существует булева функция g степени d, не тождественно равная нулю, для которой выполняется равенство $fg = 0$ или $(f \oplus 1)g = 0$ [1].

Компонентной алгебраической иммунностью AIcomp(F) векторной булевой функции называется минимальная алгебраическая иммунность компонентных функций $b \cdot F (b \in \mathbb{Z}_2^m,\ b \neq 0)$, т. е. $Alcomp(F) = min\{AI(b \cdot F) : b \in \mathbb{Z}_2^m,\ b \neq 0\}$, где $b \cdot F = b_1 f_1 \oplus \ldots \oplus b_m f_m$ [2].

В данной работе рассматриваются векторные булевы функции, имеющие вид

$$F_\pi(x) = (f(x), f(\pi(x)), f(\pi^2(x)), \ldots, f(\pi^{n-1}(x))) \qquad \textbf{(1)}$$

где π – циклический сдвиг влево.

Данный вид функции был предложен участником олимпиады NSUCRYPTO 2016 Алексеем Удовенко в решении задачи о построении векторных булевых функций с максимальной компонентной алгебраической иммунностью. Также Алексей Удовенко получил примеры векторных булевых функций $F\colon \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^m$ при следующих (n, m): (3, 3), (4, 5), (5, 4), (5, 5), (6, 6), (6, 8), (7, 3), (8, 8), (9, 2), (10, 10) [3].

Теорема 1 (см. [4]). Пусть f — булева функция с максимальной алгебраической иммунностью от нечётного числа переменных n, A — невырожденная матрица $n \times n$. Тогда функция $g(x) = f(x) \oplus f(Ax)$ обладает максимальной алгебраической иммунностью только в том случае, если под действием линейного преобразования A ровно половина элементов множества $supp(f)$ остаётся в этом множестве, где $supp(f)$ — носитель функции f.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ПОИСКА БУЛЕВЫХ ФУНКЦИЙ С ЗАДАННЫМИ СВОЙСТВАМИ

При росте числа переменных n множество булевых функций от n переменных растет дважды экспоненциально и полный перебор таких функций становится невозможным, поэтому выделяют 3 способа получения булевой функции, пригодной для криптографических целей: случайная генерация, алгебраическое конструирование и эвристики. При увеличении n подход случайной генерации значительно теряет в эффективности. Алгебраические построения сразу обеспечивают определенные свойства функций, но заведомо сужают множество решений. Однако в области эвристик появились и появляются плодотворные идеи и перспективные направления исследования [5]. В частности, широкое применение нашли генетические алгоритмы, генетическое программирование, эволюционное программирование [6].

Генетический алгоритм (ГА) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию [7].

В контексте данной задачи особями являлись вектора значений булевых функций от n переменных. В качестве оператора мутации была выбрана инверсия случайного бита. Оператор скрещивания – однородный кроссинговер.

Функции приспособленности задавались как:

$F_{fit1} = |target\_weight - weight| + (max\_ai - ai)$, где $target\_weight$ – вес целевой булевой функции = $\frac{n}{2}$, $weight$ – текущий вес булевой функции, max_ai – максимальная алгебраическая иммунность для функции от n переменных и ai – текущая алгебраическая иммунность.

Данная функция позволяет находить сбалансированные булевы функции от n переменных, обладающие максимальной алгебраической иммунностью.

n	# особей в популяции	# поколений	Вероятность мутации	Вероятность скрещивания	# уникальных функций с max AI на всех итерациях	Время работы
6	100	25	0.1	0.5	287	19.4 с.
7	100	25	0.2	0.5	103	51 с.
8	100	25	0.1	0.5	271	2 мин. 14 с.
9	100	25	0.2	0.5	136	10 мин. 54

						с.
10	100	20	0.1	0.5	174	47 мин. 7 с.
11	100	10	0.3	0.5	21	54 мин. 4 с.

Таблица 1 – параметры и результаты работы ГА с функцией приспособленности F_{fit1}.

$F_{fit2} = |target\_weight - weight| + |supp\_sum * 2 - target\_weight| + (max\_ai - ai)$, где $target\_weight$ – вес целевой булевой функции = $\frac{n}{2}$, $weight$ – текущий вес булевой функции, max_ai – максимальная алгебраическая иммунность для функции от n переменных, ai – текущая алгебраическая иммунность, $supp\_sum$ – мощность пересечения носителей функций $f(x), f(\pi(x))$.

Данная функция позволяет находить сбалансированные булевы функции от n переменных, обладающие максимальной алгебраической иммунностью, для которых выполняется теорема 1.

n	# особей в популяции	# поколений	Вероятность мутации	Вероятность скрещивания	# уникальных функций с max AI на всех итерациях	Время работы
7	100	25	0.2	0.5	28	59 с.
9	100	25	0.2	0.5	23	6 мин. 56 с.
11	100	25	0.3	0.5	2	48 мин. 18 с.

Таблица 2 – параметры и результаты работы ГА с функцией приспособленности F_{fit2}.

$F_{fit3} = max\_comp\_ai - comp\_ai$, где max_comp_ai – необходимая максимальная компонентная иммунность, $comp\_ai$ – имеющаяся компонентная иммунность.

Данная функция позволяет находить булевы функции от четного числа переменных, допускающие построение векторных булевых функций с максимальной компонентной алгебраической иммунностью.

n	# особей в популяции	# поколений	Вероятность мутации	Вероятность скрещивания	# полученных функций с max AIcomp
6	1000	10	0.5	0.4	111
8	1000	1	0.5	0.4	59
10	10	1	0.5	0.4	4

Таблица 3 – параметры и результаты работы ГА с функцией приспособленности F_{fit3}.

СУММЫ НА СЛОЯХ БУЛЕВА КУБА

Булев куб – это геометрическая интерпретация области определения булевой функции [8].

Каждый уровень представляющий булев куб $\mathbb{B}^n$ состоит из всех вершин, соответствующих наборам, у которых ровно k $(0 \leq k \leq n)$ компонент отличны от нуля (множество всех таких наборов для фиксированного k называют k-слоем булева куба.

Существует 12 булевых функций от 3-х переменных (**n = 3**), допускающих построение векторной функции с максимальной компонентной алгебраической иммунностью по конструкции (1). Все такие функции имеют одинаковое распределение сумм по слоям булева куба (таблица 4). Графическое представление функций в виде булева куба также совпадает с точностью до перенумерования вершин (рис. 1).

Вектор значений	АНФ	0-слой	1-слой	2-слой	3-слой
00101101	X1 ⊕ X2 ⊕ X2X3	0	2	1	1
00111001	X1 ⊕ X2 ⊕ X1X3	0	2	1	1
01001011	X1 ⊕ X3 ⊕ X2X3	0	2	1	1
01011001	X1 ⊕ X3 ⊕ X1X2	0	2	1	1
01100011	X2 ⊕ X3 ⊕ X1X3	0	2	1	1
01100101	X2 ⊕ X3 ⊕ X1X2	0	2	1	1
10011010	X1 ⊕ X2 ⊕ X2X3 ⊕ 1	1	1	2	0
10011100	X1 ⊕ X2 ⊕ X1X3 ⊕ 1	1	1	2	0
10100110	X1 ⊕ X3 ⊕ X2X3 ⊕ 1	1	1	2	0
10110100	X1 ⊕ X3 ⊕ X1X2 ⊕ 1	1	1	2	0
11000110	X2 ⊕ X3 ⊕ X1X3 ⊕ 1	1	1	2	0
11010010	X2 ⊕ X3 ⊕ X1X2 ⊕ 1	1	1	2	0

Таблица 4 – булевы функций от 3-х переменных, допускающие построение векторной функции с максимальной компонентной алгебраической иммунностью.

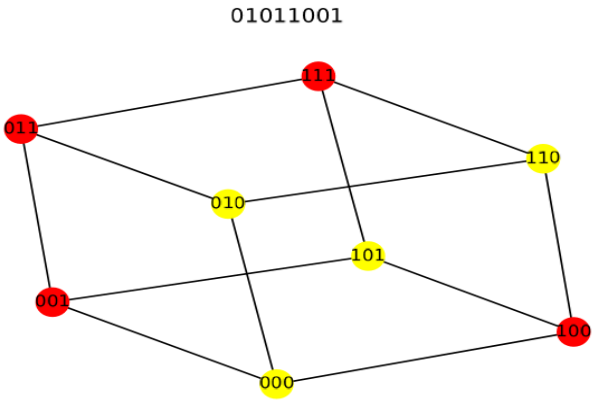


Рисунок 1 – Пример булева куба $\mathbb{B}^3$, соответствующего булевой функции, допускающей построение векторной с максимальной компонентной иммунностью.

Для **n = 4** существует 6144 булевых функции, допускающих построение векторной функции с максимальной алгебраической иммунностью по конструкции (1). Число различных комбинаций сумм по слоям булева куба составляет 76.

Данные комбинации удовлетворяют следующим ограничениям:

1. $sum(1) \neq sum(n-1)$
2. $0 < sum(1),\ sum(n-1) < n$
3. $\forall\, i \in \{1..n-1\}$: $sum(i) \neq 0$, где sum(i) – сумма на i-слое булева куба.

Из них можно выделить подмножество комбинаций слоев, характеризующихся следующими признаками:

1. Сумма значений на 0-слое и n-слое равна 1.
2. Вес функций равен $\frac{n}{2}$, т.е. функции являются сбалансированными.
3. $|sum(1) - sum(n-1)| = 1$

Число таких комбинаций равно 8 (таблица 5).

0-слой	1-слой	2-слой	3-слой	4-слой
0	1	4	2	1
0	2	2	3	1
0	2	4	1	1
0	3	2	2	1
1	2	4	1	0
1	3	2	2	0
1	1	4	2	0
1	2	2	3	0

Таблица 5 – комбинации слоев булевых функций от 4 переменных.

Графическое представление функций с суммами по слоям вида (0, 1, 4, 2, 1) представлено на рисунке 2.

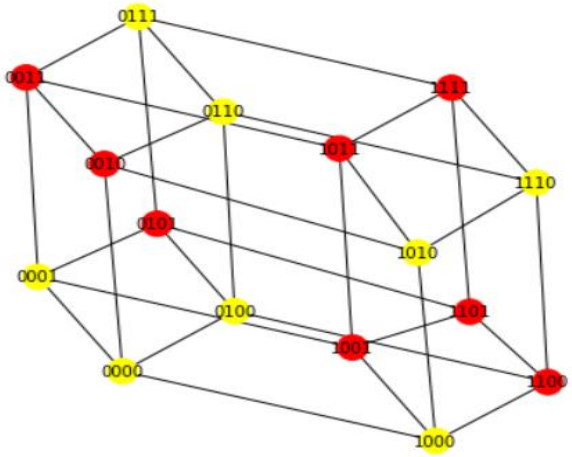


Рисунок 2 – Булев куб функции от 4-х переменных, допускающей построение векторной с максимальной компонентной иммунностью.

Используя приведенные выше ограничения, можно непосредственно сконструировать комбинации сумм слоев булева куба, для последующего перебора всех булевых функций, удовлетворяющих этим комбинациям.

Для **n = 5** путем полного перебора удалось установить, что число функций, для которых выполняется теорема 1, составляет 170.156.250, из них не менее 22.232.091 обладают максимальной алгебраической иммунностью, в том числе не менее 2177 имеют максимальную компонентную иммунность при построении векторной булевой функции по конструкции (1).

Также было выявлено, что часть ограничений функций от четырех переменных сохраняется, а именно:

1. $0 < sum(1), sum(n-1) < n$
2. $\forall i \in 1..n-1: sum(i) \neq 0$
3. $sum(1) \neq sum(n-1)$

ПЛАНЫ НА БУДУЩЕЕ

1. Использовать в качестве особи генетического алгоритма не вектор значений булевой функции, а непосредственно векторную булеву функцию, что позволит обобщить результаты, выйдя за пределы функций вида (1).

2. Рассмотреть различные комбинации эвристических методов, например, генетические алгоритмы в сочетании с алгоритмом имитации отжига.

3. Улучшить конструкцию функции приспособленности F_{fit3} для увеличения скорости работы генетического алгоритма, в частности рассмотреть конструкции булевых кубов и сформулировать множество ограничений, которым удовлетворяют векторные булевы функции с максимальной компонентной иммунностью.

4. Определить способ перехода от функций с максимальной AI от n переменных к функциям с максимальной AI от n + 1 переменной.

5. Применить методы машинного обучения без учителя для поиска скрытых закономерностей в булевых функциях с максимальной алгебраической иммунностью и максимальной компонентной иммунностью.

ЛИТЕРАТУРА

1. Токарева Н. Н. Симметричная криптография. Краткий курс. Новосибирск, 2012.
2. Carlet C. On the algebraic immunities and higher order nonlinearities of vectorial Boolean Functions. Proc. NATO Advanced Research Workshop ACPTECC, Veliko Tarnovo, Bulgaria, October 6–9, 2008, Amsterdam, IOS Press, 2009, С. 104–116.
3. Tokareva N., Gorodilova A., Agievich S., et al. Mathematical methods in solutions of the problems from the Third International Students' Olympiad in Cryptography // Прикладная дискретная математика. 2018. №40. С. 34-58.
4. Милосердов А. В. Конструкции векторных булевых функций с максимальной компонентной алгебраической иммунностью // Прикладная дискретная математика. Приложение. 2018. С.47-48.
5. Агафонова И. В., Дмитриева О. М. Ограниченный поиск криптографически сильных булевых функций // Компьютерные инструменты в образовании. 2017 №3: С.20–28.
6. L. Mariot, D. Jakobovic, A. Leporati, S. Picek Hyper-bent Boolean Functions and Evolutionary Algorithms // Theory and Applications of Models of Computation. 2019 C.262-277.
7. Генетический алгоритм [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=Генетический_алгоритм (дата обращения: 23.07.2020).
8. Канатников А. Н. Дискретная математика. Конспект лекций. Москва, 2006.

Кураторы исследования:

— Токарева Наталья Николаевна, к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research Токарева Наталья Николаевна;

— Облаухов Алексей Константинови, аспирант ИМ СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research Облаухов Алексей Константинович;

О необходимых условиях сбалансированности S-блока и его компонентной алгебраической иммунности

М.М. Запольский[1], И.С. Хильчук[1], И.Д. Чхайло[2]

[1] Новосибирский государственный университет
[2] Тюменский государственный университет

E-mail: **i.khilchuk@g.nsu.ru, m.zapolskii@g.nsu.ru, stud0000005351@study.utmn.ru**

Рассмотрена конструкция векторной функции на основе булевой функции и перестановки, получен критерий сбалансированности и, для нечетного числа переменных, необходимое условие максимальной алгебраической иммунности. Описан способ добавления переменных в булеву функцию, не уменьшающие алгебраическую иммунность.

Ключевые слова: S-блок, векторная булева функция, булева функция, алгебраическая иммунность, компонентная алгебраическая иммунность

S-блоки являются основной нелинейной частью симметричных шифров и обеспечивают их стойкость к различным видам криптоанализа, поэтому подходить к их построению необходимо очень тщательно. С математической точки зрения S-блок является векторной булевой функцией, осуществляющей отображение из множества двоичных векторов длины n в множество двоичных векторов длины m. Известно, что высокая алгебраическая иммунность векторной булевой функции позволяет шифру противостоять алгебраическому криптоанализу. Заметим также, что чаще всего в качестве S-блоков используются взаимно однозначные векторные булевы функции, что обусловлено необходимостью однозначного расшифрования, в таком случае $n = m$. В данной работе мы рассмотрели конструкцию векторной булевой функции с использованием булевой функции и перестановки и нашли критерий сбалансированности и, для нечетного числа переменных, необходимое условие максимальной алгебраической иммунности.

Рассмотрим следующую конструкцию векторной булевой функции:

$$F_\pi(x) = (f_1(x), f_2(x), \dots, f_n(x)),$$

где $f_i(x) = f(\pi^{i-1}(x)), i = 1, \dots, n$, π – сдвиг влево на единицу.

Алгебраической иммунностью булевой функции f(x) от n переменных называется минимальное число d такое, что существует не тождественно равная нулю булева функция g(x) от n переменных степени d, для которой выполняется $f(x)g(x) = 0$ или $(f(x) + 1)g(x) = 0$. Известна оценка сверху для алгебраической иммунности функции от n переменных: $AI(f) \leqslant \lceil \frac{n}{2} \rceil$.

Компонентной алгебраической иммунностью векторной булевой функции называется минимальная алгебраическая иммунность её компонентных функций.

Векторная булева функция сбалансирована, если принимает каждое значение 2^{n-m}раз. Векторная булева функция является сбалансированной тогда и только тогда, когда любая ее компонентная функция сбалансирована.Если $m = n$, то сбалансированная функция является взаимно однозначной.

Выясним, при каких условиях компонентная алгебраическая иммунность векторной функции F будет максимальной, если максимальной была алгебраическая иммунность

булевой функции f.

Известно, что если f – булева функция от нечетного числа переменных n и имеет максимальную алгебраическую иммунность, то она сбалансирована. Таким образом, нам необходимо, чтобы каждая компонентная функция для F была сбалансирована. Введем разбиение множества F_2^n на орбиты под действием циклического сдвига . Будем обозначать через C_i^k орбиту относительно π длины k под номером i, $i \in 1 \dots i_k$, где i_k– число различных орбит длины k . Для $x \in F_2^n$ через $O(x)$ будем обозначать орбиту, порожденную вектором x.

Утверждение 1. Длина орбиты C_i^k является делителем числа n.

Док-во: Следует из свойства циклического сдвига $\pi^n = id$.■

Отметим несколько замечаний которые будут необходимы в дальнейшем.

- Число орбит длины k не зависит от n при $k \leq n$;
- Существует только две орбиты длины 1;
- Если n простое число, то его разбиение на орбиты состоит из двух орбит длины 1 и $\frac{2^n-2}{n}$ орбит длины n.

Утверждение 2. Для функций F_π выполняется: $F_\pi\big(\pi(x)\big) = \pi\big(F_\pi(x)\big)$.

Док-во: Следует прямо из определения функции F_π, а также свойства $\pi^n = id$. ■

Таким образом функция F_π однозначно определяется по ее значению хотя бы на одном элементе каждой орбиты. Теперь можно привести конструкцию:

Конструкция 1. Для каждого k выберем перестановку на множестве орбит одинаковой длины π_{i_k}. Определим $F(x) = y$ таким образом, чтобы $\pi_{i_k}\big(O(x)\big) = O(y)$, где $k = |O(x)|$. Тогда мы определим функцию $F(x)$ для любого $x \in F_2^n$ которая являются взаимно однозначной и получена конструкцией F_π.

Утверждение 3. Функции, полученные по конструкции 1, являются взаимно однозначными функциями вида F_π.

Док-во: То что функции получаются из конструкции F_π при помощи некоторой функции fпроверяется непосредственно, мы легко можем построить покоординатно скалярную Булеву функцию f, которая будет соответствовать функции полученной по конструкции 1. Ее взаимная однозначность следует из работы [1]. Легко видеть, что построенная нами функция попадает под условие теоремы.■

Далее будем рассматривать случай нечетного n. Поскольку необходимым условием максимальности компонентной алгебраической иммунности векторной функции F является ее сбалансированность, мы можем утверждать, что функции с максимальной компонентной алгебраической иммунностью, полученные конструкцией F_π необходимо получаются из конструкции 1.

Представлением булевой функции через таблицу истинности называется вектор $f \in F_2^{2^n}$такой, что значение функции f на двоичном представлении числа i является его i-ая координата. i-ым битом будем обозначать значение функции f на i-ой координате. Мы будем представлять таблицу истинности в виде разбиения бинарных векторов на орбиты. Например для $n = 3$ таким разбиением будет (0)(1 2 4)(3 5 6)(7). В общем случае будем обозначать такое разбиение как $(C_1^1)(C_2^1) \dots \big(C_i^k\big) \dots \big(C_{i_n}^n\big)$. Будем через $vec_f\big(C_i^k\big)$ обозначать вектор полученный последовательной “склейкой” координат таблицы истинности f внутри орбиты и дополнять его до вектора длины n дублированием. Например для n=3 (0) →000; (1,0,0)→100 и т.д.

Утверждение 4. Функция F_π является взаимно однозначной тогда и только тогда, когда порождающая ее функция f удовлетворяет следующему свойству. Для любого k векторы $vec(C_i^k)$ для разных i лежат в разных орбитах и длины $O(vec_f(C_i^k))$ равны k.

Док-во: Непосредственно проверяется, что функции F_π, полученные из таких функций f будут удовлетворять конструкции 1, а также первая координата функций из конструкции 1 удовлетворяет условию из утверждения 4.■

Отметим ряд замечаний:

- Если мы имеем таблицу истинности функции f, которая дает взаимно однозначную F_π, тогда при циклическом сдвиге таблицы истинности внутри каждой орбиты C_i^k или при замене значений двух орбит одинаковой длины друг с другом функция F_π останется взаимно однозначной.
- Если функция F_π является взаимно однозначной, тогда сумма 0-го и 2^n-1-го бита функции f равна единице.

Утверждение 5. Если функция F_π от нечетного числа переменных имеет максимальную компонентную алгебраическую иммунность, то функция f удовлетворяет свойству из утверждения 4.

Таким образом, мы сформулировали необходимое условие того, что векторная булева функция, полученная предложенной конструкцией, имеет максимальную компонентную алгебраическую иммунность. Далее приведем некоторые частные случаи и сформулируем условия, которые проще проверять на практике.

Случай n = 3.

Перестановка вектора значений состоит из циклов (0)(1 2 4)(3 5 6)(7). Необходимым условием максимальной компонентной алгебраической иммунности будет являться:

1. сумма крайних битов = 1
2. отсутствуют циклы длины три, состоящие из единиц.

Случай n = 5.

Перестановка вектора значений состоит из циклов (0)(1 2 4 8 16)(3 6 12 24 17)(5 10 20 9 18)(7 14 28 25 29)(11 22 13 26 21)(15 30 29 27 23)(31). Необходимым условием максимальности компонентной алгебраической иммунности будет являться:

1. сумма крайних битов = 1
2. нет циклов с пятью единицами и пятью нулями.
3. существует один цикл длины пять с одной единицей, два с двумя единицами, два с тремя и один с четырьмя.

Утверждение 6. Булевы функции с максимальной алгебраической иммунностью от простого числа переменных позволяют строить векторные функции с максимальной компонентной иммунностью в случае, когда

1. сумма крайних битов вектора значений = 1,
2. отсутствуют циклы из n единиц и n нулей.

Док-во: Следует из утверждения 5, циклы из n единиц и n нулей соответствуют орбитам длины 1, а они могут быть только крайними битами, длина цикла которых равна единице.■

Известной конструкцией булевых функций с максимальной алгебраической иммунностью является конструкция Далая [2], которая определяется следующим образом

Для нечетного n:

$$f(x) = \begin{cases} 0, & wt(x) \le \left\lfloor \frac{n}{2} \right\rfloor \\ 1, & wt(x) \ge \left\lceil \frac{n}{2} \right\rceil \end{cases}$$

Для четного n:

$$f(x) = \begin{cases} 0, & \mathrm{wt}(x) < \frac{n}{2} \\ g(x), & wt(x) = \frac{n}{2} \\ 1, & wt(x) > \frac{n}{2} \end{cases}$$

Где $g(x)$ принимает произвольные значения. Для нечетного n поскольку циклический сдвиг не меняет вес вектора имеем, что конструкция Далая противоречит необходимому условию из утверждения 6. Для четного числа n мы можем сложить две компоненты векторной функции и получить функцию, равную нулю всюду, кроме $\frac{n}{2}$ слоя, далее можно сделать оценку на число аннигиляторов такой функции и понять, что всегда будут присутствовать аннигиляторы степени меньше, чем $\frac{n}{2}$. Таким образом, конструкция Далая не позволяет строить векторные булевы функции с максимальной алгебраической иммунностью при помощи приведенной в работе конструкции.

Утверждение 7. Количество орбит, длина которых больше единицы, для нечетного n – четно.

Док-во: Рассмотрим любую орбиту подстановки вектора значений для нечетного n. Каждому элементу орбиты соответствует некоторый набор значений переменных. Заметим, что в любом наборе значений переменных в орбите, количество переменных, значение которых равно нулю, не может совпадать с количеством переменных, значение которых равно единице ввиду нечетности суммы их количества. Тогда, инвертируя значения всех переменных для каждого элемента орбиты, мы получим новую орбиту, которая не совпадает с исходной, потому как в ней отличается количество единиц и нулей. Таким образом, каждая орбита имеет в свою пару еще одну, с которой не совпадает, а пары орбит не пересекаются между собой. Значит количество орбит четно.■

В случае четного количества переменных, такая сопряженная орбита может совпасть с исходной, например, в случае, когда значения переменных чередуются. В таком случае мы еще раз получаем уже описанный выше факт – при четном количестве переменных всегда существует орбита из двух элементов.

Может оказаться полезным изучить другие методы преобразований одних функций в другие, которые не уменьшают алгебраическую иммунность. В частности, это позволило бы переходить к функциям с большим числом переменных с сохранением свойства максимальной алгебраической иммунности.

Утверждение 8. Пусть $f(x_1, \dots, x_n)$ – некоторая булева функция, и $AI(f) = d$. Рассмотрим функцию $h(x_1, \dots, x_n, x_{n+1}) = f(x_1, \dots, x_n) + x_{n+1}$, где x_{n+1}- новая независимая переменная. Тогда $AI(h) \ge d$.

Док-во: Докажем, что не существует аннулятора функции $h(x_1, \dots, x_n, x_{n+1})$ степени меньше, чем d. Предположим, что такой аннулятор существует.

Пусть $g(x_1, \dots, x_n, x_{n+1})$ – этот аннулятор и $hg = 0$. По определению $g(x_1, \dots, x_n, x_{n+1}) \not\equiv 0$. Заметим, что он существенно зависит от переменной x_{n+1}, потому как в противном случае

при $x_{n+1} = 1$ получили бы $g(x_1, \dots, x_n, 1) \equiv g(x_1, \dots, x_n, 1)f(x_1, \dots, x_n)$, то есть $g(x_1, \dots, x_n, 1)$ был бы аннулятором $f(x_1, \dots, x_n)$ степени, меньше чем d, что противоречит условию.

Имеем, что $g(x_1, \dots, x_n, x_{n+1})x_{n+1} = g(x_1, \dots, x_n, x_{n+1})f(x_1, \dots, x_n)$. По определению $g(x_1, \dots, x_n, x_{n+1}) \not\equiv 0$. Заметим, что при $x_{n+1} = 0$ должно выполняться $g(x_1, \dots, x_n, 0)f(x_1, \dots, x_n) \equiv 0$ для любых значений $(x_1, \dots, x_n)$. Тогда $g(x_1, \dots, x_n, 0)$ – аннулятор исходной функции $f(x_1, \dots, x_n)$, и его степень не меньше, чем d, значит в его АНФ содержатся мономы степени d. Тогда степень $g(x_1, \dots, x_n, x_{n+1})$ не меньше, потому как в его АНФ содержаться как минимум все те же самые мономы степени d. Также это возможно, когда $g(x_1, \dots, x_n, 0) = 0$, тогда $g(x_1, \dots, x_n, x_{n+1}) = x_{n+1}g'(x_1, \dots, x_n)$. Но в таком случае $x_{n+1}g'(x_1, \dots, x_n) = x_{n+1}g'(x_1, \dots, x_n)f(x_1, \dots, x_n)$. Тогда $g'(x_1, \dots, x_n) = g'(x_1, \dots, x_n)f(x_1, \dots, x_n)$. В таком случае $g'(x_1, \dots, x_n)$ – аннулятор $f(x_1, \dots, x_n)$ степени, меньше чем d. Получили противоречие.

В случае, если $(h + 1)g = 0$, имеем, что $g(x_1, \dots, x_n, x_{n+1})x_{n+1} = g(x_1, \dots, x_n, x_{n+1})f(x_1, \dots, x_n) + g(x_1, \dots, x_n, x_{n+1})$. Как и в первом случае, $g(\mathrm{x}_1, \dots, x_n, x_{n+1})$ существенно зависит от x_{n+1}. Заметим, что при $x_{n+1} = 0$ должно выполняться $g(x_1, \dots, x_n, 0)(f(x_1, \dots, x_n) + 1) \equiv 0$ – противоречие, аналогичное полученному в первом случае. ■

Утверждение 9. Пусть $f(x_1, \dots, x_n)$ – некоторая булева функция, n - нечетно, и пусть $f(x_1, \dots, x_n)$ имеет максимальную алгебраическую иммунность. Тогда функция $h(x_1, \dots, x_n, x_{n+1}) = f(x_1, \dots, x_n) + x_{n+1}$, где x_{n+1}- новая независимая переменная, также имеет максимальную алгебраическую иммунность.

Док-во: То, что $AI(h) \geq AI(f)$, следует из утверждения 8. При этом $AI(h) \leq AI(f)$, потому как $AI(f) = \left\lceil \frac{n}{2} \right\rceil = \frac{n+1}{2}$. Значит $AI(h) = AI(f)$. ■

ЛИТЕРАТУРА

1. M. M. Zapolskiy, N. N. Tokareva On one-to-one property of a vectorial boolean function of the special type // Труды конференции SIBECRYPT 2020.
2. Deepak Kumar Dalai, Subhamoy Maitra and Sumanta Sarkar Basic Theory in Construction of Boolean Functions with Maximum Possible Annihilator Immunity// Des Codes Crypt 40, 41–58 (2006).

Кураторы исследования:

— Токарева Наталья Николаевна, к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research Токарева Наталья Николаевна;

— Облаухов Алексей Константинови, аспирант ИМ СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research Облаухов Алексей Константинович;

Алгебраическая иммунность S-блока построенного на основе булевой функции от малого числа переменных и перестановки

Д. А. Зюбина

Новосибирский государственный университет

E-mail: **d.zyubina@g.nsu.ru**

Пусть π – перестановка n элементов, f- булева функция от n переменных. Представим векторную булеву функцию $F_\pi: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ как $F_\pi(x) = \left(f(x), f(\pi(x)), f(\pi^2(x)), \dots, f(\pi^{n-1}(x))\right)$. В данной работе изучается компонентная алгебраическая иммунность векторной булевой функции F_π в зависимости от булевой функции f и перестановки π при $n = 2, 3, 4, 5$.

Ключевые слова: булева функция, векторная булева функция, алгебраическая иммунность, компонентная алгебраическая иммунность.

S-блоки играют решающую роль в обеспечении стойкости блочных шифров относительно разных типов атак. S-блок – это отображение из множества двоичных векторов длины n в себя. Проектироваться S-блоки должны наиболее тщательно, так как стойкость всего шифра существенно зависит от их криптографических характеристик. Чтобы шифр был более стойким к алгебраическому криптоанализу, булева функция должна принимать наибольшее возможное значение алгебраической иммунности.

В данной работе S-блок представлен в виде векторной булевой функции $F: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ $F_\pi(x) = \left(f(x), f(\pi(x)), f(\pi^2(x)), \dots, f(\pi^{n-1}(x))\right)$, где f – булева функция от n переменных, π – циклический сдвиг влево n элементов. Данная конструкция была предложена Алексеем Удовенко в решении олимпиадной задачи на NSUCRYPTO-2016 [1]. Он показал, что при таком построении векторной функции можно получить функцию с максимальной алгебраической иммунностью.

Алгебраической иммунностью $AI(f)$ булевой функции f называется минимальное число d такое, что существует булева функция g степени d, не тождественно равная нулю, для которой выполняется равенство $fg = 0$ или $(f \oplus 1)g = 0$. Известно, что для произвольной булевой функции f от n переменных выполнено $AI(f) \leq \lceil n/2 \rceil$.

Компонентной алгебраической иммунностью $AI_{comp}(F)$ векторной булевой функции F называется минимальная алгебраическая иммунность компонентных функций $AI_{comp}(F) = min\{AI(b * F): b \in \mathbb{Z}_2^n, b \neq 0\}$, где $b * F = b_1 f_1 \oplus \dots \oplus b_n f_n$ [2].

Также для анализа использовалась следующая теорема:

Теорема [3]. Пусть f – булева функция с максимальной алгебраической иммунностью от нечётного числа переменных n, A – невырожденная матрица $n \times n$. Тогда функция $g(x) = f(x) + f(Ax)$ обладает максимальной алгебраической иммунностью только в том случае, если под действием линейного преобразования A ровно половина элементов множества $supp(f)$ остаётся в этом множестве, где $supp(f)$ – носитель функции f.

Для $\boldsymbol{n = 2}$ путем полного перебора всех булевых функций было получено, что существует 14 булевых функций с максимальной алгебраической иммунностью ($AI = 1$). Далее, путем перебора уже полученных функций было обнаружено, что существует 8 булевых функций, на основе которых построенные векторные булевы функции также имеют максимально

возможную компонентную алгебраическую иммунность.

Для $\boldsymbol{n = 3}$ путем полного перебора всех булевых функций было получено, что существует 56 булевых функций с максимальной алгебраической иммунностью ($AI = 2$). Также путем полного перебора был получен список булевых функций, удовлетворяющих теореме [3], их количество равно 54. При пересечении двух полученных списков было найдено пересечение мощности 42. После этого на основе полученных функций были построены векторные булевы функции. В результате было обнаружено, что существует всего 12 булевых функций, на основе которых построенные векторные булевы функции имеют максимальную иммунность. Также в ходе перебора было сформулировано утверждение:

Утверждение 1. Функции от 3 переменных, удовлетворяющие теореме [3], имеющие максимальную алгебраическую иммунность, и на основе которых построенные векторные булевы функции также имеют максимальную компонентную иммунность, можно описать конструкцией:

$$f(x_1, x_2, x_3) = x_a + x_b + x_a x_c + i, \qquad \text{где } a, b, c \in \{1,2,3\}, a \neq b \neq c, i \in \{0,1\}.$$

Для $\boldsymbol{n = 4}$ путем полного перебора всех булевых функций было получено, что существует 54952 булевых функций с максимальной алгебраической иммунностью ($AI = 2$). Было решено рассмотреть всевозможные перестановки π, а не только циклический сдвиг влево, как это происходило ранее. Для 4 элементов всего существует $4! = 24$ перестановок, но 15 из них оставляют на месте как минимум одну координату, и из-за этого векторная булева функция не будет достигать своего максимума. Поэтому была проведена работа с остальными 9 перестановками. Было получено, что только при 6 перестановках существуют векторные булевы функции, которые сохраняют максимальную иммунность. Эти 6 перестановок являются полноцикловыми перестановками:

$$\pi_1 = (2,3,4,1), \pi_2 = (2,4,1,3), \pi_3 = (3,1,4,2), \pi_4 = (3,4,2,1), \pi_5 = (4,1,2,3), \pi_6 = (4,3,1,2)$$

Затем, путем перебора уже полученных функций (с максимальной алгебраической иммунностью) было обнаружено, что для каждой перестановки существует 6144 булевых функций, на основе которых построенные векторные булевы функции также имеют максимально возможную компонентную алгебраическую иммунность.

Также, для $\boldsymbol{n = 4}$ был взят список всех булевых функций, которые порождают взаимно-однозначные векторные булевы функции (их количество = 1536). В данном случае векторные булевы функции строились на основе циклического сдвига влево. В результате подсчета получилось, что 1520 функций имеют максимальную алгебраическую иммунность. После этого, на основе булевых функций с максимальной иммунностью, были построены векторные булевы функции. Было получено, что всего есть 1024 таких булевых функций на основе которых построенные векторные булевы функции сохраняют максимальную алгебраическую иммунность.

Утверждение 2. Пусть функция f – булева функция от n переменных с максимальной алгебраической иммунностью. Если векторная булева функция F_π, построенная как $F_\pi(x) = \left(f(x), f(\pi(x)), f(\pi^2(x)), \dots, f(\pi^{n-1}(x))\right)$ имеет максимальную компонентную алгебраическую иммунность, то π является полноцикловой перестановкой.

Для $\boldsymbol{n = 5}$ была рассмотрена часть булевых функций, которые порождают взаимно-однозначные векторные булевы функции (1439 функций). Из этих функций только 400 имеют максимальную алгебраическую иммунность, и только 2 порождают векторные функции с максимальной иммунностью.

По итогам проделанной работы были получены полные множества таких булевых функций

с максимальной алгебраической иммунностью от малого количества переменных, что векторная булева функция, построенная на основе булевой функции, также имеет максимальную компонентную алгебраическую иммунность. Также были рассмотрены и получены булевы функции от 4 и 5 переменных, которые порождают взаимно-однозначные векторные булевы функции с максимальной иммунностью.

В дальнейшем планируется провести анализ для большего числа переменных.

ЛИТЕРАТУРА

1. N. Tokareva, A. Gorodilova, S. Agievich, V. Idrisova, N. Kolomeec, A. Kutsenko, A. Oblaukhov, G. Shushuev Mathematical methods in solutions of the problems presented at the third international students' olympiad in cryptography // Прикладная дискретная математика. 2018.С.56–57.
2. Carlet C. On the algebraic immunities and higher order nonlinearities of vectorial Boolean functions // Enhancing Cryptographic Primitives with Techniques from Error Correcting Codes. Amsterdam: IOS Press, 2009. P. 104–116
3. Милосердов А. В. Конструкции векторных булевых функций с максимальной компонентной алгебраической иммунностью // Прикладная дискретная математика. Приложение. 2018.С.47–48.

Кураторы исследования:

— Токарева Наталья Николаевна, к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research Токарева Наталья Николаевна;

— Облаухов Алексей Константинови, аспирант ИМ СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research Облаухов Алексей Константинович;

ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

Регистры сдвига с нелинейной обратной связью

А.Ф. Ходзицкий, А.В. Сергеев

Новосибирский государственный университет

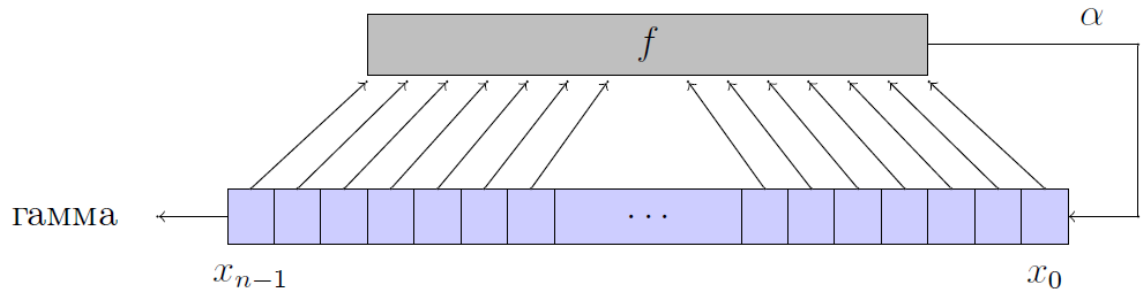
E-mail: **a.khodzitskii@g.nsu.ru, a.sergeev1@g.nsu.ru**

Объектом данного исследования являются регистры сдвига с нелинейной обратной связью, поиск функций, с помощью которых возможно их построение, исследование их свойств и конструкций.

Ключевые слова: регистр сдвига, функция максимального цикла, нелинейная булева функция

Введение

Булева функция – это отображение $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, где $\mathbb{F}_2$ - поле из элементов $\{0,1\}$. Любая f имеет единственное представление в виде полинома $f(x_1, \dots, x_n) = \bigoplus_{k=0}^{n} \bigoplus_{i_{k,\dots,i_k}} a_{i_1,\dots,i_k} x_{i_1,\dots,i_k} \oplus a_0$, который называется алгебраической нормальной формой (АНФ) или полиномом Жегалкина данной функции. Алгебраическая степень f – это максимальное количество переменных в мономе АНФ. Регистры сдвига строятся с помощью булевой функции: имея последовательность n битов $(x_1, \dots, x_n)$, $\forall j\ x_j \in \{0,1\}$, она служит аргументом булевой функции, $f(x_1, \dots x_1) \in \{0,1\}$. Полученное значение записывается в регистр на крайнюю позицию (x_0 на рис. 1), а $n+1$ значение записывается в гамму–выходящую последовательность. Так мы получаем новое состояние регистра с другой последовательностью, гамма заполняется итерационно.



Регистр сдвига с обратной связью

рис.1

Вопрос 1. Какой должна быть булева функция f, чтобы мы получали максимальный цикл последовательности гаммы?

Известны методы построения f с циклом 2^{n-1} используя линейные функции (все мономы АНФ первой степени для f и нет монома нулевой степени), с помощью нелинейной функции можно получить цикл длины 2^n, также представляют интерес свойства таких функций.

Определение 1. Нелинейные булевы функции максимального цикла для регистра обратной связи будем называть БМЦ – “булевы функции максимального цикла”.

Весь код, использованный в исследованиях, выложен на github (https://github.com/ArtemKhodzickii/BMC), все эксперименты можно повторить и удостовериться в их корректности. Также там приведены найденные нами функции максимального цикла.

Гипотеза 1. Всякая БМЦ сбалансирована.

Интуитивно ясно, что если булева функция f не будет сбалансирована, то 0 или 1 выходящей последовательности будут встречаться чаще, а функция будет чаще выдавать значения от одних и тех же аргументов.

Определение 2. лжеБМЦ – функции, удовлетворяющие всем свойствам, вытекающих из Гипотез, предъявляемым к БМЦ, но не имеющими максимальный цикл.

Гипотеза 2. В форме АНФ БМЦ не имеет слагаемого максимальной степени, но обязательно имеет слагаемые $x_1, \dots, x_{n-1}, 1$ и x_n.

Аргументы функции из $\mathbb{F}_2$ удобно представлять в качестве натуральных чисел, упорядочивая их по возрастанию, поскольку легко ассоциировать двоичный вектор с двоичным числом и переводить его в десятичную систему. Слагаемые, приведенные в гипотезе – инварианты БМЦ и лжеБМЦ относительно любых преобразований. Также было отмечено, что в их АНФ находится чётное число мономов. При этом они не имеют ни одного монома старшей степени в лексикографическом порядке.

Гипотеза 3. Существует множество аффинных преобразований, которые будут переводить БМЦ лишь в БМЦ.

Инварианты могут послужить существенным упрощением поиска множества таких преобразований. Также выявлено, что по m БМЦ можно гарантированно построить $2m$ нелинейных функций предмаксимального цикла, добавляя одни и те же мономы. Обратное, вообще говоря, неверно. При этом множество предмаксимальных функций, предположительно, замкнуто относительно аффинных преобразований особого вида.

Предложение 1. Если f – БМЦ, тогда $f(x) \oplus f\big(x \oplus (1,0,\dots 0)\big) \equiv 1$. В книге [2] приводится эквивалентное определение: $f(x_1, x_2, \dots, x_n) = 1 + f(x_1 \oplus 1, x_2 \dots x_n)$ с точностью до обозначения переменных.

Это ключевое условие для БМЦ, поскольку оно обеспечивает то, что БМЦ побывает на каждом из своих аргументов, то есть дает максимальный цикл. Данное свойство можно называть зеркальной симметричностью, поскольку представляя f таблицей истинности оно будет эквивалентно тому, что каждую из половин таблицы мы отразим в ее части, а если считать их вектором и сложить, то мы получим единичный вектор.

Гипотеза 4. Если f - БМЦ, то существует БМЦ g, двойственная к ней

$f(x) = \sim g(\sim x)$при этом f самодвойственной быть не может.

Последние Гипотеза и Предложение служат причиной появления инвариантов. Стоит теперь затронуть криптографические свойства БМЦ, ведь и там не обошлось без особенностей.

Предложение 2. Максимальный по модулю коэффициент Уолша-Адамара БМЦ равна 4 и 12 для 3 и 4 переменных соответственно.

Предложение 3. Для 3 и 4 переменных БМЦ имеет максимальную алгебраическую иммунность.

Гипотеза 5. Среди БМЦ обязательно найдется группа функций с максимальной алгебраической иммунностью.

Для 5 переменных нелинейность равна 12 или же 20, при этом нелинейность равная 12 достигается на всех функциях максимальной алгебраической иммунности, в обратную сторону, вообще говоря, это неверно. Для 5 переменных из 2048 функций максимальную иммунность имеют 896, остальные имеют алг. иммунность 2.

Гипотеза 6. Существует метод построения БМЦ от n переменных на основе БМЦ от m переменных, где $2 < m < n$.

Гипотеза 7. Корреляционная иммунность всякой БМЦ равна 0.

Предложение 4. [2] БМЦ от n переменных в точности $2^{2^{n-1}-n}$ штук. Стоит отметить, что доказательство данной формулы рекуррентно.

Все гипотезы были выдвинуты при помощи анализа функций и перебора, основанного на последнем Предложении. Полный перебор сразу для 3, 4 и 5 переменных с вышеперечисленными Гипотезами и Предложениями занял не больше 30 секунд, но уже для 6 переменных мы оценили время, которое бы ушло на перебор. Оценка этого перебора — 60 минут. Был проведен частичный перебор для n = 6, 8. Результаты перебора: для n = 3 найдены только БМЦ в количестве 2-х штук. Для n = 4 — 16 БМЦ и 16 лжеБМЦ, для n = 5 — 2048 БМЦ и 12288 лжеБМЦ. При этом выборка не сокращалась на основе Предложений относительно криптографических свойств, число лжеБМЦ уменьшилось бы незначительно, а ресурсы на проверку возросли бы в разы. Приведенный нами код может быть оптимизирован и легко использован для поиска всех нелинейных функций для малых $n\,(n \leq 7)$ за приемлемое время, при этом можно искать БМЦ для n от 8 до 13 при “правильном” заполнении начальных данных за приемлемое время. Сам же перебор легко произвести параллельными вычислениями. Стоит отметить, что лжеБМЦ составляют приблизительно $0,0003\%$ от общего количества функций 5 переменных, что наталкивает на мысль о "почти достаточных условиях". Мы встаем перед нерешенной, фундаментальной проблемой:

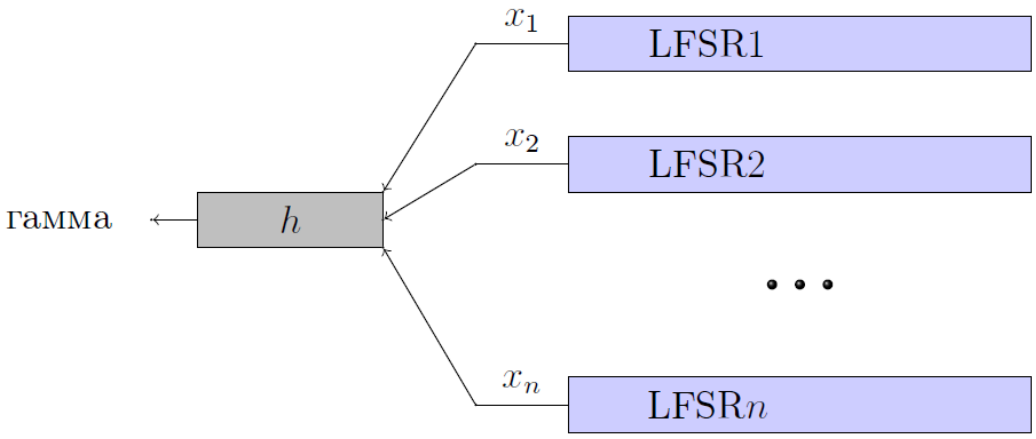
Вопрос 2. Каковы необходимые и достаточные условия определения БМЦ?

Перспективы:

БМЦ можно использовать для создания генератора последовательности псевдослучайных чисел, что нами и было предпринято, но, за нехваткой времени, не было закончено. Ожидается, что применение БМЦ улучшит криптографические свойства такой последовательности. Была использована комбинирующая модель с 8-ю функциями от 6 переменных, в качестве регистров, и комбинирующей функцией от 8 переменных.

Класс БМЦ является крайне специфическим, нет простых условий, позволяющих установить общий вид его функций, поиск достаточных условий или предложение построения некоторых конструкций БМЦ является ключевой вехой развития данного направления. Для этого служит большой простор для исследования – выявления

нетривиальных свойств таких функций, поиск закономерностей и рассмотрение преобразований над ними.



Комбинирующая модель

рис.2

ЛИТЕРАТУРА

1. Токарева Н.Н. – Симметричная криптография. Краткий курс, Новосибирский государственный университет, Новосибирск, 2012.Петров П. П. Название книги. М.: Наука,2009.
2. Solomon W. Golomb – Shift Register Sequences Secure and Limited-Access Code Generators, Efficiency Code Generators, Prescribed Property Generators, Mathematical Models, 2016.

Куратор исследования – к.ф.-м.н. доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research Токарева Наталья Николаевна.

.

SAT-РЕШАТЕЛИ В КРИПТОГРАФИИ

Решение криптографических задач с помощью SAT-решателей

А.Е. Доронин

Новосибирский государственный университет

E-mail: **artem96dor@gmail.com**

Представлен подход к решению некоторых криптографических задач, основанный на их сведении к классической задаче о выполнимости и последующем использовании SAT-решателей. Построены формулы, определяющие условия взаимной однозначности и дифференциальной равномерности векторной булевой функции, а также EA-эквивалентности пары функций.

Ключевые слова: SAT-решатели, криптография, булевы функции, взаимная однозначность, биекция, дифференциальная равномерность, EA-эквивалент\-ность

В настоящее время SAT-решатели используются для решения криптографических задач разного типа. Например, проведён криптоанализ асимметричной криптосистемы RSA [3], в результате которого удалось факторизовать числа до 417 бит; выполнен криптоанализ шифра Trivium, его упрощений и модификаций [4]. В [5] представлена гомоморфная криптосистема с открытым ключом, основанная на SAT-задаче. С помощью SAT=решателей успешно проверяется обратимость векторных булевых функций [6].

В данной работе предлагается использование SAT-решателей в задачах построения криптографических булевых функций и проверки эквивалентности двух булевых функций. Для получения набора булевых формул использованы следующие понятия и свойства.

Определение 1. Векторная булева функция $F: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ называется взаимно однозначной, если она инъективна и сюръективна, то есть одновременно выполняются следующие условия:

1. $\forall x' \in \mathbb{Z}_2^n \, \forall x'' \in \mathbb{Z}_2^n \, (x' \neq x'' \to F(x') \neq F(x''))$;

2. $\forall y \in \mathbb{Z}_2^n \, \exists x \in \mathbb{Z}_2^n \, (F(x) = y)$. [1]

Определение 2. Векторная булева функция $F: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ является дифференциально δ-равномерной, если для любого ненулевого $a \in \mathbb{Z}_2^n$и произвольного $b \in \mathbb{Z}_2^n$ уравнение $F(x) \oplus F(x \oplus a) = b$ имеет не более δ решений. [2]

Определение 3. Векторные булевы функции F G, действующие из $\mathbb{Z}_2^n$ в $\mathbb{Z}_2^n$, называются EA-эквивалент\-ными, если $G = B \circ F \circ A + C$, где A, B - аффинные перестановки; C - аффинная функция.[7]

Условия, фигурирующие в определениях, представляются в виде КНФ и подаются на вход SAT-решателя. В результате его работы происходит означивание переменных таким образом, чтобы формулы были истинными, а следовательно, условия выполнялись.

Векторные булевы функции были закодированы в двух представлениях:

1. В разреженном представлении используется 2^{2n} переменных $f_{x,y}$, из которых 2^n равны 1, остальные равны 0: $f_{x,y} = 1 \Leftrightarrow F(x) = y$, где $x, y \in \mathbb{Z}_2^n$.
2. В плотном представлении используется $n \cdot 2^n$ переменных $fb_{x,k}$: $fb_{x,k} = 1 \Leftrightarrow F_k(x) = 1$, где $F(x) = \big(F_0(x), F_2(x), \dots, F_{n-1}(x)\big)$, $F_k: \mathbb{Z}_2^n \to \mathbb{Z}_2^n, k = 0, \dots, n-1; x \in \mathbb{Z}_2^n$

Для записи условий на переменные $f_{x,y}$ и $fb_{x,k}$ понадобятся следующие вспомогательные переменные:

- $fbq_{x,y,k} = 1 \iff fb_{x,k} \neq fb_{y,k}$, где $k = 0, \dots, n-1; x, y \in \mathbb{Z}_2^n$;
- $d_{x,a,b} = 1 \iff F(x) \oplus F(x \oplus a) = b$, где $x, a, b \in \mathbb{Z}_2^n$;
- $de_{x,y,a} = 1 \iff F(x) \oplus F(x \oplus a) = F(y) \oplus F(y \oplus a)$, где $x, y, a \in \mathbb{Z}_2^n$;
- $dbq_{x,y,a,k} = 1 \iff fbq_{x,x\oplus a,k} = fbq_{y,y\oplus a,k}$, где $k = 0, \dots, n-1, x, y, a \in \mathbb{Z}_2^n$.

В КНФ эти зависимости записываются следующим образом:

$$\mathbf{SoP^D}(fb, fbq) = \bigwedge_{x,y,k} (fbq_{x,y,k} \vee fb_{x,k} \vee \overline{fb_{y,k}}) \wedge (fbq_{x,y,k} \vee \overline{fb_{x,k}} \vee fb_{y,k}) \wedge$$
$$\wedge (\overline{fbq_{x,y,k}} \vee fb_{x,k} \vee fb_{y,k}) \wedge (\overline{fbq_{x,y,k}} \vee \overline{fb_{x,k}} \vee \overline{fb_{y,k}});$$
$$\mathbf{SpDen}(f, fb) = \bigwedge_{x,y,k} (\overline{f_{x,y}} \vee fb_{x,k}) \wedge (f_{x,y} \vee \overline{fb_{x,0}^{y_0}} \vee \dots \vee \overline{fb_{x,n-1}^{y_{n-1}}});$$
$$\mathbf{Der^S}(f, d) = \bigwedge_{b,a,z,x} (f_{x,z} \vee f_{x\oplus a,z\oplus b} \vee \overline{d_{x,a,b}}) \wedge (f_{x,z} \vee \overline{f_{x\oplus a,z\oplus b}} \vee \overline{d_{x,a,b}}) \wedge$$
$$\wedge (\overline{f_{x,z}} \vee f_{x\oplus a,z\oplus b} \vee \overline{d_{x,a,b}}) \wedge (\overline{f_{x,z}} \vee \overline{f_{x\oplus a,z\oplus b}} \vee d_{x,a,b});$$
$$\mathbf{SoPEq^D}(fbq, dbq) = \bigwedge_{a,x,y,k} (dbq_{x,y,a,k} \vee fbq_{x,x\oplus a,k} \vee fbq_{y,y\oplus a,k}) \wedge$$
$$\wedge (dbq_{x,y,a,k} \vee \overline{fbq_{x,x\oplus a,k}} \vee \overline{fbq_{y,y\oplus a,k}}) \wedge (\overline{dbq_{x,y,a,k}} \vee fbq_{x,x\oplus a,k} \vee \overline{fbq_{y,y\oplus a,k}}) \wedge$$
$$\wedge (\overline{dbq_{x,y,a,k}} \vee \overline{fbq_{x,x\oplus a,k}} \vee fbq_{y,y\oplus a,k});$$

Свойства из определений 1 - 3 можно записать следующими формулами.

Теорема 1. Переменные $f_{x,y}$ задают функцию $F: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ тогда и только тогда, когда следующая формула является истинной:

$$\mathbf{F^S}(f) = \bigwedge_{x \in \mathbb{Z}_2^n} \Big(\bigwedge_{\substack{y', y'' \in \mathbb{Z}_2^n \\ y' < y''}} \overline{f_{x,y'}} \vee \overline{f_{x,y''}} \Big) \wedge \bigwedge_{x \in \mathbb{Z}_2^n} \Big(\bigvee_{y \in \mathbb{Z}_2^n} f_{x,y} \Big). \quad (1)$$

Формула (1) состоит из $2^{3n-1} - 2^{2n-1}$дизъюнкций длины 2 и 2^nдизъюнкций длины 2^n.

Теорема 2. Переменные $f_{x,y}$ задают взаимно однозначную векторную булеву функцию $F: \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ тогда и только тогда, когда следующая формула является истинной:

$$\mathbf{P^S}(f) = \bigwedge_{y\in\mathbb{Z}_2^n} \Big(\bigwedge_{\substack{x',x''\in\mathbb{Z}_2^n \\ x'<x''}} \overline{f_{x',y}} \vee \overline{f_{x'',y}} \Big) \wedge \mathbf{F^S}(f). \quad (2)$$

Формула (2) состоит из $2^{3n} - 2^{2n}$ дизъюнкций длины 2 и 2^n дизъюнкций длины 2^n.

Теорема 3. Переменные $fb_{x,k}$, $fbq_{x,y,k}$ задают взаимно однозначную векторную булеву функцию $F\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ тогда и только тогда, когда следующая формула является истинной:

$$\mathbf{P^D_{sum}}(fb, fbq) = \bigwedge_{x,y\in\mathbb{Z}_2^n} \bigvee_k fbq_{x,y,k} \wedge \mathbf{SoP^D}(fb, fbq). \quad (3)$$

В формуле (3) содержится $n(2^{2n} - 2^n)$ дизъюнкций длины 3 и $2^{2n} - 2^n$дизъюнкций длины n.

Теорема 4. Переменные $fb_{x,k}$, $d_{x,a,b}$ задают APN-функцию $F\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ тогда и только тогда, когда выполняются условия теоремы 1 и следующая формула является истинной:

$$\mathbf{APN^S}(f, d) = \mathbf{Der^S}(f, d) \wedge \bigwedge_{\substack{b\neq 0, a\neq 0, \\ x,y\neq x}} (\overline{d_{x,a,b}} \vee \overline{d_{y,a,b}}). \quad (4)$$

В формуле (4) содержится порядка 2^{4n}дизъюнкций длины 3 и 2.

Теорема 5. Переменные $fb_{x,k}$, $fbq_{x,y,k}$ и $dbq_{x,y,a,k}$задают APN-функцию $F\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ тогда и только тогда, когда следующая формула является истинной:

$$\mathbf{APN^D}(fb, fbq, dbq) = \mathbf{SoPEq^D}(fbq, dbq) \wedge \mathbf{SoP^D}(fb, fbq) \wedge \bigwedge_{a,x,y} \bigvee_k \overline{dbq_{x,y,a,k}}. \quad (5)$$

В формуле (5) содержится порядка 2^{3n} дизъюнкций длины 3 и n.

На основе полученных формул генерируется входной файл для SAT-решателя. Формулы можно также использовать для тестирования работы новых SAT-решателей, созданных для криптографических задач.

В дальнейшем планируется описание свойств нелинейности булевой функции и в частности бент-функции в виде SAT-задачи, а также реализация задачи проверки эквивалентности двух векторных булевых функций.

ЛИТЕРАТУРА

1. Верещагин Н. К., Шень. А. – Часть 1. Начала теории множеств // Лекции по математической логике и теории алгоритмов. 4-е изд., испр. — М.: МЦНМО, 2012. — 112 с.
2. Городилова А. А. – От криптоанализа шифра к криптографическому свойству булевой функции // Прикладная дискретная математика. 2016. №3 (33).
3. Огородников Ю. Ю. – Комбинированная атака на алгоритм RSA с использованием sat-

подхода // Динамика систем, механизмов и машин. Омск: ОмГТУ, 2016. С. 276–284.

4. Заикин О. С., Отпущенников И. В., Семёнов А. А. – Оценки стойкости шифров семейства Trivium к криптоанализу на основе алгоритмов решения проблемы булевой выполнимости. // Прикладная дискретная математика. Приложение. 2016. № 9. С. 46–48.
5. Schmittner S. E. – A SAT-based Public Key Cryptography Scheme. IACR Cryptol. ePrint Arch. 2015.
6. Wille R., Lye A., Niemann P. – Checking reversibility of Boolean functions. LNCS. 2016. V. 9720. P. 322–337.
7. Canteaut A., Perrin L. – On CCZ-Equivalence, Extended-Affine Equivalence, and Function Twisting // Finite Fields and their Applications. 2018. № 56.

Куратор исследования – Калгин Константин Викторович , к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Применение ROBDD для решения криптографических задач

С. Е. Ким

Новосибирский государственный технический университет,

E-mail: **stanislav.kim.coder@gmail.com**

Одним из методов решений систем булевых уравнений является Binary decision diagrams (BDD). В BDD существует ряд особенностей, которые позволяют ей работать эффективнее, чем SAT решатели. Однако ограничением BDD является память (в отличии от SAT решателя, где основное ограничение время работы): с увеличением размера функции, размер BDD сильно возрастает. Поэтому были проведены исследования эффективности построения BDD для больших систем булевых уравнений.

Ключевые слова: Binary decision diagrams, boolean functions, оптимизация, Conjunctive normal form

Binary decision diagrams (BDD) – бинарная диаграмма решений, вид представления булевой функции в виде дерева решений, где узел это переменная, а ребра – её значение (истина/ложь).

Пример представления в виде BDD функции (1) (рис.1)

$$f(\bar{x}) = (x_0 \wedge x_1) \vee (x_2 \wedge x_3 \wedge x_0) \quad (1)$$

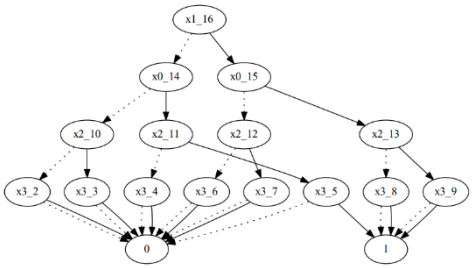


Рис. 1. Пример представления формулы (1) в виде BDD

Reduced Ordered Binary Decision Diagram (ROBDD) – сжатая BDD, существуют два правила сжатия BDD в ROBDD:

1) Если два ребра из узла ведут в одну вершину, то этот узел можно заменить на вершину, в которую этот узел указывал;
2) Если у двух узлов, с одинаковой переменной, совпадают вершины, на которые указывают ребра, то эти узлы можно объединить в один узел.

Пример представления в виде ROBDD функции (1) (рис.2):

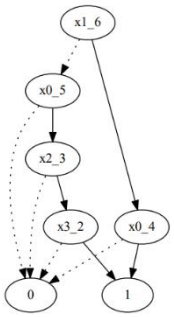


Рис. 2. Пример представления формулы (1) в виде ROBDD

В работе [1] описаны алгоритмы реализующие ряд операций над ROBDD. Один из таких алгоритмов, это объединение двух ROBDD в одну с помощью логического оператора (Apply)[1].

Пример сложения двух функций (2-4) (рис.3)

$$f_1(\bar{x}) = x_0 \wedge x_1 \tag{2}$$

$$f_2(\bar{x}) = x_2 \wedge x_3 \wedge x_0 \tag{3}$$

$$f(\bar{x}) = f_1 \vee f_2 \tag{4}$$

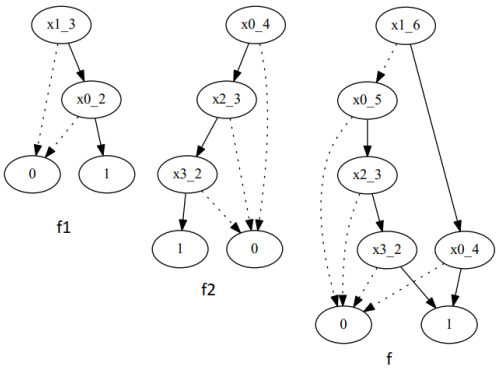


рис. 3. Пример объединения двух функций (2,3) с помощью алгоритма Apply ROBDD

Была проанализирована эффективность определения истинности формулы, представленной в КНФ с помощью ROBDD. Для построения диаграммы для всей формулы, вначале строятся ROBDD для каждого дизъюнкта, а затем они объединяются.

Для представления затрачиваемого времени, приведена оценка работы алгоритмов для построения и обработки BDD (таблица 1):

Алгоритм	Оценка
Создание BDD	$O(\sum_{i=1}^{m} 2^m)$, где m количество переменных входящих в выражение
Сжатие BDD (Reduce)	$O(N\ log\ N)$, где N количество узлов в дереве
Объединение BDD (Apply)	$O(N_G * N_V)$, где N_G количество узлов в дереве G, а N_V количество узлов в дереве V

Таблица 1. Оценка времени работы алгоритмов

Для создания BDD дизъюнкта КНФ требуется малые затраты времени, так как они содержат достаточно мало переменных. Время в алгоритмах сжатия и объединения BDD зависит от размера BDD, поэтому далее будут продемонстрированы результаты оценки размеров получаемой BDD.

Сначала исследования проводились для КНФ небольших размеров (50 переменных, 200 дизъюнктов) [2].

Рис. 4. График динамики изменения размера ROBDD

Данный пример (рис.4) демонстрирует, что размер ROBDD ближе к центру, сильно возрастает, это связанно с особенностями построения ROBDD. На размер ROBDD влияют порядок объединения дизъюнктов и порядок переменных.

Далее проводились исследования влияния порядка объединения дизъюнктов и порядка переменных.

Порядок объединения дизъюнктов можно определить тремя способами:

1) В зависимости от размера дизъюнкта;
2) В зависимости от частоты встречаемости переменной;
3) Смешанная зависимость.

Зависимость от размера дизъюнкта – подаются выражения по возрастанию/убыванию количества переменных в дизъюнкте.

Зависимость от частоты встречаемости переменной – для начала считается, сколько раз и какая переменная встретилась во всей КНФ, назовем эту величину q_i, где $i = \overline{1,n}, n$ –количество переменных. Далее вычисляется вес каждого дизъюнкта (5), и объединяются выражения по возрастанию/убыванию веса i-го дизъюнкта, w_i:

$$w_i = \sum_{k=1}^{m_i} q_{s_{ik}}, \quad (5)$$

где m_i – размер дизъюнкта под номером i,

S_{ik} – номер k-ой переменной в i –ом дизъюнкте,

q_i – встречаемость переменной во всей формуле.

Смешанная зависимость – подвыражения подаются по возрастанию веса, заданные формулой (6)

$$p_i = A * m_i + B * w_i, \quad (6)$$

где A, B задаваемые коэффициенты.

Таким образом было произведено несколько наблюдений

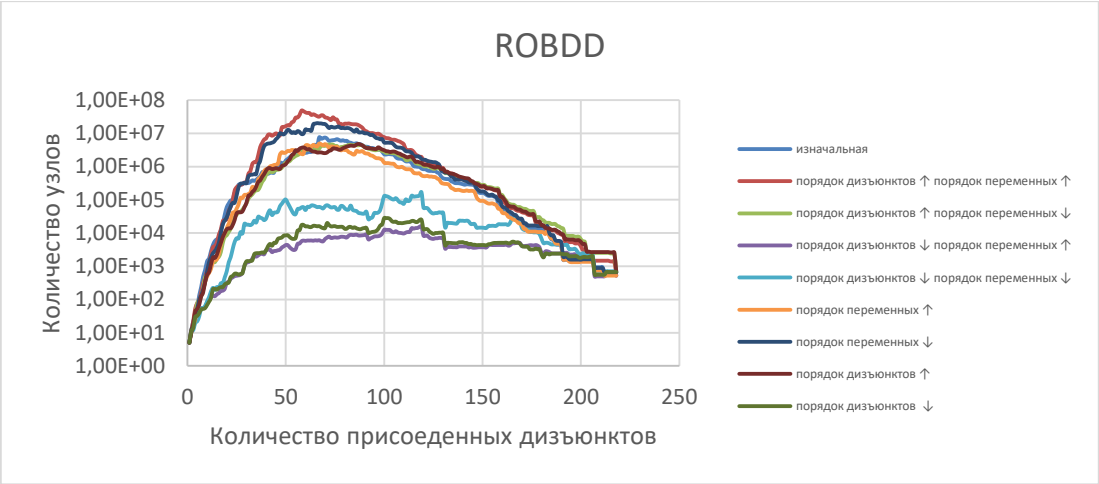


Рис. 5. Графики динамики изменения размера ROBDD в зависимости от порядка

Для анализа оптимизаций была взята сгенерированная КНФ [2], содержащая 218 дизъюнктов и 50 переменных, при этом каждое выражение содержало по 3 переменные, поэтому оптимизация по размеру дизъюнктов здесь не продемонстрирована.

Из наблюдений видно (рис.5), что самым оптимальным решением является убывающий порядок дизъюнктов по частоте переменных, и возрастающий порядок переменных. Немного уступает простая сортировка по порядку дизъюнктов. Стоит также заметить, что при некоторых конфигурациях, подсчет становится значительно дольше, и памяти для этого требуется значительно больше.

Далее были проведены сравнения изначальной системы и системы с субоптимальным порядком дизъюнктов и переменных для других простых сгенерированных КНФ [2], в результате среднее количество узлов уменьшилось в 50-360 раз.

Следующие исследование было проведено на больших уравнениях, сгенерированных по поточному шифру Grain (80 переменных, 110 000 дизъюнктов) (рис.6).

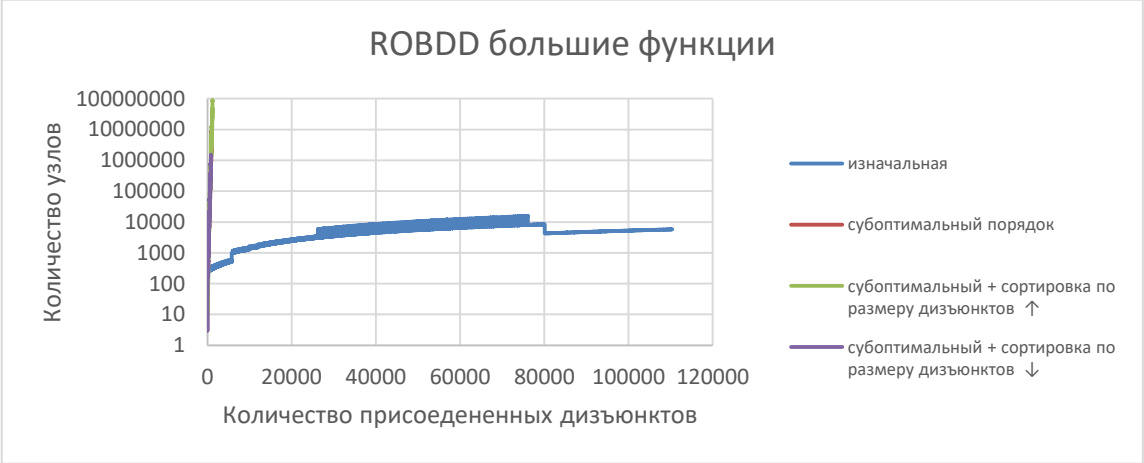


Рис. 6. График динамики изменения размера ROBDD для большого уравнения

К сожалению, изначальный порядок оказался более оптимальным, чем субоптимальные порядки. Это связанно со следующими особенностями изначального порядка переменных: младшие переменные отвечают за исходные биты ключа, а самые старшие переменные за биты гаммы (генерируемые последовательностью).

Также был исследован другой метод построения КНФ, построение нескольких деревьев одновременно. Идея заключалась в том, чтобы не присоединять маленькие

ROBDD к огромной (итоговой) ROBDD, а объединять каждую маленькую ROBDD попарно с другой ROBDD. В итоге склеивались две большие ROBDD. Однако существенный недостаток этого метода, это требуемая память. Хотя данный метод по скорости является эффективнее, чем метод последовательного объединения. Возможно проблема с памятью решится, если, например, строить по принципу группирования множеств построения, разделенных по переменным.

Исследовано влияние порядка переменных, и порядка присоединения дизъюнктов, для построения КНФ. Можно сделать вывод, что порядок играет значительную роль в построении ROBDD, так как время и память требуемая для построения итоговой ROBDD может сильно меняться. На малых КНФ, было установлено, что порядок, в котором переменные организованы по возрастанию, а объединение дизъюнктов по убыванию частоты встречаемости переменных, является субоптимальным. Однако на больших КНФ субоптимальный порядок оказался абсолютно не эффективным. Возможно стоит обратить внимание на структуру исходной задачи, так как большие задачи, были получены из шифров, где порядок дизъюнктов не был случайным, в отличие от малых задач, которые были случайно сгенерированы.

В дальнейшем планируется сравнить размеры и динамику построения ROBDD из КНФ и АНФ шифра Grain.

ЛИТЕРАТУРА

1. Randal E.Bryant.Graph-Based Algorithms for Boolean Function Manipulation//1986 С.1–28.
2. SATLIB - Benchmark Problems URL: https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

Куратор исследования – Калгин Константин Викторович, к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Разработка SAT-решателя с нуля

С.Ю. Побединский

Новосибирский государственный технический университет

E-mail: sergeypobwer@gmail.com

В работе рассматривается реализации SAT-решателя на основе алгоритма DPLL [1] с эвристикой выбора переменных на основе их частотности. Выполнен анализ эффективности каждой эвристики в ускорении программы как по отдельности, так и совместно. На основе полученных данных выявилась проблема оригинального алгоритма.

Ключевые слова: DPLL, Unit propagation, Pure literal, КНФ

SAT-задача – булева формула, состоящая только из имён переменных, скобок и операций $\wedge$ (И), $\vee$ (ИЛИ) и $\neg$ (НЕ). Задача заключается в следующем: можно ли назначить всем переменным, встречающимся в формуле, значения «ложь» и «истина» так, чтобы формула стала истинной.

SAT-решатель – программа, которая ищет значения переменных, при которых формула принимает значение истина. Базовый алгоритм, которые лежит в основе всех современных SAT-решателей – DPLL, разработан в 1962 году Мартином **Д**эвисом, Хилари **П**атнэмом, Джорджем **Л**огеманом и Дональдом **Л**авлендом. Основное отличие алгоритма перебора в глубину (DFS, Depth First Search) от DPLL – наличие шагов «распространение переменной» (unit propagation) и определение «чистых литералов» (pure literal)

DPLL(v)

1. Если есть ложный дизъюнкт – return FALSE
2. Если все переменные означены – return TRUE
3. Для каждого дизъюнкта, в котором все литералы ложные кроме одного, unit_propagation (вывод значения последнего литерала)
4. Для каждой чистой переменной pure_literal (литерал)
5. выбрать следующую переменную v
6. return DPLL(v=0) || DPLL(v=1)

DFS(v)

1. Если есть ложный дизъюнкт – return FALSE
2. Если все переменные означены – return TRUE
3. выбрать следующую переменную v
4. return DPLL(v=0) || DPLL(v=1)

Кроме самого алгоритма DPLL реализована предобработка формулы, заключающаяся в поиске заведомо истинных дизъюнктов и вычеркивания их из формулы.

Распространение переменной (unit propagation) – если дизъюнкт содержит ровно один литерал, значение которого ещё неизвестно, и все остальные литералы ложные, то этот последний литерал принимает значение истина, поскольку этот дизъюнкт может принять значение «истина» только в случае присвоения последнему литералу соответствующего значения. Таким образом, не нужно делать выбор значения переменной. На практике за этим следует каскад присвоений переменным значений, таким образом существенно сокращается количество вариантов перебора.

Исключение «чистых» переменных – если некоторая переменная входит в формулу только с одной «полярностью» (то есть либо только без отрицаний, либо только с отрицаниями) во все дизъюнкты, которые ещё не истинны, то она называется чистой. «Чистым» переменным всегда можно задать значение так, что все содержащие их дизъюнкты станут истинными.

Эвристику исключения «чистой» переменной (pure literal) трудно реализовать эффективно. Так как предполагается проход по всем действующим скобкам, что значительно увеличивает время работы программы. Поэтому идея, представленная в оригинальном алгоритме DPLL, в современных SAT-решателях не используется. Не исключено, что на некоторых формулах эвристика может придать ускорение.

Реализован решатель ограничений в КНФ (классический SAT-решатель) на основе алгоритма DPLL с эвристикой выбора переменных на основе их частотности с возможностью подключения\отключения эвристик.

Было реализовано 5 вариантов решателя:

1. DFS – обход в глубину
2. DFS + UP – обход в глубину с распространением переменных
3. DFS + UP + PL – обход в глубину с распространением переменных и исключением «чистых» переменных, полный вариант алгоритма DPLL
4. DFS + FREQ – обход в глубину с учетом частотности вхождения переменных в формулу
5. DFS + UP + FREQ – обход в глубину с распространением переменных и частотностью вхождения переменных

Для тестирования пяти реализаций использовались тесты с сайта SAT Benchmarks [2] в формате Uniform Random-3-SAT (с тремя литералами в дизъюнкте) в диапазоне от 50 до 250 переменных. Результаты тестирования представлены в Таблице и на Рисунке 1. Так как эвристика исключения «чистой» переменной показала отрицательную эффективность, реализация с ней не представлена в Таблице.

Среднее время работы (ограничение 30 сек.)		Количество переменных								
		50	75	100	125	150	175	200	225	250
Реализация	DFS+UP+FREQ	0.0003	0.0018	0.0078	0.0353	0.1384	0.681	2.6953	92%	44%
	DFS+UP	0.0016	0.0177	0.2247	2.5148	76%	23%	3%	1%	
	DFS+FREQ	0.3123	26%	1%						
	DFS	20%								

Таблица. Среднее время работы разных реализаций с разным количеством переменных. В случае когда не все задачи успевали решиться за 30 секунд, показан процент решенных задач.

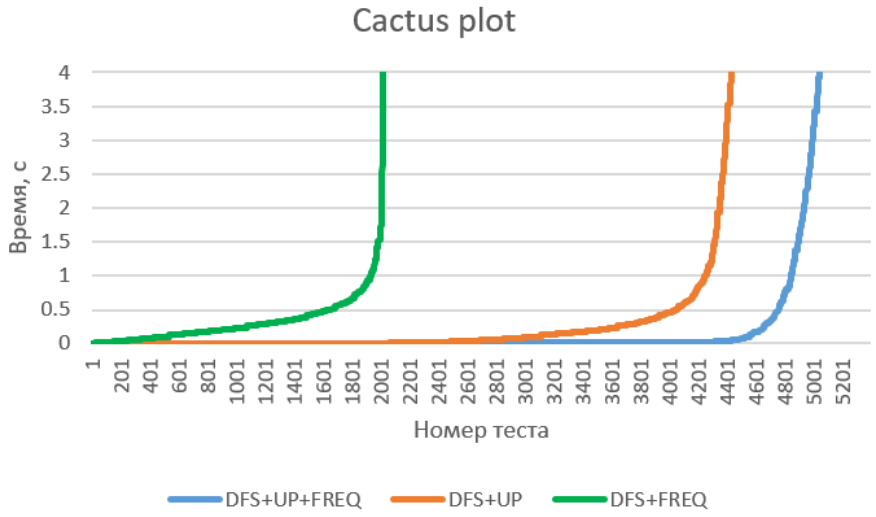


*Рис.*1 Cactus-plot для трёх вариантов реализаций (DFS+UP+FREQ, DFS+UP, DFS+FREQ). Время решения задач отсортировано по возрастанию для каждой реализации. Тестирование проводилось на всех задачах [2] с количеством переменных от 50 до 250.

Эвристика распространения переменной даёт намного больший прирост в ускорении в сравнении со всеми другими оптимизациями по отдельности (100 – 1000 раз). Учитывание частоты вхождения переменных в формулу на этапе выбора следующей переменной на большинстве тестов показывает ускорение в 50 – 100 раз. Комбинирование алгоритма распространения переменной и выбора переменных по их частотности без подключения эвристики исключения «чистой» переменной даёт наилучшие показатели по времени.

В дальнейшем предполагается реализация алгоритма CDCL [3], основанный на уже написанном алгоритме DPLL.

Также после работы алгоритма распространения переменной и перед работой основного алгоритма было бы уместно выполнить DP-алгоритм, Рисунок 2. Где, выбирая переменную, требуется преобразовать скобки, пока не останутся только с уникальными наборами переменных, параллельно удаляя очевидные дизъюнкты с уже известными истинными переменными.

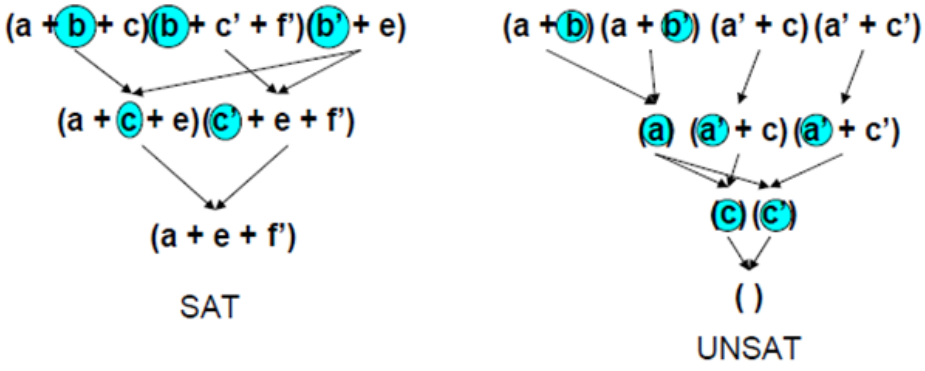


Рис.2 Пример работы DP-алгоритма.

Разработка CDCL придаст существенный прогресс в применении SAT-решателей на практике и позволит решать уравнения с сотнями тысяч переменных.

На основе полученных реализаций показано, что отличие DPLL от DFS благодаря шагу unit propagation является существенным и позволяет достичь многократного ускорения работы

алгоритма, и тем самым даёт возможность решать существенно бóльшие по количеству переменных задачи.

ЛИТЕРАТУРА

1. Davis, Martin; Putnam, Hilary. A Computing Procedure for Quantification Theory. Journal of the ACM, 7 (3): 201–215. 1960. doi:10.1145/321033.321034.
2. SATLIB - Benchmark Problems URL: https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html
3. J.P. Marques-Silva; Karem A. Sakallah (November 1996). "GRASP-A New Search Algorithm for Satisfiability". Digest of IEEE International Conference on Computer-Aided Design (ICCAD). pp. 220–227

Куратор исследования – Калгин Константин Викторович, к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Применение SAT-решателей для криптоанализа потоковых шифров

Д. А. Софронова

Новосибирский государственный университет

E-mail **d.sofronova1@g.nsu.ru**

В работе представлен транслятор, позволяющий преобразовывать описание криптографической задачи (криптоанализ шифра или хэш-функции, поиск APN функций) в КНФ. После преобразования, SAT-решатель устанавливает истинность формулы и находит набор, выполняющий КНФ. Отличительные особенности данной разработки – универсальность, малый объем исходного кода (300 строк C++), легко модифицируемая и расширяемая реализация.

Ключевые слова: криптоанализ, SAT-решатель, атака "угадай-и-вычисли".

В основе одного из методов анализа симметричных шифров лежит использование SAT-решателей. По алгоритму, задающему исходную криптографическую функцию, строится КНФ. Если для данной КНФ удается найти выполняющий набор, то имеем решение задачи криптоанализа исходной функции. SAT-задача – задача определения выполнимости логической формулы [1]. SAT-решатель – программа, которая ищет набор значений переменных, на котором формула истинна. Логическая формула записывается в конъюнктивной нормальной форме (конъюнкция дизъюнкций переменных или их отрицаний, далее – КНФ). Известно, что эта задача NP-полная. Несмотря на это, для множества практических задач SAT-решатели определяют выполнимость формул с тысячами переменных за приемлемое время. Для проведения криптоанализа с помощью SAT-решателя необходим только механизм для представления криптографических алгоритмов в виде КНФ в формате DIMACS.

На данный момент существует несколько разработок, позволяющих на выходе получать КНФ. Здесь приведем краткое описание двух разработок, специализирующихся на криптоанализе шифров, Grain of salt [2] и Transalg [3].

Transalg универсален и позволяет сводить к задаче о выполнимости не только криптографические задачи, но и некоторые задачи биоинформатики. Описание шифров происходит на специальном си-подобном языке с последующей генерацией КНФ. На данный момент при помощи Transalg построены SAT-кодировки многих симметричных алгоритмов, а также хэш-функций [4]. Являясь полноценным транслятором, Transalg анализирует текст описания с помощью лексического, синтаксического и семантического анализаторов, что делает его достаточно сложным для модификации и расширения.

Grain of Salt (далее – GoS) – программный комплекс описания поточных шифров и последующего автоматического проведения атаки "угадай-и-вычисли", который разработал автор cryptominisat [5], M. Soos. Данный вариант хорошо оптимизирован с помощью карт Карно, espresso (логический оптимизатор), поэтому выходная КНФ имеет меньший размер по сравнению с КНФ, полученной без оптимизаций. GoS предназначен для описания поточных шифров, построенных на базе регистров сдвига. Другая важная особенность — автоматизация проведения атаки "угадай-и-вычисли" на шифр. Автором [2] были построены SAT-кодировки шифров Grain, Trivium, Bivium, Crypto1 и Hitag2. Данная разработка позволяет описать только шифры,

основанные на регистрах сдвига и фильтрующих функциях, другие не могут быть представлены в этом программном комплексе (например, A5/1 из-за отсутствия поддержки if/else конструкции, другие симметричные шифры и хэш-функции).

В данной работе представлен программный комплекс, одновременно универсальный, легко расширяемый, простой и понятный для пользователей (в том числе на уровне реализации). Под криптографическими задачами далее подразумеваем не только задачи анализа шифров и хэш-функций, но и задачи поиска APN функций, определения EA эквивалентности булевых и векторных функций.

Основная идея заключается в том, что криптографическая задача (алгоритм или множество ограничений) описывается на языке C++ с использованием специальных классов varBool и varInt, у которых переопределены все операторы. Полиморфизм в C++ позволяет переопределить работу операторов для новых типов так, что при выполнении некоторых действий над данными происходит формирование КНФ (в зависимости от операций добавляются разные конструкции) или же реальное исполнение алгоритма. Через указание параметров при компиляции можно получить реализацию исходного алгоритма или генератор, получающий на выходе КНФ. Также есть возможность задать значения определенных переменных varBool до генерации КНФ, например, для частичного задания битов ключа при проведении атаки "угадай-и-вычисли". При использовании C-интерфейса SAT-решателя cryptominisat можно запускать решатель без промежуточной записи КНФ в файл.

Работа программы построена на операциях, обрабатывающих новые типы и неявно формирующих КНФ на основе логики операций. Немаловажным плюсом является то, что большинство шифров описываются на языке C. Построение задачи криптоанализа таких шифров легко осуществляется в проекте заменой типов данных в коде. Аналогичным образом преобразуются алгоритмы, описанные на языке TransAlg. Программа является гибкой, использование возможностей языка C++ (циклы, условные операторы, классы, шаблоны) позволяет описывать алгоритмы разной сложности. На данном этапе с использованием транслятора и описанных в нем механизмов регистров линейного сдвига построены SAT-кодировки шифров, описанных в [1], а также известный генератор A5/1. Получившиеся последовательности КНФ содержат больше переменных, чем последовательности, полученные работой GoS, но имеют меньше строк. Также количество переменных и строк в разработанном проекте зависит от количества неизвестных битов регистра, у GoS – не зависит. Из этого следует, что проект нуждается в оптимизации.

ЛИТЕРАТУРА

1. Biere A., Heule M., Maaren H. and Walsh T. Handbook of Satisfability // Frontiers in Artificial Intelligence and Applications. Vol.185. IOS Press, 2009.
2. Soos M. Grain of salt—an automated way to test stream ciphers through SAT solvers // Tools. Vol.10. Pp. 131-144, 2010
3. Semenov A., Otpuschennikov I., Gribanova I., Zaikin O. and Kochemazov S. Translation of algorithmic descriptions of discrete functions to SAT with application to cryptanalysis problems // Logical Methods in Computer Science. 2020. Vol. 16. Iss. 1. Pp. 29:1–29:42.
4. Otpuschennikov I., Semenov A., Gribanova I., Zaikin O. and Kochemazov S. Encoding Cryptographic Functions to SAT Using TRANSALG System //Proc. ECAI 2016, Frontiers in Artificial Intelligence and Applications, vol.285. IOS Press, 2016.

5. Soos M., Nohl K. and Castelluccia C. Extending SAT Solvers to Cryptographic Problems // Proc. 12th International Conference, SAT 2009, Swansea, UK, June 30 – July 3, 2009. LNCS. Vol. 5584. P. 244-257. 6.
6. URL: https://gitlab.com/satencodings/satencodings/(дата обращения: 20.05.2020).

Куратор исследования – Калгин Константин Викторович, к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

КРИПТОАНАЛИЗ СИММЕТРИЧНЫХ ШИФРОВ

Разностные характеристики побитового сложения по модулю 2 относительно сложения по модулю 2^n

Д.А. Ахтямов [3], Т.А. Бонич [2],Б.Ф. Жантуликов[2], Е.А. Ищукова[1], М.А. Панферов[2], И.А. Сутормин[2], К.М. Титова[2]

[1]Южный федеральный университет

[2]Новосибирский государственный университет,

[3]The Hebrew University of Jerusalem

E-mail: **akhtyamoff1997@gmail.com, t.bonich@g.nsu.ru, b.zhantulikov@g.nsu.ru, uaishukova@sfedu.ru, m.panferov@g.nsu.ru, ivan.sutormin@gmail.com, sitnich@gmail.com**

В работе рассмотрены свойства величины $\mathrm{adp}^{\oplus}(\alpha,\beta\to\gamma)$, которая является вероятностью разностей входных α, β и выходной γ операции XOR относительно операции сложения по модулю 2^n. Данная величина используется при проведении дифференциального криптоанализа в шифрах архитектуры ARX. Было доказано, что $max_{\beta\in\mathbb{Z}_2^n}\, adp^{\oplus}(0,\beta\to\gamma) = adp^{\oplus}(0,\gamma\to\gamma)$, а также установлены некоторые другие свойства $adp^{\oplus}(\alpha,\beta\to\gamma)$ и $adp^{\oplus}(0,\gamma\to\gamma)$.

Ключевые слова: Дифференциальный криптоанализ, ARX.

1. Введение

В симметричной криптографии популярными являются шифры архитектуры ARX. Они используют три основные операции: сложение (addition) – $x + y \pmod{2^n}$, циклический сдвиг (rotation) – $R(x_1, x_2, \dots, x_n) = (x_n, x_1, \dots x_{n-1})$, XOR - $x\ XOR\ y = x \oplus y$, где $x, y \in \mathbb{Z}_2^n$. С x мы также ассоциируем целое число $x_1 + x_2 2^1 + \cdots + x_n 2^{n-1}$.

Для таких шифров применим дифференциальный криптоанализ. Этот метод криптоанализа основан на изучении преобразования разностей открытых текстов в разности шифртекстов, подробнее см. в [1]. В данном случае, за разность удобно брать либо разность по модулю $2^n(+)$, либо XOR $(\oplus)$.

Если выбирается XOR как разность, то эффективность дифференциального криптоанализа зависит от величины $xdp^{+}(\alpha,\beta\to\gamma) = Pr_{x,y}[\big((x\oplus\alpha)+(y\oplus\beta)\big)\oplus(x+y)=\gamma$. В другом случае («-» в качестве разности), эффективность зависит от $adp^{\oplus}(\alpha,\beta\to\gamma) = Pr_{x,y}[\big((x+\alpha)\oplus(y+\beta)\big)-(x\oplus y)=\gamma]$.

В работе [2] приводится формула для подсчета $adp^{\oplus}$ с помощью матриц за линейное количество арифметических операций. Она выглядит следующим образом:

$$adp^{\oplus}(\alpha,\beta\to\gamma) = L\cdot A_{4\alpha_{n-1}+2\beta_{n-1}+\gamma_{n-1}}\cdot\ldots\cdot A_{4\alpha_0+2\beta_0+\gamma_0}\cdot C,$$

где L= $(1,1,1,1,1,1,1,1), C = (1,0,0,0,0,0,0,0)^T$,

$$A_0 = \frac{1}{4}\begin{pmatrix} 4 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

и $A_k, k \neq 0$, получается перестановкой из A_0 i-той строки с $i \oplus k$ и j-того столбца с $j \oplus k$, то есть $(A_k)_{i,j} = (A_0)_{i\oplus k, j\oplus k}$.

Также в работе [3] были предложены способы получения аналогичных формул не только для $adp^{\oplus}(\alpha, \beta \to \gamma)$, но и $xdp^{+}(\alpha, \beta \to \gamma)$ и других функций. В работе [2] была утверждена теорема (теорема 3), которая гласит, что для любой γ, $adp^{\oplus}(0, \gamma \to \gamma) = max\, adp^{\oplus}(\alpha, \beta \to \gamma)$. Но доказательство этой теоремы было опущено («The proof is omitted from the conference version») и далее нигде не появилось. Эта проблема нас заинтересовала, поэтому данная работа посвящена свойствам величины $adp^{\oplus}(0, \gamma \to \gamma)$ и особенностям $adp^{\oplus}$ в целом.

2. Результаты

Из линейного представления $adp^{\oplus}$ и анализа произведений матриц A_0 и A_3 мы получили рекуррентные формулы для $adp^{\oplus}(0, \gamma \to \gamma)$.

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} a_n \\ b_n \end{pmatrix} = \begin{cases} \begin{pmatrix} a_{\frac{n}{2}} + \frac{1}{4} b_{\frac{n}{2}} \\ \frac{1}{4} b_{\frac{n}{2}} \end{pmatrix}, & \text{если } n \text{ четное} \\ \begin{pmatrix} \frac{1}{4} a_{\frac{n-1}{2}} \\ b_{\frac{n-1}{2}} + \frac{1}{4} a_{\frac{n-1}{2}} \end{pmatrix}, & \text{если } n \text{ нечетное} \end{cases}$$

Пусть $\tilde{\gamma}$- целое число, полученное обращением битов γ. Тогда

Предложение 1.

$$adp^{\oplus}(0, \gamma \to \gamma) = a_{\lfloor \frac{\tilde{\gamma}}{2} \rfloor} + b_{\lfloor \frac{\tilde{\gamma}}{2} \rfloor}.$$

Для доказательства максимальности $adp^{\oplus}(0, \gamma \to \gamma)$ мы пробовали разные подходы и доказали один из частных случаев, а именно когда один из аргументов равен нулю.

Теорема 1. Для каждого $\gamma \in \mathbb{Z}_2^n$ верно следующее равенство

$$\max_{\beta \in \mathbb{Z}_2^n} adp^{\oplus}(0, \beta \to \gamma) = adp^{\oplus}(0, \gamma \to \gamma).$$

Для вероятностей $adp^{\oplus}(\alpha, \beta \to \gamma)$, не равных единице, мы знаем оценку сверху из работы [2]. Используя матричное представление $adp^{\oplus}$ нам удалось вывести и нижние оценки.

Предложение 2. Если $adp^{\oplus}(\alpha, \beta \to \gamma) \neq 0$, то $adp^{\oplus}(\alpha, \beta \to \gamma) \geq \frac{1}{2^{2n-3}}$

С помощью полученных ранее рекуррентных формул мы также вывели оценки снизу для $adp^{\oplus}(0, \gamma \to \gamma)$.

Предложение 3. $adp^{\oplus}(0, \gamma \to \gamma) \geq \frac{\omega(1,3,1,-4)_n}{2^{2n-3}}$,
где $\omega(p, q, r, s)_n$ - последовательность Хорадама − обобщение чисел Фибоначчи, определенное значениями $H_0 = p, H_1 = q,$ и $H_n = sH_{n-1} + rH_{n-2}$.

Предложение 4. Минимальное значение $adp^{\oplus}(0, \gamma \to \gamma)$ будет достигнуто на элементе $\gamma = k_{\forall}k\neg kk \dots k\neg k1$, где $k_{\forall}, k \in \{0,1\}$.

Предложение 5. Для любых различных $a, b, c, d \in \mathbb{Z}_2^n$ справедливо неравенство:
$$adp^{\oplus}(a, b \to d) + adp^{\oplus}(a, c \to d) + adp^{\oplus}(b, c \to d) + adp^{\oplus}(a, b \to c) \leq 1.$$

Интересным частным случаем этого неравенства может быть d = 0, который позволяет для произвольных a, b, c связать $adp^{\oplus}(a, b \to d)$ с $adp^{\oplus}(0, a \to b)$, $adp^{\oplus}(0, b \to c)$, о которых известно больше.

Предложение 6. Для любого $a \in \mathbb{Z}_2^n$ и $u, v \in \mathbb{Z}_2^n$ таких, что $u + v = 0 \ (mod\ 2^{n-1})$ выполнено:
$$adp^{\oplus}(a, u \to u) = adp^{\oplus}(a, v \to v).$$

Также была посчитана сумма всех $adp^{\oplus}(0, \gamma \to \gamma)$.

Теорема 2. Для любого n выполнено:
$$\sum_{\gamma \in \mathbb{Z}_2^n} adp^{\oplus}(0, \gamma \to \gamma) = 3\left(\frac{3}{2}\right)^{n-2}.$$

Так как явной формулы для $adp^{\oplus}(0, \gamma \to \gamma)$ нет, этот факт может быть использован для некоторых оценок.

3. Эксперименты

Было проверено, что неравенство $adp^{\oplus}(a, \beta \to \gamma) \leq adp^{\oplus}(0, \gamma \to \gamma)$ выполняется для всех $n \leq 9$.

Обозначим через $v(\alpha, \beta, \gamma)$ вектор-столбец длины 8 равный

$$v(\alpha, \beta, \gamma) = A_{4\alpha_{n-1}+2\beta_{n-1}+\gamma_{n-1}} \cdot \dots \cdot A_{4\alpha_0+2\beta_0+\gamma_0} \cdot C,$$

Тогда, согласно теореме 1 из работы [2]

$$\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) = \sum_{i=0}^{n-1} v(\alpha, \beta, \gamma)_i.$$

Обозначим за $M(\alpha, \beta, \gamma)$ максимальную координату $v(\alpha, \beta, \gamma)$:

$$M(\alpha,\beta,\gamma) = \max_{i=0,\dots,n-1} v(\alpha,\beta,\gamma)_i$$

Потребность в величине $M(\alpha,\beta,\gamma)$ возникает при рекуррентном выражении $adp^{\oplus}(a,\beta\to\gamma)$ через $adp^{\oplus}(a',\beta'\to\gamma')$, где $a',\beta',\gamma'\in\mathbb{Z}_2^{n-1}$.

Гипотеза 1. Для всех $\alpha,\beta,\gamma\in\mathbb{Z}_2^n$ справедливо $M(\alpha,\beta,\gamma)\le M(0,\gamma,\gamma)$.

Гипотеза верна при всех $n\le 9$. Один из возможных путей доказательства гипотезы – рекуррентная оценка $M(\alpha,\beta,\gamma)$ через $M(\alpha',\beta',\gamma')$, аналогично $adp^{\oplus}(a,\beta\to\gamma)$. На этом пути используется «второй максимум» $m(\alpha,\beta,\gamma)$:

$$m(\alpha,\beta,\gamma) = \max_{i=0,\dots,n-1,i\neq k} v(\alpha,\beta,\gamma)_i,$$

где k - первая координата с $M(\alpha,\beta,\gamma)=v(\alpha,\beta,\gamma)_k$. К сожалению, следующая гипотеза оказалась неверной в общем случае:

Гипотеза 2. $m(\alpha,\beta,\gamma)\le m(0,\gamma,\gamma)$.

После проведения численных экспериментов были сформулированы некоторые впоследствии доказанные утверждения, а на некоторые не тратилось время. Были проверены на корректность матрицы из работы [2] и создан GUI для подсчета $adp^{\oplus}$ методом S-функций [3].

ЛИТЕРАТУРА

1. E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems, Journal of Cryptology, Vol. 4,pp. 3–72., No. 1, 1991.
2. H. Lipmaa, J. Wall´en, and P. Dumas. On the Additive Differential Probability of Exclusive-Or. In B. K. Roy and W. Meier, editors, FSE, vol. 3017 of Lecture Notes in Computer Science, pp. 317–331. Springer, 2004.
3. Mouha, N., Velichkov, V., De Canniere, C., Preneel, B.: The Differential Analysis of S-Functions. In: ` Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. Lecture Notes in Computer Science, vol. 6544, pp. 36–56. Springer, Heidelberg 2011.

Кураторы исследования:

– Nicky Mouha(USA), PhD, научный сотрудник отдела компьютерной безопасности Национального института стандартов и технологий США (NIST);

– Коломеец Николай Александрович, к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, кафедры параллельного программирования ФИТ НГУ, н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Построение алгебраического описания шифров Simon и Speck

Н.Д. Атутова[1], Д.А. Зюбина[1], С.И. Разенков[2]

[1]Новосибирский государственный университет,
[2]Томский государственный университет

E-mail: **n.atutova@g.nsu.ru, d.zyubina@g.nsu.ru, sirazenkov@stud.tsu.ru**

Первым этапом алгебраического криптоанализа является составление системы уравнений, описывающих преобразования рассматриваемого шифра. В настоящей работе рассматривается задача реализации алгоритма составления системы булевых уравнений, описывающих преобразования LRX-шифра Simon и ARX-шифра Speck.

Ключевые слова: Алгебраический криптоанализ, Speck, Simon, система булевых уравнений

Основная идея алгебраического криптоанализа состоит в составлении сложной системы булевых уравнений, описывающих преобразование шифра. Система строится на основе полностью известного алгоритма шифрования.

$$\begin{cases} p_1 k_2 \oplus p_2 c_2 k_3 = k_4, \\ (p_1 \oplus k_1) \oplus (p_2 \oplus k_2) = c_5, \\ c_8 k_1 k_2 k_4 \oplus p_3 k_3 = 1, \\ \dots \end{cases}$$

Зашифрование на неизвестном криптоаналитику ключе некоторого количества открытых текстов позволяет провести означивание системы — подстановку в уравнения системы битов открытого текста P и шифртекста C.

На следующем этапе осуществляется решение данной системы булевых уравнений с помощью различных методов. Неизвестными в данной системе являются биты ключа - они соответствуют её решению. В силу существования ключа система является непротиворечивой.

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

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

где $\alpha_z \in \mathbb{F}_2$ для любого $z \in F_2^n$(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.

В данной работе рассматривается задача автоматизированного получения алгебраического описания шифров Simon и Speck в форме АНФ. Получаемые системы уравнений в дальнейшем планируется использовать для реализации алгебраических атак на данные шифры. Simon и Speck — семейства блочных шифров, представленные АНБ США в 2013 году [1].

SIMON

Блочный шифр Simon с n-битовым словом и $2n$-битовым блоком обозначается Simon $2n$, где n может принимать значение 16, 24, 32, 48 или 64. Simon $2n$ с m-битовым ключом (длиной mn-бит) будет называться Simon $2n/mn$. Например: Simon 64/128 относится к версии Simon действующей на 64-битных блоках открытого текста и использующей 128-битный ключ. Шифр Simon представляет собой сбалансированную сеть Фейстеля.Алгоритм разработан для простой аппаратной реализации на различных уровнях без потери производительности.

Шифр Simon основан на использовании композиции следующих операций:

1. побитовой операции AND, &
2. побитовой операции XOR, $\oplus$
3. операций циклического сдвига S^j на j бит

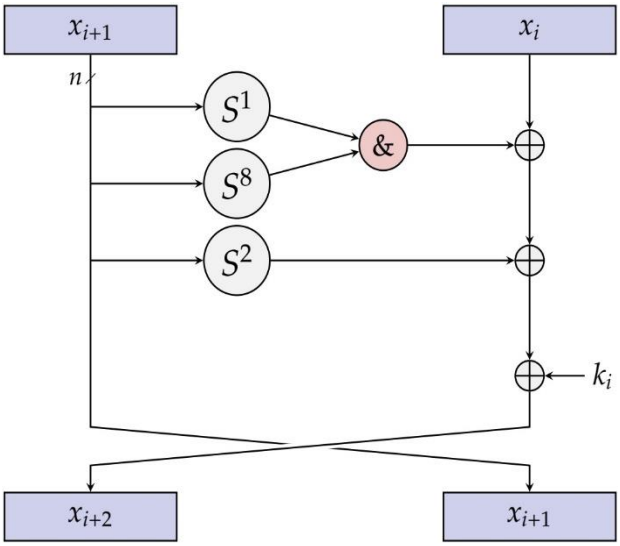


рис.1: Описание раундовых преобразований

block size $2n$	key size mn	word size n	key words m	const seq	rounds T
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

рис.2: Параметры

Задаются первые m ключей, каждый состоит из n бит. Последовательность ключей вычисляется рекуррентно (c — постоянная, а z_j — фиксированная периодическая последовательность):

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+1}, \text{ при } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+2}, \text{ при } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus (S^{-1}) (S^{-3} k_{i+3} \oplus k_{i+1}), \text{ при } m = 4. \end{cases}$$

Составим описание промежуточного раунда. Если на вход i-го раунда поступает блок $x_{i+1}||x_i$, то выходной блок $x_{i+2}||x_{i+1}$ есть

$$x_{i+2} = x_i \oplus F(x_{i+1}) \oplus k_i,$$

где

$$F(x_{i+1}) = (S^1 x_{i+1}) \& (S^8 x_{i+1}) \oplus S^2 x_{i+1}.$$

Введём новую переменную для каждого выхода побитовой операции &, тогда для описания r раундов получим $n(r-2)$ квадратичных уравнений от $n(r-2)+k$ неизвестных.

Для вывода алгебраического описания шифра Simon в форме АНФ была написана программа на языке Python.

Пример:

Simon32/64

$n = 16, m = 4, T = 2$ (T — число раундов)

Открытый текст: (0110010101100101 0110100001110111)

Секретный ключ (k_0, k_1, k_2, k_3) неизвестен:

$k_0 = k_0[0], k_0[1], \ldots, k_0[15]$

$k_1 = k_1[0], k_1[1], \ldots, k_1[15]$

$k_2 = k_2[0], k_2[1], \ldots, k_2[15]$

$k_3 = k_3[0], k_3[1], \ldots, k_3[15]$

При двух раундах шифрования используются только k_0 и k_1.

$\widetilde{x}$ — левая половина шифртекта.

$$\widetilde{x}[0] = k_0[1] + k_0[2] + k_1[0] + k_0[1]k_0[8] + 1$$
$$\widetilde{x}[1] = k_0[3] + k_0[9] + k_1[1] + k_0[2]k_0[9]$$
$$\widetilde{x}[2] = k_0[3] + k_0[4] + k_0[10] + k_1[2] + k_0[3]k_0[10] + 1$$
$$\widetilde{x}[3] = k_0[5] + k_0[11] + k_1[3] + k_0[4]k_0[11] + 1$$
$$\widetilde{x}[4] = k_0[6] + k_0[12] + k_1[4] + k_0[5]k_0[12]$$
$$\widetilde{x}[5] = k_0[7] + k_1[5] + k_0[6]k_0[13]$$
$$\widetilde{x}[6] = k_0[7] + k_0[8] + k_0[14] + k_1[6] + k_0[7]k_0[14]$$
$$\widetilde{x}[7] = k_0[9] + k_0[15] + k_1[7] + k_0[8]k_0[15] + 1$$

$$\widetilde{x}[8] = k_0[9] + k_0[10] + k_1[8] + k_0[0]k_0[9] + 1$$
$$\widetilde{x}[9] = k_0[1] + k_0[11] + k_1[9] + k_0[1]k_0[10] + 1$$
$$\widetilde{x}[10] = k_0[11] + k_0[12] + k_1[10] + k_0[2]k_0[11] + 1$$
$$\widetilde{x}[11] = k_0[12] + k_0[13] + k_1[11] + k_0[3]k_0[12]$$
$$\widetilde{x}[12] = k_0[13] + k_0[14] + k_1[12] + k_0[4]k_0[13] + 1$$
$$\widetilde{x}[13] = k_0[5] + k_0[14] + k_0[15] + k_1[13] + k_0[5]k_0[14]$$
$$\widetilde{x}[14] = k_0[0] + k_1[14] + k_0[6]k_0[15] + 1$$
$$\widetilde{x}[15] = k_0[0] + k_0[1] + k_0[7] + k_1[15] + k_0[0]k_0[7]$$

SPECK

Шифр Speck был разработан, чтобы обеспечить отличную производительность в аппаратном и программном обеспечении. Обозначение для различных вариантов Speck полностью аналогично используемому для Simon Например, Speck 96/144 — размер блока 96 бит и размер ключа 144 бита.

Данный шифр является представителем структуры ARX-шифров.

Основан на использовании операций:

1. побитовых операции XOR, $\oplus$
2. сложения по модулю 2^N, $\boxplus$
3. циклического сдвига $\mathcal{S}^j$ на j бит влево и $\mathcal{S}^{-j}$ на j бит вправо

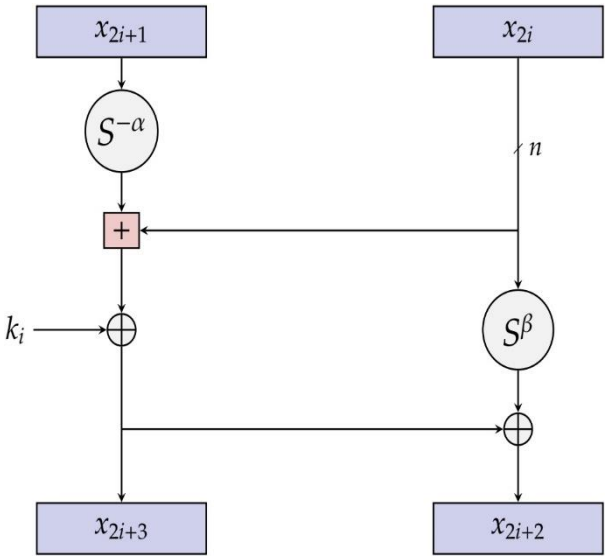


рис.3: Описание раундовых преобразований

block size $2n$	key size mn	word size n	key words m	rot α	rot β	rounds T
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

рис.4: Параметры

Ключевое расписание построено на использовании функции. Обозначим через K ключ шифра Speck $2n$. Можно записать $K = (l_{m-2}, \dots, l_0, k_0)$, где $k_0, l_i \in F_2^n$ и $m \in \{2,3,4\}$.

$$l_{i+m-1} = (k_i \boxplus S^{-\alpha} l_i) \oplus i,$$
$$k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1}.$$

Через k_i обозначается i -й раундовый ключ, $0 \le i \le T$.

Составим расписание промежуточного раунда. Если на вход i –го раунда поступает блок $x_{2i+1}||x_{2i}$, то выходной блок $x_{2i+3}||x_{2i+2}$ есть

$$x_{2i+3} = (S^{-\alpha} x_{2i+1} \boxplus x_{2i}) \oplus k_i,$$
$$x_{2i+2} = S^{\beta} x_{2i} \oplus x_{2i+3}.$$

Для описания раундовой функции необходимо оценить количество переменных для описания нелинейной операции сложения по модулю 2^n. Построим ее алгебраическое описание:

Рассмотрим три вектора

$$(x_{n-1}, x_{n-2}, \dots, x_1, x_0), (y_{n-1}, y_{n-2}, \dots, y_1, y_0), (z_{n-1}, z_{n-2}, \dots, z_1, z_0).$$

Сложение по модулю 2^n

$$(x, y) \mapsto z = x \boxplus y \mod 2^n$$

является отображением, в котором каждый бит выходного результата z_i зависит только от битов x_j, y_j, где $y = 0,1, \dots, i-1$. Данное отображение можно описать с помощью системы, содержащей одно линейное и $n-1$ квадратичных уравнений, зависящих исключительно от переменных из слов x, y и z [3].

$$\begin{cases} z_0 = x_0 \oplus y_0 \\ z_1 = x_1 \oplus y_1 \oplus x_0 y_0 \\ z_2 = x_2 \oplus y_2 \oplus x_1 y_1 \oplus (x_1 \oplus y_1)(x_1 \oplus y_1 \oplus z_1) \\ \vdots \\ z_i = x_i \oplus y_i \oplus x_{i-1} y_{i-1} \oplus (x_{i-1} \oplus y_{i-1})(x_{i-1} \oplus y_{i-1} \oplus z_{i-1}) \\ \vdots \\ z_{n-1} = x_{n-1} \oplus y_{n-1} \oplus x_{n-2} y_{n-2} \oplus (x_{n-2} \oplus y_{n-2})(x_{n-2} \oplus y_{n-2} \oplus z_{n-2}) \end{cases}$$

Равенства из полученной системы целиком определяют функцию сложения по модулю 2^n, но эта система еще не является переопределенной.

Дополнительные квадратичные уравнения можно получить следующими способами:

- $3n$ квадратичных уравнений можно получить путем умножения линейного уравнения из (#) на любые переменные. Но векторное пространство, состоящее из таких уравнений, содержит лишь $3n - 1$ линейно-независимых векторов в силу $(z_0 \oplus x_0 \oplus y_0) = z_0(z_0 \oplus x_0 \oplus y_0) \oplus x_0(z_0 \oplus x_0 \oplus y_0) \oplus y_0(z_0 \oplus x_0 \oplus y_0)$;
- Два уравнения получаются из $z_1 = x_1 \oplus y_1 \oplus x_0 y_0$ при помощи умножения на переменные x_0 и y_0;
- Для $i = \overline{2, n-1}$: $z_i = x_i \oplus y_i \oplus x_{i-1} \oplus y_{i-1} \oplus x_{i-1} y_{i-1} \oplus x_{i-1} z_{i-1} \oplus y_{i-1} z_{i-1}$. Следовательно, домножением по отдельности на $(x_{i-1} \oplus y_{i-1})$, $(x_{i-1} \oplus z_{i-1})$ и $(y_{i-1} \oplus z_{i-1})$ получается система из трёх уравнений ранга 2. Таким образом, имеем дополнительно $2(n-2)$ квадратичных уравнений.

В итоге получаем систему из $6n - 3$ алгебраических уравнений, одно из которых линейное, а остальные имеют степень 2. Такое описание операции сложения по модулю $2n$ будет использовано в ходе алгебраического криптоанализа шифра SPECK.

В рамках дальнейшей работы будут исследованы другие подходы к алгебраическому описанию, например, метод Раддума-Семаева, а также эффективные методы переопределения систем уравнений рассматриваемых шифров.

ЛИТЕРАТУРА

1. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers -- The Simon and Speck families of lightweight block ciphers, NSA Research Directorate, 19 June 2013.Петров П. П. Название книги. М.: Наука,2009.
2. N. Courtois, T. Mourouzis, G. Song, P. Sepehrdad and P. Susil, "Combined algebraic and truncated differential cryptanalysis on reduced-round SIMON," 2014 11th International Conference on Security and Cryptography (SECRYPT), Vienna, 2014, pp. 1--6.
3. N.T. Courtois, B. Debraize (2008) Algebraic Description and Simultaneous Linear

Approximations of Addition in Snow 2.0. In: Chen L., Ryan M.D., Wang G. (eds) Information and Communications Security. ICICS 2008. Lecture Notes in Computer Science, vol 5308. Springer, Berlin, Heidelberg.

Кураторы исследования:

– Агиевич Сергей Валерьевич, к.ф.-м.н., заведующий НИЛ проблем безопасности информационных технологий НИИ прикладных проблем математики и информатики Белорусского государственного университета (г. Минск, Беларусь);

– Александр Владимирович Куценко, аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Алгебраические атаки на шифры Simon и Speck

Д. А. Леонович[1], Е.А. Маро[2], С.Д. Филиппов[3]

[1]Новосибирский государственный Технический университет,
[2]Южный федеральный университет
[3]Санкт-Петербургский государственный университет

E-mail: **marokat@gmail.com, leonovich.d.a13@gmail.com, filippowstepan@yandex.ru**

В данной работе рассматриваются алгебраические атаки на шифры Simon и Speck. Рассмотрены методы атак на основе линеаризации, расширенной линеаризаци (XL) и использовании SAT-решателей для решения систем булевых уравнений, описывающих преобразование шифров Simon и Speck. Изучен алгоритм ElimLin.

Ключевые слова: алгебраический криптоанализ, Speck, Simon, XL-метод, ElimLin, SAT-решатель, системы уравнений

Основная идея алгебраического криптоанализа состоит в составлении сложной системы булевых уравнений, описывающих преобразование шифра. Система строится на основе полностью известного алгоритма шифрования.

$$\begin{cases} p_1 k_2 \oplus p_2 c_2 k_3 = k_4, \\ (p_1 \oplus k_1) \oplus (p_2 \oplus k_2) = c_5, \\ c_8 k_1 k_2 k_4 \oplus p_3 k_3 = 1, \\ \dots . \end{cases}$$

Зашифрование на неизвестном криптоаналитику ключе некоторого количества открытых текстов позволяет провести означивание системы - подстановку в уравнения системы битов открытого текста P и шифртекста C.

На следующем этапе осуществляется решение данной системы булевых уравнений с помощью различных методов. Неизвестными в данной системе являются биты ключа - они соответствуют её решению. В силу существования ключа система является непротиворечивой.

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

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

где $\alpha_z \in \mathbb{F}_2$ для любого $z \in F_2^n$(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.

В данной работе рассматривается задача реализации алгебраических атак на шифры Simon и Speck на основе алгебраического описания, представленного в форме АНФ. Simon и Speck - семейства блочных шифров, представленные АНБ США в 2013 году [1].

При использовании SAT-решателя в программной среде Sage были получены следующие результаты:

Параметры Simon:	Кол-во уравнений	Кол-во неизвестных	Параметры КНФ	Время решения SAT
$T = 3, m = 1$	80	80	96 лит., 432 клоз.	5.09 сек.
$T = 5, m = 2$	128	128	176 лит., 1104 клоз.	45.94 сек.
$T = 7, m = 2$	192	192	272 лит., 1968 клоз.	более 2 часов[2]
$T = 8, m = 2$	224	224	320 лит., 2400 клоз.	–

Методы, основанные на линеаризации

Идея линеаризации: каждый моном — отдельная переменная. Пусть d - максимальная степень уравнений системы. Выбирается число $D > d$, как правило $D = d + 1$ или $D = d + 2$. После этого каждое уравнение умножается на каждый возможный моном степени не более $D - d$. К исходной системе, дополненной новыми уравнениями, применяется алгоритм линеаризации.

Если n' — число переменных после линеаризации, то достаточно большом ранге ($r \approx n'$)система имеет либо одно решение, либо малое количество решений. Для каждого из получаемых решений линеаризованной системы проводится проверка, является ли оно решением исходной системы.

XL-алгоритм (eXtended Linearization):

на вход поступает система из m уравнений от n неизвестных, имеющая степень d.

на выходе решение/-я системы уравнений, если система имеет достаточный ранг.

- выбирается число $D > d$;
- составляется список L всех мономов, которые имеют степень не более $D - d$ (включая 1);
- все уравнения системы домножаются на каждый элемент списка L (количество уравнений составит $m|L|$);
- полученная система линеаризуется и решается.

Будем использовать обозначения: T — число раундов, m — число ключей, задаваемых явным образом. Целью взлома является отыскание всех битов первых m ключей. Количество неизвестных в таблице соответствует количеству мономов после линеаризации.

Результаты XL- метода для шифра Simon.

Метод решения	Параметры Simon	Кол-во уравнений	Кол-во неизвестных	Количество решений
Линеаризация	$T = 3, m = 1$	48	48	4, только 1 подходит
Расширенная линеаризация (XL)	$T = 3, m = 1$	1584	992	1, только 1 подходит
Линеаризация	$T = 4, m = 1$	64	80	65536
Расширенная линеаризация (XL)	$T = 4, m = 1$	3136	2616	3256, только 1 подходит.

Результаты XL-метода для шифра Speck.

Метод решения	Параметры Speck	Кол-во уравнений	Кол-во неизвестных	Количество решений
Линеаризация	$T = 3, m = 1$	532	242	—
Расширенная линеаризация (XL)	$T = 3, m = 1$	129276	242	—

Как видно из таблицы получаемая система уравнений имеет большое количество решений. Таким образом, XL-метод для шифра Speck не дал результатов.

ElimLin - еще один вид атаки, основанный на линеаризации. Этот метод эффективен для решения системы уравнений небольшой степени (до четвертой). Однако заменой переменных можно свести нелинейные уравнения к квадратичным. Так как в результате введения новых переменных для шифров Simon и Speck были получены системы уравнений не выше 2 степени, ElimLin может быть эффективен [3].

Для нахождения линейной оболочки системы используется приведение к ступенчатому виду методом Гаусса.

Основные шаги ElimLin:

- ищутся все линейные уравнения в линейной оболочке уравнений системы. Они являются элементами пересечения двух векторных пространств: пространства мономов степени 1 и линейной оболочки уравнений исходной системы;
- переменные линейных уравнений поочерёдно выражаются через остальные члены, пока в системе не останется линейных уравнений. Система, получаемая в результате работы алгоритма, вероятно имеет меньшее число уравнений.

Известно (см., например, [2]), что все линейные уравнения из линейной оболочки системы полиномиальных уравнений S могут быть найдены в линейной оболочке линейных уравнений, найденных после выполнения процедуры приведения матрицы к ступенчатому виду методом исключения Гаусса, примененной к системе. Следствием данного факта является то, что линейные уравнения, найденные на итерациях алгоритма ElimLin, примененного к исходной системе квадратичных уравнений, образуют базис множества всех линейных уравнений из линейной оболочки данного множества уравнений.

В рамках дальнейшей работе планируется исследовать другие методы алгебраических атак, например, алгоритм Бухбергера, а также реализовать атаки с использованием метода ElimLin.

ЛИТЕРАТУРА

1. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers — The Simon and Speck families of lightweight block ciphers, NSA Research Directorate, 19 June 2013.
2. Bard G., Algebraic Cryptanalysis, 356 p., Springer (2009).
3. N. Courtois, T. Mourouzis, G. Song, P. Sepehrdad and P. Susil, "Combined algebraic and truncated differential cryptanalysis on reduced-round SIMON," 2014 11th International Conference on Security and Cryptography (SECRYPT), Vienna, 2014, pp. 1—6.

Кураторы исследования:

– Агиевич Сергей Валерьевич, к.ф.-м.н., заведующий НИЛ проблем безопасности информационных технологий НИИ прикладных проблем математики и информатики Белорусского государственного университета (г. Минск, Беларусь);

– Александр Владимирович Куценко, аспирант ММФ НГУ, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

КРИПТОАНАЛИЗ СИСТЕМ С ОТКРЫТЫМ КЛЮЧОМ

Оптимизация параметров алгоритма Полларда p-1

К. В. Натарова

Московский физико-технический институт

E-mail: **natarova.kv@phystech.edu**

В работе представлено исследование зависимости оптимизации работы метода Полларда p-1 от методов вычисления степени и начального значения постоянных параметров алгоритма. Реализован программный код нескольких вариаций подбора критериев.

Ключевые слова: (p − 1)-алгоритм Полларда, факторизация чисел

Поставленная задача. Изучить принцип работы метода Полларда p-1, эффективность его работы в зависимости от изменения параметров. Реализовать уже исследованные вариации метода Полларда p-1, сравнить полученные результаты с результатами исследования авторского алгоритма.

В криптографии и шифровании важную роль играет факторизация больших чисел, и метод Полларда p-1 является одним из вариантов решения задачи их разложения на простые множители.

Алгоритм Полларда факторизации числа n заключается в следующем [1]:

1. Выбор произвольного числа k.
2. Выбор произвольного числа a, $1 < a < n$.
3. Вычисление d = НОД (a, n). Если $d > 1$, мы получили нетривиальный делитель числа n. Если $d = 1$, переходим к шагу 4.
4. Вычисляем $D = (a^k - 1, n)$ и если $1 < D < n$, то D является делителем n. Если $D = 1$, возвращаемся к шагу 1, а при D = N к шагу 2 и выбираем новое a.

Проанализирована статья А. С. Климиной «Оптимизация (p − 1)-алгоритма Полларда» [2] и измерена скорость работы предложенных в ней алгоритмов для вычисления степени k:

1. k — факториал некоторого числа
2. k — произведение одинаковых степеней простых чисел

Дополнительно проверен авторский вариант нахождения:

1. k – произведение степеней простых чисел, зависящих от количества множителей

Авторский алгоритм. $k = p^m_1 p^{m-1}_2 \ldots p^1_m$, где $p_1, \ldots, p_m$ — первые m простых.

Результаты. Так как метод А. С. Климиной для вычисления одного из параметров для работы метода требует возведения в очень большие степени, это наложило ограничения на размер чисел, с которыми можно было работать. Среднее время работы с десятизначными числами для способов 1 и 2 составило 0.010240 и 0.000211 секунд соответственно, для авторского метода – 0.000164 секунд. Среднее время работы с пятнадцатизначными числами для способов 1 и 2 составило 0.010260 и 0.000196 секунд соответственно, для авторского метода – 0.000196 секунд. Среднее время работы с двадцатизначными числами для способов 1 и 2 составило 0.065252 и 0.000307 секунд соответственно, для авторского метода – 0.000343

секунд. В каждом из случаев не поддалось разложению примерно 60% чисел. Метод находил один из простых множителей от каждого числа, необязательно являвшегося произведением двух простых.

Планы на будущие исследования. Оптимизировать код для нахождения разложения большего количества чисел с большей разрядностью, переработать структуру программы для работы только с произведениями двух простых чисел.

ЛИТЕРАТУРА

1. Климина А.С., Жданов О.Н. Оптимизация выбора параметров для алгоритма Полларда // Актуальные проблемы авиации и космонавтики. 2013. №9.
2. А. С. Климина, Оптимизация (p−1)-алгоритма Полларда, ПДМ. Приложение, 2013, выпуск 6, 118–119

Кураторы исследования:
— Токарева Наталья Николаевна, к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research;
— Облаухов Алексей Константинович, аспирант ИМ СО РАН, ассистент кафедры теоретической кибернетики ММФ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Анализ базовой версии криптосистемы с открытым ключом, основанной на сложности решения систем уравнений в конечных полях

Е.В. Завалишина, Д.Р. Парфенов

Новосибирский государственный университет

E-mail: **e.zavalishina@g.nsu.ru, d.parfenov@g.nsu.ru**

В данной работе представлены полученные за время проведения летней школы-конференции «Криптография и информационная безопасность» данные криптоанализа базовой версии системы с открытым ключом, основанной на сложности решения систем уравнений в конечных полях, являющейся перспективной для постквантовой криптографии. Авторам удалось установить точный размер ключевого пространства, общий вид уравнений, а также размер системы сравнений, что в будущем позволит построить оценку количества эквивалентных ключей.

Ключевые слова: открытый ключ, криптоанализ, системы полиномиальных уравнений в конечных полях, кандидаты для постквантовой криптографии.

В 2016 году the National Institute of Standards and Technology представил доклад под названием Report on Post-Quantum Cryptography [1], в котором полагает, что пришло время подготовиться к переходу на квантово-устойчивую криптографию, так как некоторые задачи, лежащие в основе использующихся на практике криптографических алгоритмов, могут быть решены квантовыми компьютерами.

В связи с этим одним из авторов настоящей работы и соавторами была предпринята попытка создать новый алгоритм шифрования данных с открытым ключом, основанный на решении системы однородных полиномиальных уравнений в целых числах [2]. Кратко опишем основной принцип работы.

Имеем исходный текст $P = (P_1, \dots, P_n)^T$, где $0 < P_i < p - 1$.

Пусть $K_{priv} = \{A, B\},\ K_{pub} = \{f(x), p\}$, где

- p – целочисленный простой модуль, такой что $\varphi(p)$ взаимно просто с 3;
- A и B – обратимые $n \times n$ матрицы над $GF(p)$;
- $f(x) = \left(f_1(x), \dots, f_n(x)\right)^T$, где $f_i(x)$ – полином от переменных $x = (x_1, \dots, x_n)^T$, вычисленный в три шага по модулю p:
  1) $u(x) = A \times x$;
  2) $s(x) = \left(\left(u_1(x)\right)^3, \dots, \left(u_n(x)\right)^3\right)$;
  3) $f(x) = B \times s(x)$.

Шифртекст $C = (C_1, \dots, C_n)^T$ вычисляется как $C = f(P_1, \dots, P_n)$. Расшифрование производится обратными преобразованиями.

Авторами работы было выявлено, что существуют случаи, когда из различных секретных ключей появляются одинаковые открытые ключи – эквивалентные ключи. Ключи называются эквивалентными, если при зашифровании любого открытого текста,

шифртексты совпадают.

В первую очередь необходимо оценить размер пространства ключей. С этим связано следующее утверждение.

Утверждение 1. Пусть n – выбранный размер матрицы, тогда размер ключевого пространства равен $N = \left((p^n - 1)(p^n - p) \dots (p^n - p^{n-1})\right)^2$.

Доказательство. Секретный ключ однозначно определяет открытый ключ и состоит из двух обратимых $n \times n$ матриц над $GF(p)$. Матрица обратима тогда и только тогда, когда её определитель не равен нулю. Определитель матрицы не равен нулю тогда и только тогда, когда строки матрицы являются линейно независимыми. Это позволяет подсчитать количество обратимых $n \times n$ матриц над $GF(p)$ следующим образом.

В качестве первой строки матрицы мы можем взять любую ненулевую строку длины n – таких строк в точности $p^n - 1$. Далее, при выборе второй строки, мы выбираем уже из оставшихся строк, линейно независимых по отношению к первой. Количество таких строк $p^n - p$. Наконец, при выборе n-ой строки мы выбираем одну из $p^n - p^{n-1}$.

Таким образом, количество обратимых $n \times n$ матриц над $GF(p)$ равно

$$(p^n - 1)(p^n - p) \dots (p^n - p^{n-1}).$$

Поскольку две матрицы выбираются независимо друг от друга, размер ключевого пространства равен квадрату этого числа:

$$N = \left((p^n - 1)(p^n - p) \dots (p^n - p^{n-1})\right)^2. \blacksquare$$

Как мы можем видеть по таблице 1, значение двоичного логарифма количества различных ключей растёт и достигает достаточно больших значений уже при малых n и p. Отсюда следует, что перебор секретных ключей является сложной задачей.

p	n	$[log_2 N]$
5	2	17
5	3	41
11	3	61
11	4	110
11	5	172
17	3	73
17	4	130
23	10	904

Таблица. 1. Размер ключевого пространства относительно количества переменных для некоторых значений p и n

Набор полиномов, использующийся в качестве открытого ключа в базовой версии, имеет строго определенную структуру. Следующее утверждение описывает общий вид такого ключа.

Утверждение 2. Пусть $A = \left(a_{ij}\right)$ и $B = \left(b_{ij}\right)$ – обратимые $n \times n$ матрицы над $GF(p)$. Вектор полиномов $f(x_1, \dots, x_n)$ – открытый ключ криптосистемы. Тогда k-ая строка вектора открытого ключа имеет следующий вид для любых $i = 1 \dots n,\ j = 1 \dots n,$ $t = 1 \dots n,\ q = 1 \dots n$:

$$f_k(x_1, \dots, x_n) = \sum_i x_i^3 \propto_{ki} + \sum_{i \neq j} x_i^2 x_j \beta_{kij} + \sum_{i<j<t} x_i x_j x_t \gamma_{kijt} \ (mod\ p),$$

$$\propto_{ki} = \sum_j a_{ji}^3 b_{kj} \ (mod\ p),$$

$$\beta_{kij} = \sum_t 3a_{ti}^2 a_{tj} b_{kt} \ (mod\ p),$$

$$\gamma_{kijt} = \sum_q 6a_{qi} a_{qj} a_{qt} b_{kq} \ (mod\ p).$$

Утверждение верно для любых n и p, удовлетворяющих условиям построения криптосистемы.

Доказательство. По построению открытого ключа.

На первом шаге после $u(x) = A \times$ x имеем $u_i(x) = \sum_{j=1}^n a_{ij} x_j$

Далее вычисляем $s(x) = \left(\left(u_1(x)\right)^3, \dots, \left(u_n(x)\right)^3\right)$. При этом $s_i(x) = \left(u_i(x)\right)^3 = = \left(\sum_{j=1}^n a_{ij} x_j\right)^3 = \sum_{j=1}^n a_{ij}^3 x_j^3 + \sum_{k \neq j} 3a_{ij}^2 x_j^2 a_{ik} x_{ik} + \sum_{k<j<t} 6a_{ij} x_{ij} a_{ik} x_{ik} a_{it} x_{it}$.

Далее получаем $f(x) = B \times s(x)$, откуда $f_k(x) = \sum_{j=1}^n b_{kj} s_j(x) = \sum_i (x_i^3 \sum_j a_{ji}^3 b_{kj}) + + \sum_{i \neq j} (x_i^2 x_j \sum_t 3a_{ti}^2 a_{tj} b_{kt}) + \sum_{i<j<t} (x_i x_j x_t \sum_q 6a_{qi} a_{qj} a_{qt} b_{kq})$. ■

Утверждение 3. Набор решений системы сравнений вида

$$\sum_j a_{ji}^3 b_{kj} = \propto_{ki} \ (mod\ p),$$

$$\sum_t 3a_{ti}^2 a_{tj} b_{kt} = \beta_{kij} (mod\ p),$$

$$\sum_q 6a_{qi} a_{qj} a_{qt} b_{kq} = \gamma_{kijt} (mod\ p)\ .$$

является набором эквивалентных ключей криптосистемы, где a_{ij} и b_{ij} – элементы обратимых $n \times n$ над $GF(p)$ матриц A и B, $i = 1 \dots n, j = 1 \dots n, t = 1 \dots n, q = 1 \dots n$. Утверждение верно для любых n и p, удовлетворяющих условиям построения криптосистемы.

Доказательство. Пусть даны две отличные друг от друга пары матриц: A, B и A', B' такие, что их элементы a_{ij}, b_{ij} и a'_{ij}, b'_{ij} при заданных $\propto_{ki}$, $\beta_{kij}, \gamma_{kijt}$ являются решениями системы сравнений. В этом случае полученные открытые ключи совпадают по построению $f(x) = f'(x)$, а значит, являются эквивалентными. ■

Утверждение 4. Пусть n – количество переменных, тогда число сравнений в системе из утверждения 3 равно $n(n + 2C_n^2 + C_n^3)$. Утверждение верно для любых n и p, удовлетворяющих условиям построения криптосистемы. Количество сравнений системы не зависит от модуля.

Доказательство. Следует из общего вида k-ой строки матрицы открытого ключа: в каждой строке будет n коэффициентов $\propto_{ki}$, $2C_n^2$ коэффициентов β_{kij} и C_n^3 коэффициентов γ_{kijt}.

Таким образом, на каждую переменную приходится по $(n + 2C_n^2 + C_n^3)$ сравнений, а поскольку всего n переменных, то в системе будет $n(n + 2C_n^2 + C_n^3)$ сравнений.

■

Также были получены некоторые экспериментальные данные путём перебора всех возможных секретных ключей для $n = 2$.

На графике (рис.1) и таблице 2 видно, что количество эквивалентных ключей для каждого открытого ключа растёт гораздо медленнее, чем общее количество ключей, откуда можно сделать вывод, что при увеличении простого модуля доля эквивалентных ключей постоянно уменьшается.

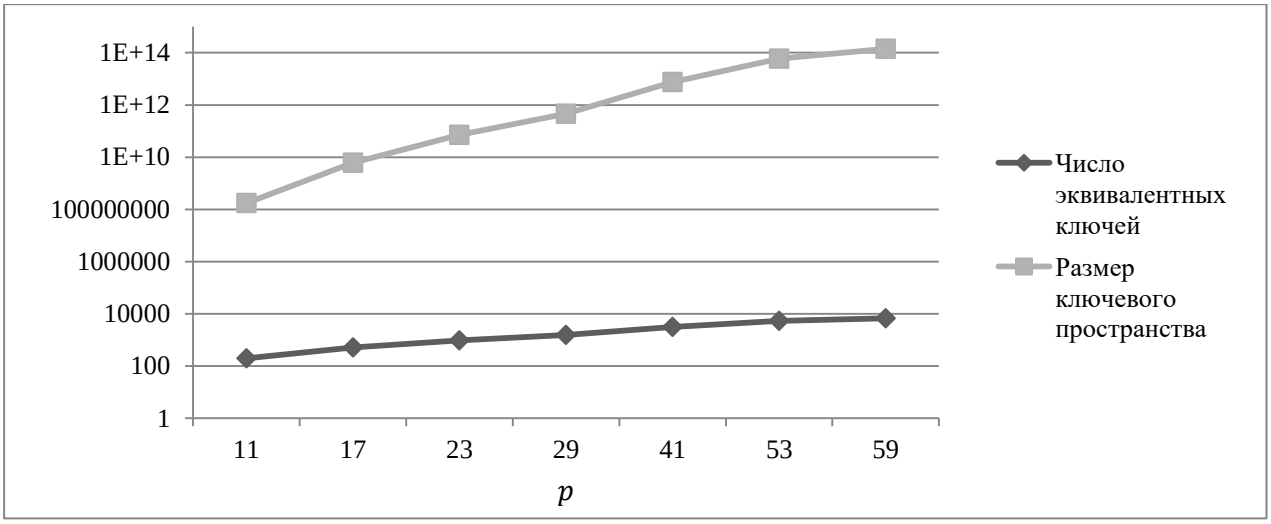


Рис.1

n	m	K_{eq}	K_{total}	K_{eq}/K_{total}
2	11	200	174240000	1,14784E-06
2	17	512	6136528896	8,34348E-08
2	23	968	71378740224	1,35615E-08
2	29	1568	465233126400	3,37035E-09
2	41	3200	7591127040000	4,21545E-10
2	53	5408	59889768367104	9,02992E-11
2	59	6728	141813801273600	4,74425E-11

Таблица 2

ЛИТЕРАТУРА

1. National Institute of Standards and Technology NIST Internal or Interagency Reports (IR) 8105 Report on Post-Quantum Cryptography, Gaithersburg, Maryland, April 2016, 15pp.
2. Волков, Е., Баранов А., Завалишина Е. Криптографическая система с открытым ключом Second Conference on Software Engineering and Information Management (SEIM-2017), 2017. С. 41-44.

Куратор исследования – к.ф.-м.н. доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН, зав. лаб. криптографии JetBrains Research, Токарева Наталья Николаевна.

ПОСТКВАНТОВАЯ КРИПТОГРАФИЯ

Постквантовая криптография: NTRU

А. О. Бахарев[1], А.П. Горяйнова[2]

[1]Новосибирский государственный университет,
[2]Тюменский государственный университет

E-mail: **a.bakharev@g.nsu.ru, gor@72.ru**

В силу существования квантовых алгоритмов из класса BQP, решающих задачи факторизации и дискретного логарифмирования, современные криптосистемы, основанные на сложности решения данных задач, станут менее эффективными при появлении квантовых компьютеров с достаточным количеством кубитов. В настоящей работе рассмотрена NTRU - криптосистема с открытым ключом, основанная на теории решеток. Реализованы алгоритмы генерации ключей, зашифрования и расшифрования. Изучена атака, основанная на построении решетки и последующем поиске в ней вектора с малой нормой. Атака реализована для малых значений параметров.

Ключевые слова: постквантовая криптография, теория решеток, криптография с открытым ключом

Криптосистема NTRU зависит от трех целочисленных параметров (N, p, q) и четырех множеств $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\phi, \mathcal{L}_m$ из полиномов степени $N-1$ с целочисленными коэффициентами. Числа p и q не обязательно должны быть простыми, но будем считать, что $gcd(p,q) = 1$, а q всегда будет значительно больше p. Мы работаем в кольце $R = Z[X]/(X^N - 1)$. Элемент $F \in R$ будет записан как многочлен или вектор,

$$F = \sum_{i=0}^{N-1} F_i x^i = [F_0, F_1, \dots, F_{N-1}].$$

Мы пишем $\circledast$ для обозначения умножения в R. Эта операция явно определяется как результат циклической свертки,

$$F \circledast G = H, H_k = \sum_{i=0}^{k} F_i G_{k-i} + \sum_{i=k+1}^{N-1} F_i G_{N+k-i} = \sum_{i+j \equiv k (mod\ N)} F_i G_j.$$

Когда выполняем умножение по модулю (скажем) q, то имеем в виду взятие коэффициентов по модулю q.

Множество сообщений $\mathcal{L}_m$ состоит из таких многочленов кольца R, у которых коэффициенты лежат в интервале $\left(-\frac{p-1}{2}, \frac{p-1}{2}\right)$. Определим множество $\mathcal{L}(d_1, d_2)$, как множество многочленов, у которых ровно d_1 коэффициентов равны 1, d_2 коэффициентов равны -1, а оставшаяся часть равна 0. Тогда $\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1)$, $\mathcal{L}_g = \mathcal{L}(d_g, d_g)$, $\mathcal{L}_\phi = \mathcal{L}(d, d)$, для параметров d_f, d_g, d. При этом центрированная L^2 норма многочленов $f \in \mathcal{L}_f$ имеет вид $|f|_2 = \sqrt{2d_f - 1 - N^{-1}}$, для $g \in \mathcal{L}_g$ она составит $|g|_2 = \sqrt{2d_g}$, и для $\phi \in \mathcal{L}_\phi$ примет вид $|\phi|_2 = \sqrt{2d}$.

1. Генерация ключей.

Для создания ключа NTRU Дэн (расшифровальщик) случайным образом выбирает 2 полинома $f, g \in \mathcal{L}_g$. Полином f должен удовлетворять дополнительному требованию, чтобы у него были инверсы по модулю q и по модулю p. В случае подходящего выбора параметров, это будет верно для большинства вариантов выбора f, а реальное вычисление этих инверсов легко осуществить с помощью модификации евклидового алгоритма. Обозначим обратные к ним через F_q и F_p, т.е.

$$F_q \circledast f \equiv 1(mod\ q) \text{ и } F_p \circledast f \equiv 1(mod\ p). \quad (1)$$

Далее находим многочлен:

$$h \equiv F_q \circledast g(mod\ q) \quad (2)$$

Открытый ключ Дэна - многочлен h. Секретный ключ Дэна - многочлен f.

2. Зашифрование.

Предположим, что Кэти (шифровальщик) хочет послать сообщение Дэну (расшифровальщику). Она начинает с выбора сообщения m из набора текстов $\mathcal{L}_m$. Затем она случайным образом выбирает многочлен $\phi \in \mathcal{L}_\phi$и использует открытый ключ h Дэна для вычисления

$$e \equiv p\phi \circledast h + m\ (mod\ q).$$

Это зашифрованное сообщение, которое Кэти передает Дэну.

3. Расшифрование.

Предположим, что Дэн получил сообщение от Кэти и хочет расшифровать его, используя свой закрытый ключ f. Чтобы сделать это эффективно, Дэн должен был предварительно вычислить многочлен F_p, описанный в Разделе 1.

Чтобы расшифровать e, Дэн сначала вычисляет

$$a \equiv f \circledast e(mod\ q),$$

где он выбирает коэффициенты a в интервале от $-q/2$ до $q/2$. Теперь, рассматривая a как многочлен с целочисленными коэффициентами, Дэн восстанавливает сообщение с помощью вычисления

$$F_p \circledast a(mod\ p).$$

Примечание. Для соответствующих значений параметров существует чрезвычайно высокая вероятность того, что процедура расшифрования восстановит исходное сообщение. Однако, некоторые варианты параметров могут вызывать периодический сбой в расшифровании, поэтому, вероятно, следует включать несколько контрольных битов в каждый блок сообщения. Частой причиной сбоя расшифрования будет то, что сообщение неправильно центрировано, в этом случае Дэн сможет восстановить сообщение, выбрав коэффициенты $a \equiv f \circledast e(mod\ q)$ в несколько другом интервале, например, от $-q/2 + x$ до $q/2 + x$ для некоторого небольшого (положительного или отрицательного) значения x. Если значение x не работает, то мы говорим, что у нас есть пробел-пауза и сообщение не может быть расшифровано так же легко. Для правильно подобранных значений параметров это будет происходить так редко, что на практике может быть проигнорировано.

Атака на основе решеток

Напомним несколько фактов о решетках. Пусть $u_1, \dots, u_n \in \mathbb{R}^m$ линейно независимые вектора и $n \leq m$. Решётка L, натянутая на $(u_1, \dots, u_n)$, состоит из всех целочисленных линейных комбинаций $u_1, \dots, u_n$, то есть:

$$L = \mathbb{Z}u_1 \bigoplus \dots \mathbb{Z}u_n = \left\{ \sum_{i=1}^{n} b_i u_i \mid b_i \in \mathbb{Z} \right\}.$$

Множество $(u_1, \dots, u_n)$ называется базисом решётки. Решётка может быть удобно представлена матрицей B, у которой на месте строк стоят векторы $u_1, \dots, u_n$.

Есть несколько естественных вычислительных проблем, связанных с решёткам. Важной проблемой является нахождение кратчайшего вектора: учитывая базовую матрицу B для L, вычислить ненулевой вектор $v \in L$ такой, что v минимален. Теоретически, наименьший вектор может быть найден при помощи полного перебора, но на практике это займёт очень много времени. В 1982 году Lenstra, Lenstra и Lovasz представили Алгоритм LLL. Алгоритм LLL может найти относительно маленькие вектора за полиномиальное время, но даже LLL занимает много времени.

Шаги LLL:

1. Сначала создается копия исходного базиса, которая при помощи метода Грама-Шмидта ортогонализуется для того, чтобы по ходу алгоритма текущий базис сравнивался с полученной копией на предмет ортогональности. $(b_1^*,\ b_2^* \dots, b_d^*)$.
2. Если какой-либо коэффициент Грама — Шмидта $\left|\mu_{k,j}\right| (c\ j < k)$, где
$$\mu_{k,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle},$$
по модулю больше 0,5, то это говорит о том, что необходимо вычесть из вектора b_k вектор b_j домноженный на округленный $\mu_{k,j}$ (округление нужно, так как вектор решётки остается вектором этой же решётки только при умножении на целое число, что следует из её определения).
3. Пересчитывается $\mu_{k,j}$ для $j < k$.
4. Для b_k проверяется $\left(\delta - \mu_{k,k-1}^2\right)||b_{k-1}^*||^2 \leq ||b_k^*||^2$. Если условие не выполнено, то индексы проверяемых векторов меняются местами. И условие проверяется снова для того же вектора (уже с новым индексом).
5. При помощи ортогонализации Грама-Шмидта пересчитывается копия исходного базиса $(b_1^*,\ b_2^* \dots, b_d^*)$ и пересчитываются $\mu_{k-1,j}$ и $\mu_{k,j}$ для $j < k$.
6. Если остался какой-либо $\mu_{k,j}$ по модулю превышающий 0,5, то надо вернуться к пункту 2.
7. Когда все условия проверены и сделаны изменения, алгоритм завершает работу.

Атака на ключ.

Рассмотрим матрицу $2N$ на $2N$, состоящую из четырех блоков N на N:

$$\begin{pmatrix} \alpha & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{N-1} \\ 0 & \alpha & \dots & 0 & h_{N-1} & h_0 & \dots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha & h_1 & h_2 & \dots & h_0 \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{pmatrix}$$

Пусть L - решётка, порожденная этой матрицей. Поскольку открытый ключ имеет вид $h \equiv F_q \circledast g(mod\ q)$, решетка L будет содержать вектор $r = (\alpha f, g)$, где параметр $\alpha = |g|_2/|f|_2$.

С помощью алгоритма LLL была реализована атака на криптосистему NTRU при малых значениях параметров. Были рассмотрены два набора: $p = 3,\ q = 61,\ N = 5,\ df = 3,\ dg = 2,\ d = 2$ и $p = 3,\ q = 61,\ N = 11,\ df = 4,\ dg = 4,\ d = 4$. Анализ векторов в преобразованном (кратчайшем) базисе позволил в некоторых случаях найти секретный ключ.

ЛИТЕРАТУРА

1. J. Hoffstein, J. Pipher, J.H. Silverman NTRU: A new high speed public key cryptosystem, Preprint; presented at the rump session of Crypto 96.
2. Schnorr, C.P., Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66, 181–199 (1994).
3. Hoffstein J., Pipher J., Silverman J.H. NTRU: A ring-based public key cryptosystem. Algorithmic Number Theory.1998, 267-288.
4. Abderrahmane Nitaj Cryptanalysis of NTRU with two public keys. Cryptology ePrint Archive.2011/477.
5. J. Hoffstein, J.H. Silverman, W. Whyte NTRU cryptosystem technical report #12, version 2: Estimated breaking times for NTRU lattice.

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

Постквантовая криптография: NTS-KEM

Е.В. Семенова[1], И.Д. Трацевский[2]

[1]Томский государственный университет,
[2]Тюменский государственный университет

E-mail: **seemenkina@gmail.com, wipelayer@yandex.ru**

Появление универсального квантового компьютера достаточной разрядности несет угрозу многим современным криптосистемам, основанным на сложности задач факторизации и дискретного логарифмирования. Тем не менее, существуют классы вычислительно-трудных задач, которые потенциально являются труднорешаемыми даже с использованием квантовых компьютеров. Криптосистемы, стойкость которых основана на сложности решения таких задач, называются постквантовыми. В рамках данного проекта рассмотрена криптосистема NTS-KEM, которая основана на NP-трудной задаче декодирования полных линейных кодов. В качестве практической задачи реализован один из вариантов криптосистемы NTS-KEM(12,64) на языке Golang.

Ключевые слова: постквантовая криптография, коды Гоппы, механизм инкапсуляции ключа

В рамках проекта «Постквантовая криптография» были поставлены следующие задачи:

1. Изучить один из алгоритмов постквантовой криптографии, участвующий в конкурсе NIST.
2. Реализовать данный алгоритм на языке программирования Golang.
3. Провести эксперименты с полученной реализацией, а также сравнить работу полученной реализации с эталонной реализацией авторов выбранного алгоритма.

Алгоритм инкапсуляции ключа NTS-KEM [1], основанный на кодах, представлен на конкурсе NIST по выбору нового стандарта постквантовых алгоритмов [2]. Другими представителями “криптографии на кодах” является McEliece и Niederreiter [3, 4]. Несмотря на то, что NTS-KEM представляет собой криптографию с открытым ключом, алгоритм нацелен на инкапсуляцию ключа.

Определение 1. Механизм инкапсуляции ключа (далее будет использовано обозначение KEM - Key encapsulation mechanism) состоит из трех алгоритмов:

1. Алгоритм генерации случайного ключа (далее KGen) преобразует входное секретное значение k в пару (pk, sk) открытого и закрытого ключа соответственно.
2. Алгоритм инкапсуляции ключа (далее Encap) с использованием открытого ключа pk возвращает пару (C, K) шифртекста и инкапсулированного ключа.
3. Алгоритм декапсуляции ключа (далее Decap) из закрытого ключа sk и шифртекста C вычисляет значение инкапсулированного ключа K.

Где может применяться KEM? Как известно, криптография с открытым ключом часто используется для передачи секретного ключа, в дальнейшем используемого в симметричной криптосистеме. Этот ключ распределяется между отправителем и адресатом, а затем с его помощью с использованием симметричной криптосистемы осуществляется зашифрование большого массива данных.

Для более подробного описания системы введём некоторые определения из алгебры и теории кодирования [5].

Пусть F_2 - конечное поле из двух элементов, а F_2^n - векторное пространство векторов длины n над полем F_2 . Весом Хэмминга вектора $x \in F_2^n$ называется число его координат, отличных от нуля. Двоичным линейным кодом длины n называется произвольное подпространство пространства F_2^n. Кодовым расстоянием d кода C называется минимальное расстояние между двумя различными кодовыми словами данного кода. Количество ошибок, исправляемое линейным кодом с кодовым расстоянием d, вычисляется как $t = \lfloor (d-1)/2 \rfloor$. Двоичный линейный код длины n, имеющий размерность k и кодовое расстояние d, будем называть $[n, k, d]_2$-кодом.

Пусть C - линейный $[n, k, d]_2$ код. Матрица $\boldsymbol{G} \in F_2^{k \times n}$, строки которой образуют базис кода C, называется порождающей. Если $v \in F_2^k$, то $c = v \cdot \boldsymbol{G} \in F_2^n$ является кодовым словом в коде C. Проверочной матрицей кода C называется матрица $\boldsymbol{H} \in F_2^{(n-k)\times n}$ такая, что $x \cdot \boldsymbol{H}^T = \mathbf{0}$ для каждого $x \in C$. Для проверочной матрицы справедливо соотношение $\mathbf{GH}^{\mathrm{T}}$=0. Синдромом вектора $v \in F_2^n$ называется вектор $s = v \cdot \boldsymbol{H}^T \in F_2^{n-k}$.

Определение 2. [1] Двоичным кодом Гоппы C_G называется класс линейных $[n,k,d]_2$ кодов, который описывается многочленом Гоппы $G(z) = g_0 + g_1 z + \cdots + g_{t-1} z^{t-1} \in F_{2^m}[z]$ и имеет минимальное кодовое расстояние d=2t+1.

Рассматриваемый в [1] многочлен Гоппы G(z) обладает следующими свойствами: он не имеет корней в поле F_{2^m} (подразумевается $n = 2^m$) и не имеет кратных корней в любом расширении поля, что гарантирует исправление t ошибок кодом C_G. В этом случае код Гоппы имеет следующие параметры: $[n,n-mt,2t+1]_2$.

Пусть $\{a_0,a_1,\ldots,a_{n-1}\}$ - все элементы поля F_{2^m}. Проверочная матрица $\boldsymbol{H}_m \in F_{2^m}^{t\times n}$кода C_G строится с использованием $G^2(z)$ как показано на (Рис.1).

$$\mathbf{H}_m = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0^{\tau-1} & a_1^{\tau-1} & a_2^{\tau-1} & \cdots & a_{n-1}^{\tau-1} \end{bmatrix} \cdot \begin{bmatrix} G(a_0)^{-2} & 0 & 0 & \cdots & 0 \\ 0 & G(a_1)^{-2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & G(a_{n-1})^{-2} \end{bmatrix}$$
$$= \begin{bmatrix} G(a_0)^{-2} & G(a_1)^{-2} & G(a_2)^{-2} & \cdots & G(a_{n-1})^{-2} \\ a_0 G(a_0)^{-2} & a_1 G(a_1)^{-2} & a_2 G(a_2)^{-2} & \cdots & a_{n-1} G(a_{n-1})^{-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0^{\tau-1} G(a_0)^{-2} & a_1^{\tau-1} G(a_1)^{-2} & a_2^{\tau-1} G(a_2)^{-2} & \cdots & a_{n-1}^{\tau-1} G(a_{n-1})^{-2} \end{bmatrix}.$$

Рис 1. Построение проверочной матрицы

Введем функцию $B(a_i) = (b_{i0}, \ldots, b_{i(m-1)})$, которая возвращает двоичное представление элемента a_i поля F_{2^m}. Если применить функцию $B(.)^{\mathrm{T}}$ к двоичному описанию каждого элемента матрицы $\boldsymbol{H}_m$, то получим матрицу $\boldsymbol{H} \in F_2^{mt\times n}$ранга mt=n-k. Порождающая матрица $\boldsymbol{G} \in F_2^{k\times n}$ кода Гоппы может быть легко вычислена на основе матрицы $\boldsymbol{H}$.

Основное преимущество кода Гоппы состоит в том, что существует быстрый алгоритм декодирования кодов Гоппа (метод Петерсона [6] или алгоритм Берлекэмпа-Месси [7,8]), но общая проблема найти слово кода по данному весу в линейном двоичном коде является NP-трудной. Во-вторых, любой неприводимый многочлен над полем $GF(2^m)$ является подходящим для задания кода Гоппы, следовательно, порождающая матрица кода является почти случайной. Следовательно, коды Гоппы легко задать, но, в то же время, сложно распознать - их количество растет экспоненциально в зависимости от длины слова и степени порождающего полинома.

Схема работы криптосистемы NTS-KEM:

KGen: генерируется полином Гоппа G(z) степени t, который описывает код Гоппа C_G, исправляющий t ошибок, длины $n = 2^m$ и мощности $k = n - tm$. Порождающая матрица кода $\mathbf{G} = [I_k \mid Q]$ вычисляется из проверочной матрица кода $\mathbf{H}_m$. Пусть p перестановка на n элементах, a z случайный двоичный вектор длины l. Также вводятся вектор a - все элементы поля, описанные через базис и вектор h - значения полинома G(z) в каждой точке поля. Далее к ним применяется перестановка p и удаляется часть значений.

Открытый ключ pk = (Q,t,l) включает в себя компактное представление порождающей матрицы кода, а так же константы t и l. Секретный ключ sk = (a*, h*, p, z, pk) состоит из элементов, которых достаточно, чтобы восстановить усеченный вид проверочной матрицы, используемый при декодировании. А также из перестановки p, случайного z и открытого ключа, что позволяет облегчить проверку того, что открытый ключ мог измениться до процесса декапсуляции ключа.

Encap: случайным образом генерируется вектор ошибок e длины n с весом Хэмминга t. Используя некоторую псевдослучайную функцию и вектор ошибок e, подсчитывается вектор $k_e = H_l(e)$. Затем вектор e разбивается на три части: $e = (e_a \mid e_b \mid e_c)$, где e_a имеет длину k-l, e_b - длины l, e_c - длины n-k. Следом строится вектор сообщения: $m = (e_a \mid k_e)$. Используя матрицу Q сообщение кодируется: $c = (c_b \mid c_c)$, где $c_b = k_e + e_b$ и $c_c = (e_a \mid k_e) \cdot Q + e_c$. Также, с помощью псевдослучайной функции, подсчитывается $k_r = H_l(k_e \mid e)$. В качестве возвращаемых значений будет выступать пара (k_r , c)

Decap: используя вектора a* и h* восстанавливается проверочная матрица $\boldsymbol{H}_m^*$. С помощью матрицы $\boldsymbol{H}_m^*$ подсчитываются все 2t синдромы s: $s = c \cdot (\boldsymbol{H}_m^*)^T$. Затем, с помощью алгоритма Берлекэмпа-Мэсси, находится полином σ(x), выступающий локатором ошибок. Используя σ(x), находится вектор e`, к которому для получения вектора ошибки e применяется перестановка p. С вектора e можно декапсулировать ключ.

Стоит отметить, что алгоритм NTS-KEM позволяет обеспечивать три уровня секретности согласно NIST [2]. Категории 1,3,5 обеспечивают классическую секретность для 128, 192, 256 битной длины ключа или блока шифра. Также определена постквантовая секретность для тех же категории и составляет соответственно 64, 96, 128 бит ключа или блока шифра.

Соответствие этим категориям в NTS-KEM(m,t) реализовано за счет задания параметров системы - m и t. Соответствующие реализации обеспечивают NTS-KEM(12,64), NTS-KEM(13,80), NTS-KEM(13,136). Для алгоритма доказана неразличимость для атак на основе подобранного шифротекста [1]. Также для криптосистем, основанных на кодах характерен специфический класс атак, основанных на ISD (information-set decoding). Например, современная атака основана на поиске коллизий между синдромом шифртекста и синдромом от выбранных столбцов проверочной матрицы.

Стоит отметить, что эффективных квантовых атак “кодовых” алгоритмов NTS-KEM и McEliece пока не предложено. Чтобы как-то приблизить квантовые компьютеры к таким атакам, используется алгоритм Гровера [9] или, как обсуждалось выше, вычисление синдромов от случайных столбцов проверочной матрицы. Авторы NTS-KEM в работе [1] также рассматривают устойчивость к другим типам атак, которым подвержены не только криптосистемы, основанные на кодах, но и другие.

В ходе работы была подробно изучена работа алгоритма NTS-KEM и сопутствующие этому разделы алгебры и теории кодирования. Также начата разработка алгоритма и за период летней школы удалось реализовать основные функции KEM, но при этом некоторые математические алгоритмы, необходимые для работы было решено сделать в более упрощенном виде, что повлияло на время работы полученной программы по сравнение с эталонной реализацией на языке C. В качестве будущих планов можно выделить несколько

этапов: реализация алгоритмов быстрых вычислений там, где это необходимо, а также добавление тестов, предлагаемых NIST для проверки корректности работы алгоритмов постквантовой криптографии.

ЛИТЕРАТУРА

1. Martin Albrecht, Carlos Cid, Kenneth G. Peterson, Cen Jung Tjhai, Martin Tomlinson. NTS-KEM Second round Submission. 2019.
2. NIST. Post-quantum cryptography standardization: call for proposals. 2016.
3. Robert J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. Deep Space Network Progress Report 42– 44 (1978), pp. 114–116.
4. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory, 15: 159–166, 1986.
5. Florence J. MacWilliams and Neil J. A. Sloane. The Theory of Error-Correcting Codes. North-Holland Publishing Company, 1977.
6. Nicholas J. Patterson. The algebraic decoding of Goppa codes. IEEE Transaction of information theory, 21(2):203-207, 1975.
7. Elwyn R. Berlekamp. Algebraic Coding Theory, New York: McGraw-Hill, 1968.
8. J. L. Massey. Shift-register synthesis and BCH decoding, IEEE Trans. Information Theory, IT-15: 122—127. 1969.
9. Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing p. 212. 1969.

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

БЛОКЧЕЙН-ТЕХНОЛОГИИ

Интеграция алгоритмов доказательства с нулевым разглашением в смарт-контракты Ethereum

А. А. Валитов, Д.А. Сафенрейтер, А. А. Лаханский

Новосибирский государственный университет

E-mail: **andreivalitov2001@gmail.com, d.safenreiter@g.nsu.ru, lil_pop96@bk.ru**

В рамках данного проекта был исследован принцип работы протокола zk-SNARK, а также технологии, используемые в нем. Из-за актуальности создания децентрализованных приложений крайне необходим высокоуровневый язык ZoKrates, который будет скрывать излишнюю сложность протокола zk-SNARK, а также позволять связывать программу с блокчейном Ethereum. В результате был изучен принцип работы инструментария ZoKrates с нуля, было проведено исследование его низкоуровневого устройства, а также были внесены изменения в стандартную библиотеку посредством добавления реализации новой функции.

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

На сегодняшний день слово “блокчейн” знакомо многим: технология, берущая истоки еще в прошлом веке, получила свое распространение и признание с появлением криптовалюты Биткоин. За ней последовали новые работы, расширяющие возможности сети блокчейна Биткоина — открылась целая область для разработок и исследований.

Среди них можно выделить проект Ethereum [1]. Он позволил применить концепцию блокчейна не только для децентрализованных валют, но и вообще для любых кейсов, в которых необходимо гарантировать выполнение действий каждой из сторон.

В наше время существуют две основные проблемы у современного блокчейна — это проблемы масштабируемости и конфиденциальности (в случае, если в данной области применения она является необходимой). Первая проблема возникает из-за того, что концепция требует обработки транзакций на каждом узле системы. Вторая — по причине того, что по существу данные блокчейна являются общедоступными, ведь они должны обрабатываться на каждом узле. Именно эти две проблемы считаются основным препятствием для внедрения блокчейна в разные области науки и бизнеса.

Решить последнюю проблему возможно с помощью алгоритмов сокрытия данных в распределенных системах. Один из самых известных и хорошо изученных — это доказательство с нулевым разглашением (ZK-Proof). Его цель заключается в том, чтобы проверяющий мог удостовериться в том, что проверяемый обладает знанием секретного параметра, удовлетворяющего некоторым отношениям, не раскрывая этот секрет проверяющему или кому-либо еще.

При данном подходе эти секретные данные по-прежнему продолжают влиять на транзакции в сети, не попадая в публичный доступ. Таким образом, доказательство с нулевым разглашением — это верный путь решения проблемы конфиденциальности данных пользователей в блокчейн-системах.

Строго говоря, доказательство с нулевым разглашением — это криптографический протокол, который позволяет одной стороне подтвердить истинность утверждения другой стороне, при этом не раскрывая никакой дополнительной информации о ней (ни содержания,

ни источника, из которого доказывающий узнал о правдивости). И самый распространенный тип такого доказательства в распределенных реестрах — это протокол zk-SNARK [2].

Для работы данный протокол должен удовлетворять определенным параметрам:

1. Полнота: если утверждение верно, то легальный (честный) верификатор может быть убежден в этом с помощью легального доказательства.
2. Корректность: если доказывающий лжет, то он не сможет убедить верификатора.
3. Нулевое разглашение (Zero Knowledge): если утверждение истинно, верификатор не будет знать, что скрывается в этом утверждении.

Внутри своей реализации zk-SNARK содержит сложную математику, основанную на работе с полиномами и гомоморфными скрытиями. Описать же ход выполнения доказательства можно, не владея серьезной математической базой.

zk-SNARK состоит из трех алгоритмов G, P и V:

1. Алгоритм G является генератором ключей. На этом шаге создаются два общедоступных ключа: ключ проверки (для доказывающего) и верифицирующий ключ (для верификатора). Эти ключи являются доступными для любой из заинтересованных сторон.
2. Алгоритм P позволяет породить доказывающей стороне доказательство на основе своего ключа, включая свои секретные данные. Именно на этом шаге мы получаем возможность показать лишь знание этих данных, а не сами данные.
3. Алгоритм верификатора V возвращает по итогу работы логическую переменную. Здесь проверяющий использует тот выход алгоритма P, который сформировала доказывающая сторона. Таким образом, благодаря данному алгоритму, он сможет понять, знает ли вторая сторона данные, не получая их самих.

Протокол позволяет повысить уровень приватности пользователей в публичных блокчейн сетях, а также укрепить информационную безопасность за счет замены неэффективных способов аутентификации и верификации.

Поскольку Ethereum представляет из себя платформу для создания децентрализованных онлайн сервисов, которые работают на блокчейне, то для него протокол zk-SNARK представляет исключительный интерес, ведь сам zk-SNARK позволяет одной стороне (доказывающему) доказать другой стороне (проверяющему), что утверждение верно, не раскрывая информацию, не касающуюся достоверности утверждения.

Ethereum выполняет вычисления на всех узлах сети, что приводит к высоким затратам, ограничениям по сложности и низкой конфиденциальности. zk-SNARK позволяет проверять вычисления в цепочке: части алгоритма zk-SNARK добавляются в сеть в виде заранее скомпилированных смарт-контрактов. После того, как генератор создал доказательный и проверочные ключи, любой доказывающий может использовать этот ключ для создания доказательства вне сети. А далее, алгоритм проверки может выполняться внутри смарт-контракта.

Однако, эти вычисления очень сложно понять, из-за чего работа с этими вычислениями становится чрезвычайно сложна.

ZoKrates, набор инструментов для работы с zk-SNARK на Ethereum, устраняет этот пробел [3]. Он помогает создавать автономные программы и связывать их с блокчейном Ethereum, таким образом расширяя возможности децентрализованного приложения.

ZoKrates уже может создавать простые схемы, но реализация сложных схем пока невозможна [4]. К тому же, все алгоритмы необходимо реализовывать вручную и при

изменении смарт-контракта также необходимо будет сгенерировать новые параметры.

В рамках проекта была проделана следующая работа: в течение обучения в Криптошколе был изучен алгоритм протокола zk-SNARK; мы впервые поработали с платформой Ethereum, разобрали и дополнили код ZoKrates, а также успешно разработали несколько тестовых смарт-контрактов.

Поскольку в криптографических системах очень часто используется операция взятия остатка от деления, например, в протоколе Диффи — Хеллмана, то очевидно, что в ZoKrates эта функция необходима. Мы добавили её в стандартную библиотеку ZoKrates. Операция была реализована через цикл с заранее ограниченным числом итераций, где на каждом шаге от исходного числа отнимается делитель.

Главная проблема такого подхода в том, что мы заранее не можем знать, сколько раз нам придётся отнять делитель, и для решения этой проблемы мы можем ограничить цикл очень большим числом и надеяться, что его хватит. Эту проблему нельзя решить в рамках языка ZoKrates, но можно избавиться от неё, реализовав эту операцию в ядре ZoKrates, что и является нашими планами на дальнейшую работу над этим проектом.

Концепция развития инструмента ZoKrates требует не только добавление операции взятия остатка от деления, но и других математических функций, а также создания полноценной документации ко всему проекту. На данный момент открыто более шестидесяти запросов пользователей на добавление нового функционала. Именно реализация как минимум нескольких из них и предложение авторам ZoKrates интегрировать эти дополнения могут стать новой задачей для нашей команды.

ЛИТЕРАТУРА

1. Vitalik Buterin. Ethereum Whitepaper: // Ethereum website. 2013. URL: https://ethereum.org/en/whitepaper/ (Дата обращения: 25.07.2020).
2. Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M. SNARKs for C: Verifying program executions succinctly and in zero knowledge: // CRYPTO 2013, Proceedings of the 33rd Annual Cryptology Conference, Part II, volume 8043 of LNCS. Santa Barbara, CA, USA, 2013. P. 90 – 108.
3. Jacob Eberhardt, Stefan Tai. Zokrates-Scalable Privacy-Preserving Off-Chain Computations: // IEEE International Conference on Internet of Things and IEEE Green Computing and IEEE Smart Data. - Berlin, 2018.
4. Jacob Eberhardt, Stefan Tai, Thibaut Schaeffer. ZoKrates official documentation. 2018-2020. URL: https://zokrates.github.io/ (Дата обращения: 19.07.2020).

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.

Разработка смарт-контракта для сервиса купли-продажи ипотечных закладных

В.А. Аламов[1], Д.А. Быков[2]

[1]Томский государственный университет,
[2]Новосибирский государственный университет,

E-mail: **alamovvladimir@gmail.com, d.bykov1@g.nsu.ru, mshchenyev@yandex.ru**

Смарт-контракт Ethereum это компьютерная программа, которая отслеживает и обеспечивает исполнение обязательств. В данной статье будет рассмотрена реализация смарт-контракта для сервиса купли-продажи ипотечных закладных. Для решения задачи были использован язык программирования Solidity, фреймворк разработки Truffle Suite, среда разработки Remix IDE. В результате был написан смарт-контракт купли-продажи ипотечной закладной.

Ключевые слова: смарт-контракт, блокчейн, язык программирования Solidity

Введение

Блокчейн — это разновидность децентрализованных систем, которая занимается сбором, хранением и управлением данных. Основным свойством блокчейна является его децентрализованность. Поэтому компрометация или вывод из строя одного узла не приведет к нарушению работоспособности всей сети. Следующим свойством блокчейна является возможность достижения консенсуса в недоверенной среде. Исторически первый алгоритм достижения консенсуса — Proof-of-Work (PoW). PoW — подход, который полностью зависит от вычислительных мощностей каждого члена сети для решения проблем и достижения консенсуса при проведении транзакции. Другой алгоритм — Proof-of-Stake (PoS). PoS — опирается на вероятностную модель для выбора валидаторов, где вероятность того, что валидатор получит блок для решения, прямо пропорциональна количеству монет, внесенных им в качестве залога для защиты сети.

Транзакции хранятся в структуре данных, называемой блоками, каждый последующий блок хранит значение хэш-функции от предыдущего, что гарантирует неизменяемость и целостность данных.

Одной из реализаций блокчейна является Ethereum. Первое упоминание об Ethereum было в 2013 г., когда программист из Канады Виталик Бутерин выпустил white paper, но первая реализация появилась лишь в 2015. Чтобы продукт имел успех на рынке, он должен, либо предлагать то, что не предлагали раньше, либо быть дешевле. Ethereum предлагает инструменты для разработки смарт-контрактов.

Смарт-контракт — это программа, которая хранится в распределенном реестре и выполняет некоторый прописанный в ней код в ответ на адресованные ей транзакции. При исполнении смарт-контракты заверяются цифровой подписью каждой из сторон. Для реализации смарт-контрактов в Ethereum существует несколько языков, но самым стабильным является Solidity.

Solidity [1] — JavaScript-подобный, тьюринг-полный, ООП язык со статической типизацией. Solidity транслируется в байткод EVM (виртуальная машина Ethereum) и исполняется на ней. Он позволяет разработчикам создавать самодостаточные приложения, содержащие бизнес-логику, результирующую в неотменяемые транзакционные записи блокчейна.

В качестве цели данной работы была выбрана реализация смарт-контракта купли-продажи ипотечной закладной.

Закладная по ипотеке — это документ, обладающий характеристиками ценной бумаги, который свидетельствует о праве его владельца на получение кредитных средств, обеспеченных ипотекой, а также на имущество, обремененное ею. На практике закладная используется с целью ускорения и упрощения операций с недвижимым имуществом.

До нас подобный кейс был реализован Райффайзенбанком. С 7 февраля 2020 года Райффайзенбанк заменил бумажную закладную на электронную ценную бумагу, запись о которой хранится в реестре блокчейн. Банк посчитал, что ее внедрение поможет его клиентам и рынку в целом за счет сокращения сроков передачи электронной закладной на учет и хранение, снижения издержек на депозитарные операции и улучшения клиентского опыта заемщиков, которым больше не придется получать закладные в бумажном виде [2].

Открытость и децентрализованность блокчейна — это его преимущество, но в то же время и его недостаток. Прозрачность всех проводимых транзакций сужает допустимую область применения этой технологии, приводит к потере большой части бизнес-рынка. Поэтому общая задача была разработать безопасный сервис для купли-продажи ипотечных закладных. В рамках данной работы стояла задача разработать смарт-контракт купли-продажи ипотечной закладной для дальнейшего его использования в общем проекте.

1. Разработка смарт-контракта

Для достижения общей цели, нашей командой были поставлены следующие задачи:

1. Изучить язык Solidity — один из языков разработки смарт-контрактов.
2. Изучить Truffle [3] — консольный фреймворк для разработки, тестирования и развертывания смарт-контрактов.
3. Изучить Remix IDE [4]— онлайн среда разработки смарт-контрактов.
4. Написать смарт-контракт, реализующий куплю-продажу ипотечной закладной.
5. Протестировать смарт-контракт.

Для реализации смарт-контракта купли-продажи ипотечной закладной был использован язык программирования Solidity. Компиляция была осуществлена посредством компилятора solc 0.5.13. Это не последняя версия компилятора, но было принято решение остановиться на ней, как на стабильной и все еще актуальной. Минус эволюции компиляторов solc заключается в том, что каждая новая версия языка не поддерживает многих методов и возможностей предыдущей.

Основная разработка велась в IDE Remix. Особенностью Remix является то, что он по умолчанию имеет все нужные плагины для комфортной разработки. Но даже не это является его главным преимуществом. Его основной плюс перед другими IDE в том, что его не надо ставить себе на машину, а достаточно запустить в веб-браузере.

Также на ранних этапах разработки использовался консольный фреймворк Truffle, который позволяет разрабатывать, тестировать и развертывать смарт-контракты. Но от него отказались так как Remix удобнее.

Для разработки смарт-контракта был выбран подход, при котором каждый участник нашей команды сначала напишет свой смарт-контракт купли-продажи, а потом будет выбран лучший.

Первым делом была разработана структура ипотечной закладной. Наша ипотечная закладная имеет следующие поля:

1. Номер закладной.
2. Цена квартиры.
3. Дата создания закладной.
4. Дата оформления ипотеки.

5. Дата последней выплаты.
6. Владелец закладной.

В коде смарт-контракта при продаже закладной проверяли соответствие каждому из следующих условий:

1. Дата последней выплаты заемщиком средств ипотеке должна превышать полгода.
2. Дата создания закладной не должна быть позже даты выдачи ипотеки.

Нам пришлось добавить дополнительное поле для облегчения работы с закладной, а именно порядковый номер в массиве закладных. А также поле, отображающее действительна ли данная закладная.

Главной функцией смарт-контракта является функция передачи закладной. Она корректно отрабатывает лишь в случае, когда все условия выше соблюдены, иначе транзакция останавливается и возвращается к предыдущему состоянию. Обращаясь к смарт контракту, сгенерированному ZoKrates[5][6], также проверяется еще одно условие, а именно, что сумма продажи закладной меньше 90% от стоимости квартиры. Таким образом, информация о стоимости продажи закладной не попадает в блокчейн.

Был взят проверенный и безопасный контракт Owner для предоставления функциональности установки и смены владельца нашего контракта. Владелец наделен особыми правами по вызову функций контракта, например, он может добавлять новые банки.

Часть функций смарт-контракта была протестирована Unit-тестами на языке solidity с помощью плагина “Solidity Unit testing” в Remix IDE.

Заключение

В результате работы были достигнуты все поставленные задачи. Были изучены технологии разработки смарт-контрактов: язык программирования Solidity, фреймворк Truffle и онлайн среда разработки Remix. Был разработан и протестирован смарт-контракт купли-продажи ипотечной закладной. Смарт-контракт был развернут в тестовой локальной сети и интегрирован в веб-приложение, разработанное другими участниками команды .

ЛИТЕРАТУРА

1. Документация Solidity, [Электрон. ресурс], URL: https://solidity.readthedocs.io/en/v0.5.3/using-the-compiler.html
2. Райффайзенбанк готов хранить электронные закладные в блокчейн, [Электрон. ресурс], URL: https://www.raiffeisen.ru/about/press/releases/125680/
3. Документация Truffle Suite, [Электрон. ресурс], URL: https://www.trufflesuite.com/docs
4. Remix IDE, Онлайн компилятор, URL: https://remix.ethereum.org/
5. Zokrates - Scalable Privacy - Preserving Off - Chain Computations [Электрон. ресурс] / Jacob Eberhardt, Stefan Tai - научный журнал - Берлин : 2018.: https://www.ise.tu-berlin.de/fileadmin/fg308/publicat.
6. Кондырев Д.О. разработка метода сокрытия приватных данных для системы тендеров на основе технологии блокчейн//прикладная дискретная математика.2020.№48.С.5

Куратор исследования – Сазонова олина Андреевна, аспирантка ФИТ НГУ, ассистент кафедры общей информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Внедрение протокола доказательства с нулевым разглашением для сервиса купли-продажи ипотечной закладной

И.А. Матеюк, А.А. Базаров

Новосибирский государственный университет

Email: i.mateyuk@g.nsu.ru, a.bazarov1@g.nsu.ru

В рамках данной темы проведено исследование протокола zk-Proof и его последующее внедрение в смарт-контракт, реализующий куплю-продажу ипотечной закладной. Так как для сделки использовалась платформа Ethereum на базе блокчейн, то подробности транзакции были доступны всем участником P2P сети. Использование zk-Proof позволило скрыть часть информации и, как следствие, не допустило злоупотребление этими данными третьими лицами. В качестве инструмента для внедрения zk-Proof был использован ZoKrates — стремительно развивающийся набор инструментов для zk-SNARKs в Ethereum. В результате исследования нам удалось изучить концепцию доказательства с нулевым разглашением и интегрировать ее в предоставленный нашими коллегами смарт-контракт. Таким образом, мы развернули рабочее приложение по продаже ипотечной закладной между банками, использующее сокрытие суммы транзакции.

Ключевые слова: Ethereum, zk-Proof, zk-SNARKs, ZoKrates, смарт-контракт, solidity.

Введение

Общей задачей нашего проекта являлась разработка сервиса для купли-продажи ипотечных закладных, используя протокол доказательства с нулевым разглашением. В рамках данной работы требовалось внедрить этот протокол в сервис. Основная работа велась с блокчейн на платформе Ethereum. Блокчейн — это разновидность децентрализованных систем, которая занимается сбором, хранением и управлением данными.

Перед началом исследования были поставлены следующие задачи:

1. Поиск и изучение уже существующих исследований, результатов и продуктов, использующих zk-Proof.
2. Выбор инструмента для работы с протоколом zk-Proof, учитывая специфическую область применения.
3. Реализация функций zk-Proof, позволяющих сохранить конфиденциальность персональных данных и проверку суммы транзакции.
4. Внедрение zk-Proof в смарт-контракт для купли-продажи ипотечной закладной и написание API для Java application.

Кейс по продаже закладной уже был описан Райффайзен Банком. Они делали ипотечную закладную, но мы решили реализовать этот кейс иначе — скрывая данные транзакции, а именно сумму сделки. Открытость и децентрализованность блокчейна — это его достоинство, но в то же время это и недостаток. Прозрачность всех проводимых транзакций сужает допустимую область применения этой технологии. В нашей реализации большая часть внимания была уделена обеспечению сохранности персональных данных.

Инструментарий

Столкнувшись с проблемой конфиденциальности, эффективности и масштабируемости транзакций в сети блокчейн, нами было проведено исследование рынка готовых решений. Ввиду возраста технологии (всего 12 лет) таковых оказалось довольно немного.

Для решения задачи мы использовали zk-SNARK, в основе которого лежит **Zero-Knowledge Proof** — это протокол, обладающий свойствами полноты и корректности, который позволяет обмениваться данными между двумя сторонами без использования пароля или любой другой информации, связанной с транзакцией. Это метод, с помощью которого одна сторона может доказать другой стороне (проверяющему), что она знает эту истину, не передавая содержание этой истины проверяющему. Используя доказательства с нулевым разглашением, мы также обеспечили конфиденциальность транзакций, что позволило расширить область применения смарт-контракта.

Solidity был выбран в качестве объектно-ориентированного языка программирования для создания смарт-контрактов на платформе Ethereum.

Для реализации zk-SNARK в нашем проекте был использован набор инструментов **ZoKrates.** Он скрывает значительную сложность, присущую доказательствам с нулевым разглашением, и предоставляет разработчикам более знакомую и высокоуровневую программную абстракцию. Для этого он реализует набор инструментов для определения, интеграции и развертывания вычислений вне цепочки. Этот набор состоит из предметно-ориентированного языка, компилятора и генераторов для проверки и доказательства смарт-контрактов.

Разработка проводилась на базе браузерной IDE Remix. Ее отличительной чертой является наличие плагина для работы с ZoKrates. Там же проводились создание ключей, генерация доказательств и внедрение.

Разработка и внедрение

Разработка с использованием ZoKrates состояла из нескольких этапов, для наглядности на рис.1 приведена иллюстрация, которая содержит весь цикл.

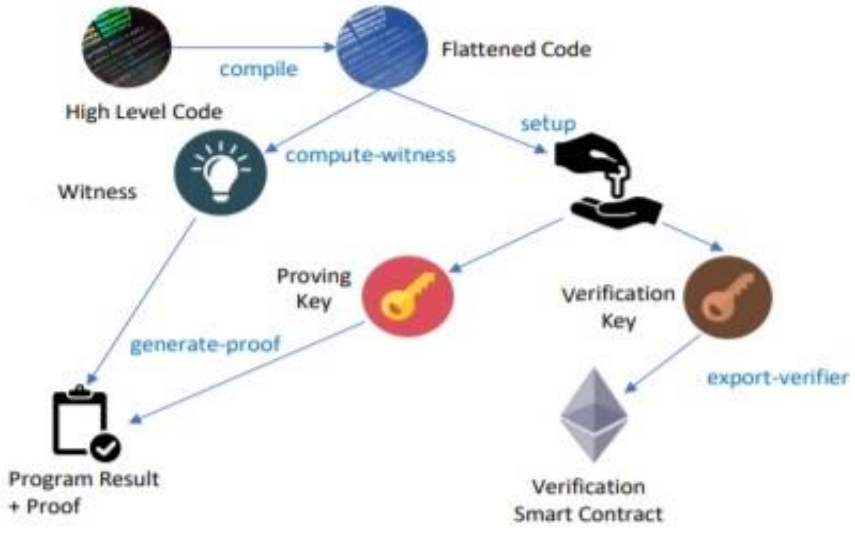


рис. 1, цикл разработки в ZoKrates.

В блокчейн процесс интеграции ZoKrates выглядит следующим образом: блоки алгоритма проверки добавляются в Ethereum в виде предварительно скомпилированных контрактов. Генератор ключей 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. Далее опишем каждый этап и сложности, с которыми нам пришлось столкнуться.

Конвейер ZoKrates (рис.1) позволил упростить нам развертывание zk-Proof в нашем смарт-контракте. Для этого было пройдено несколько этапов:

1. **compile**: Для начала был написан модуль main.zok, который содержал в себе всю логику. В контексте нашей задачи он проверял валидность суммы транзакции при покупке ипотечной закладной у банка, при этом не раскрывая никаких подробностей. При компиляции была получена схема исходной функции в двоичном и читаемом человеком виде.
2. **setup**: Далее были сгенерированы доказательный и проверочный ключи для этой схемы. На этом этапе было важно следить за тем, чтобы процесс был псевдослучайным, иначе алгоритм не обеспечивает должной криптостойкости.

   Доказательство zk-SNARKs утверждает, что проверяющий знает некоторую приватную информацию, которая удовлетворяет исходной функции в main.zok. Эта приватная информация называется свидетелем, и, следовательно, у каждого свидетеля будут разные доказательства. Свидетели должны быть закодированы, прежде чем они будут использованы zk-SNARKs.

   Далее нами было сгенерировано доказательство, при этом использовались проверочный ключ и закодированный свидетель из предыдущего шага.
3. **export-verifier**: Следом доказательство было подготовлено для внедрения в смарт-контракт купли-продажи закладной на платформе Ethereum. Был создан контракт на языке solidity для его проверки. Проводилось тестирование контрактов и ключей доказательства в клиенте приложения и блокчейне. Схему их взаимодействия можно увидеть на рис.2.

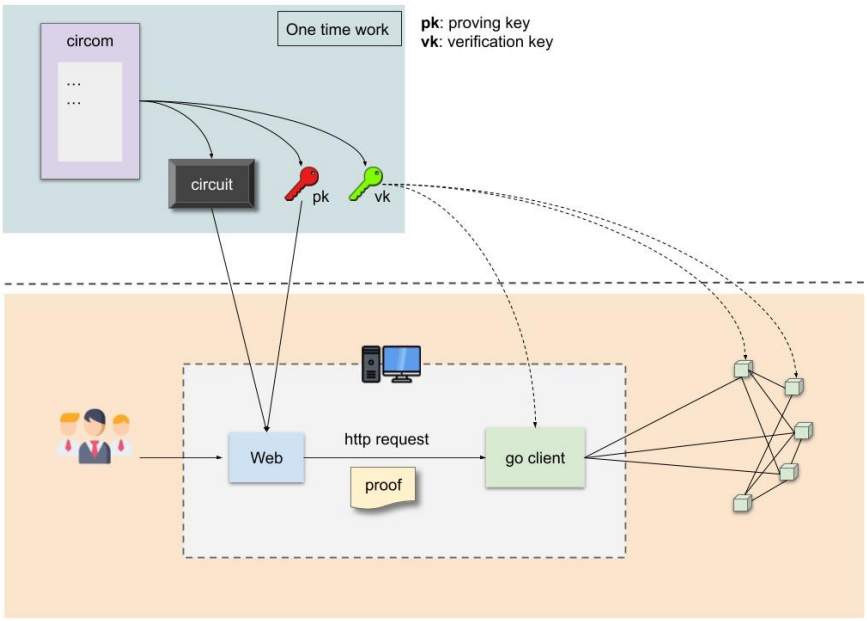


рис. 2, использование zk-SNARK в Java приложении.

Трудности

В процессе работы над внедрением zk-SNARK в наш сервис купли-продажи мы столкнулись с некоторыми недостатками ZoKrates.

I. У нашей команды возникли сложности с интеграцией готового контракта в код и с использованием проверяющего и доказывающего ключей в Java-приложении наших коллег. В ZoKrates отсутсвует удобное API для интеграция во многие языки. Чтобы это исправить, понадобилось дописать скрипт, который позволил связать ZoKrates с Java.

II. Популяризация и повсеместное введение zk-Proof в блокчейн — довольно трудная задача, потому что за каждую операцию в Ethereum нужно заплатить, единицей оплаты является “газ”, а проверка данных zk-SNARKs стоит дорого. Это связано с тем, что на проверку доказательств требуется большой объем вычислительных мощностей. Их стоимость в эфириуме исчисляется в “газе”. Сейчас цена проверки смарт-контрактов на zk-SNARKs составляет около 1,6 миллиона газа, что делает нерентабельными сделки на меньшие суммы.

III. Доказательства с нулевым разглашением не являются доказательствами в математическом смысле этого термина, потому что есть некоторая небольшая вероятность, что обманом доказывающая сторона сможет убедить Проверяющего в ложном утверждении. Иными словами, доказательства с нулевым разглашением — это вероятностные доказательства, а не детерминированные. Тем не менее, есть методы, позволяющие уменьшить ошибку корректности до пренебрежимо малых значений.

Заключение

После длительного и детального изучения протокола zk-Proof он был успешно интегрирован в проект. С помощью набора инструментов ZoKrates удалось реализовать и

внедрить zk-SNARK в разработанный командой сервис купли-продажи ипотечных закладных. Это позволило производить транзакции, не раскрывая дополнительной информации третьим лицам. ZoKrates имеет огромный потенциал, его применение в нашем проекте сильно облегчило задачу интеграции zk-Proof в кейс. Проблемы, с которыми мы столкнулись, задали нам вектор дальнейших исследований в этой области. Мы планируем способствовать популяризации этого инструмента путем написания удобного API, который позволял бы без лишних трудностей интегрировать его в различные приложения. Мы верим и очень надеемся, что с течением времени блокчейн вместе с zk-Proof найдут массовое применение во многих отраслях и привлекут к себе внимание новых специалистов.

ЛИТЕРАТУРА

1. Zokrates - Scalable Privacy - Preserving Off - Chain Computations [электронный ресурс] / Jacob Eberhardt, Stefan Tai - научный журнал - Берлин : 2018.: https://www.ise.tu-berlin.de/fileadmin/fg308/publicat..
2. Thibaut Schaeffer - official documentation [электронный ресурс] : https://zokrates.github.io/
3. Christian Reitwießner - zkSNARKs in a Nutshell [электронный ресурс] : http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf
4. Jens Groth - On the Size of Pairing-based Non-interactive Arguments [электронный ресурс] / 2016 : https://eprint.iacr.org/2016/260
5. Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof systems : 1989. - P. 186—208.

Куратор исследования – Сазонова Полина Андреевна, аспирантка ФИТ НГУ, ассистент кафедры общей информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

Разработка веб-приложения для сервиса купли-продажи ипотечных закладных

Щербина Д.А., Раимбеков А.Р.

E-mail: daniil.sh775@gmail.com, azimraimbekov@gmail.com

[1]Томский государственный университет

В рамках разработки сервиса купли-продажи ипотечных закладных с использованием семейства протоколов zk-proof, в данной работе разрабатывалась серверно-клиентская часть приложения. Используя библиотеку Web3j на языке Java, был написан модуль, позволяющий исполнять логику смарт-контракта и передавать данные на сервер. В качестве инструмента сокрытия данных был выбран ZoKrates. Нами был написан Shell-скрипт, компилирующий исходные файлы ZoKrates. Таким образом, модуль, взаимодействующий с инструментом ZoKrates интегрирован в наше приложение Java. Серверная часть также была написана на языке Java с помощью микро-фреймворка Spark. Для получения и визуализации данных от пользователя был реализован веб-интерфейс.

Ключевые слова: Ethereum, zk-Proof, zk-SNARKs, ZoKrates, смарт-контракт, solidity, Web3j, Spark, микро-фреймворк.

Введение

В рамках проекта проведения сделки купли-продажи между двумя банками ипотечной закладной разрабатывался сервис, который позволяет банку создать или продать закладную в сети Ethereum. Задача нашей работы состояла в реализации серверной и клиентской части.

Блокчейн — это разновидность децентрализованных систем, которая занимается сбором, хранением и управлением данными. Основным свойством блокчейна является его децентрализация. Поэтому компрометация или вывод из строя одного узла не приведет к нарушению работоспособности всей сети. При совершении сделки купли-продажи ипотечных закладных на блокчейн-платформе Ethereum с открытием подробностей сделки, информация о задолженности может быть получена злоумышленниками и использована в их целях. Поэтому необходимо реализовать сделку так, чтобы при ее совершении подробности не были разглашены, но обе стороны были уверены в достоверности информации о передаваемой закладной. Для решения этой задачи был сделан выбор в пользу использования семейства протоколов zk-proof (доказательство с нулевым разглашением).

Цели и задачи

Для реализации веб-приложения были поставлены следующие задачи:

1. Разработать backend-часть сервиса на языке Java [5] с использованием библиотеки Web3j [2] для связи смарт-контракта с сервером.
2. Интегрировать разработанный коллегами по проекту модуль, использующий ZoKrates (набор инструментов для zk-SNARKs в Ethereum) [1].
3. Разработать сервер на языке Java связывающий веб-интерфейс и Java Application.
4. Разработать веб-интерфейс для пользователя.
5. Реализовать безопасную авторизацию пользователей в сети Ethereum.
6. Интегрировать клиентскую и серверную часть.

Реализация

Перед разработкой веб-приложения был проведен сбор требований и на их основе составлен перечень функционала, который должен быть реализован в программной системе:

- Возможность создать закладную пользователем на веб-странице.
- Возможность продажи закладной пользователем на веб-странице.
- Возможность отследить изменения, просматривая динамически изменяющуюся таблицу закладных и адреса владельцев.
- Авторизация пользователей с использованием плагина MetaMask [6].

Ниже на рис. 1 представлена схема архитектуры сервиса купли-продажи ипотечных закладных.

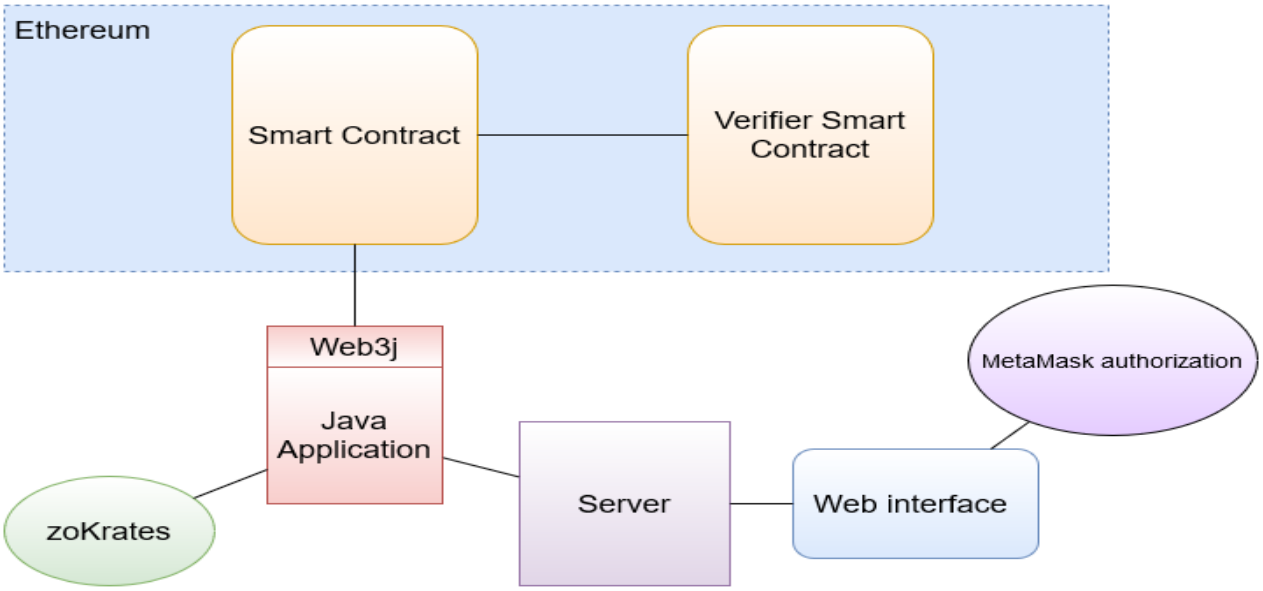


Рис. 1 Схема работы сервиса

1. Разработка обвязки смарт-контракта

Для реализации взаимодействия веб-приложения со смарт-контрактом была выбрана библиотека Web3j. Web3j — библиотека Java для работы со смарт-контрактами и интеграции с клиентами (узлами) в сети Ethereum (Рис. 2).

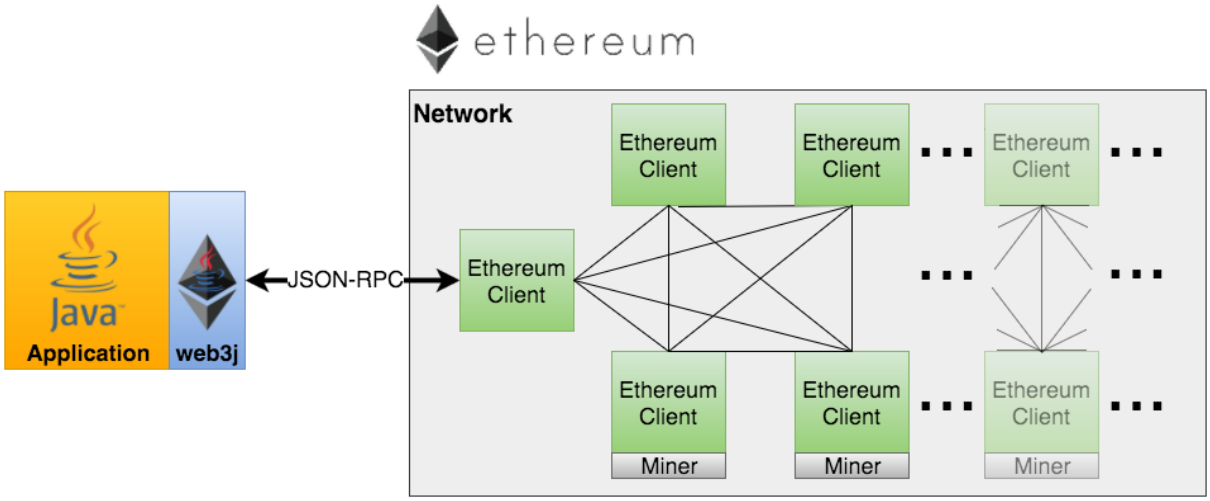


Рис. 2 Схема связи библиотеки Web3j и сети Ethereum

Она позволяет осуществлять работу с блокчейн Ethereum без дополнительных затрат на написание собственного кода интеграции для платформы. Web3j конвертирует исходный

смарт-контракт на языке Solidity в Java класс. В нем описываются все функции смарт-контракта и он позволяет выполнять такие действия как развернуть/загрузить смарт-контракт из нашего приложения. Также использование библиотеки позволяет производить транзакции в сети Ethereum с данными, полученными от пользователя с веб-интерфейса.

Нами был разработан рабочий Java application, который связывает серверную часть приложения и смарт-контракт, запущенный в сети Ethereum.

Для тестирования нашей системы использовалась тестовая сеть Ganache. Также наше приложение можно развернуть и подключить к любой сети Ethereum.

2. Интеграция модуля, реализованного с использованием ZoKrates

Модуль с использованием ZoKrates был реализован нашими коллегами по проекту. В ходе работы выяснилось, что в ZoKrates не предусмотрены методы интеграции в такие языки как Java. Для использования модуля в приложении, нами был написан отдельный Shell-скрипт, который позволяет выполнять команды ZoKrates и компилировать доказательство из нашего Java Application.

3. Разработка серверной части

В качестве основного фреймворка для разработки серверной части был выбран Java Spark [3]. Spark Framework — это простой и выразительный веб-фреймворк Java/Kotlin DSL, созданный для быстрой разработки. Нами было реализовано отображение веб-интерфейса клиенту, обмен данными между веб-формой и самим сервером. Также сервер подключен к нашему Java Application, может принимать и передавать данные, необходимые для совершения транзакций.

4. Разработка веб-интерфейса

Frontend — это часть системы которая отвечает за взаимодействие с пользователем. Пользователь взаимодействует с системой посредством веб-страницы.

Компоненты которые использовались для разработки frontend-части:

- HTML — язык разметки документов для создания структуры страницы: заголовки, абзацы, списки и так далее [4].
- CSS — язык для описания и стилизации внешнего вида документа. Благодаря CSS-коду браузер понимает, как именно отображать элементы.
- JavaScript — это язык, который создавался, чтобы оживить веб-страницы. Его задача — реагировать на действия пользователя, обрабатывать клики мышкой, перемещения курсора, нажатия клавиш.

Основная задача заключалась в визуализации функционала пользователю. Нами разработана веб-страница, содержащая формы для заполнения данных. На языке JavaScript и HTML были описаны базовые проверки введенных данных, чтобы не нагружать серверную часть. Так как тестирования сервиса происходило на localhost, данные с веб-страницы отправляются на сервер, не используя цифровую подпись и прочие методы безопасности. Но при размещении веб-сервиса в сети Интернет, следует учитывать безопасность и шифрование данных при передаче на сервер.

5. Интеграция авторизации пользователей

Для идентификации пользователей в нашем приложении был интегрирован плагин MetaMask. MetaMask был создан для удовлетворения потребностей безопасных и удобных веб-сайтов на основе Ethereum. В частности, он управляет учетной записью и подключением пользователя к блокчейну. MetaMask позволяет пользователям управлять учетными записями и их ключами различными способами, изолируя их от контекста сайта. Это значительное улучшение безопасности по сравнению с хранением пользовательских ключей на одном

центральном сервере или даже в локальном хранилище, что может позволить осуществить массовые кражи учетных записей.

При совершении транзакции пользователю предлагается авторизоваться через MetaMask для отправки введенных данных с форм. Метод авторизации с использованием MetaMask был описан нами на языке JavaScript. С помощью MetaMask сервер узнает адрес пользователя и может проводить транзакции от его имени.

Вывод

В рамках данного проекта были изучены аспектов разработки на языках Java, JavaScript, CSS и HTML. На Java была описана серверная часть с использованием библиотеки Web3j, для связи смарт-контракта с сервером внедрялся микро-фреймворк Spark для хостинга веб-страницы. Язык JavaScript был связующим звеном между микро-фреймворком Spark и HTML-формой веб-приложения.

На данный момент наш веб-интерфейс работает на протоколе http и производит авторизацию с помощью MetaMask. Через веб-страницу пользователь может совершать транзакции продажи и создание новых закладных, а также просматривать таблицу базы закладных.

Мы планируем перейти на протокол https для стойкого шифрования, использовать цифровую подпись (SSL сертификаты). Также расширить функционал серверной части и веб-интерфейса.

ЛИТЕРАТУРА

1. Zokrates — Scalable Privacy — Preserving Off - Chain Computations [электронный ресурс] / Jacob Eberhardt, Stefan Tai — научный журнал — Берлин : 2018 : https://www.ise.tu-berlin.de/fileadmin/fg308/publicat.
2. Web3j official documentation [электронный ресурс] : https://docs.web3j.io/.
3. Spark official documentation [электронный ресурс]: http://sparkjava.com/.
4. HTML documentation [электронный ресурс]: http://htmlbook.ru/html.
5. Java documentation [электронный ресурс]: https://docs.oracle.com/en/java/.
6. MetaMask documentation [электронный ресурс]: https://docs.metamask.io/guide/.

Куратор исследования - Сазонова Полина Андреевна, аспирантка ФИТ НГУ, ассистент кафедры общей информатики ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research,.

СЕКРЕТНЫЕ ЧАТЫ

Разработка секретного чата TGmini

А.П. Евсюков[1], В.В. Скудина[1], В.О. Карнаухова[2], Н.С. Ляпич[3], Ю.И.Эйсвальд[1], А.А.Котельникова[1], С.В.Помыкалов[5], А.А.Диденко[4]

[1]Новосибирский государственный университет,
[2]Алтайский государственный технический университет им.И.И.Ползунова,
[3]МАОУ “Лицей №6”, г. Бердск,
[4] ОСШЛМФИ, г. Усть-Каменогорск,
[5] МБОУ "ЛСОШ № 2", г. Луховицы

E-mail: **yevsyukof@gmail.com, v.skudina@g.nsu.ru, karnauhova01@bk.ru, nikita.lyapich@gmail.com, y.ejsvald@g.nsu.ru, a.kotelnikova@g.nsu.ru, Savelypomykalov@yandex.ru, a.rudskoj@icloud.com**

Проблема с передачей данных в групповых секретных чатах до сих пор не решена. В существующих реализациях обнаруживают все больше уязвимостей, поэтому авторами тезисов была предпринята попытка написать собственное приложение, поддерживающее секретные групповые чаты, алгоритм работы которого исключает некоторые из этих уязвимостей. В данных тезисах представлено описание алгоритма работы и сведения о программной реализации этого приложения.

Ключевые слова: секретный чат, групповой секретный чат, шифрование данных, передача данных, язык программирования Java

Пожалуй, самым популярным способом общения между людьми сейчас являются различные мессенджеры. Все больше людей задумываются о защите своих переписок и используют секретные чаты.

Особый интерес представляют групповые секретные чаты. Не все мессенджеры их поддерживают, например в Telegram[1] групповой чат нельзя сделать секретным. Но в Signal [2] или WhatsApp [3] такая функциональность присутствует, хотя они имеют свои уязвимости. Авторы данной работы предпринимают попытку создать секретный чат с поддержкой шифрования групповых чатов, исправив эти уязвимости.

ИЗВЕСТНЫЕ УЯЗВИМОСТИ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Главной уязвимостью секретных чатов, подобных WhatsApp, является способность, в случае захвата контроля над сервером, перехватить идентификатор группы и добавить в нее фиктивного участника без получения им пригласительного кода одного из участников группы. Прежде всего эта уязвимость связана с тем, что контролем добавления участников в группу занимается сервер. Также эта недоработка связана с отсутствием проверки факта участия в группе пользователя, отправившего управляющее сообщение с включением нового участника в группу. В Signal эта уязвимость также присутствует, но идентификатор конференции, размером 128 бит, необходимо угадать, что является сложной задачей.

Другая уязвимость ассоциирована уже с клиентом. Она связана с хранением сообщений на устройстве пользователя или в облачном хранилище. Они хранятся в зашифрованном виде, однако, ключ для их расшифрования также сохраняется на диске. Имея доступ к файлам на устройстве, злоумышленник может получить данные.

А так как в мессенджерах предусмотрена возможность открыть параллельную сессию на другом устройстве, это еще больше упрощает задачу взломщику.

К тому же, в обоих мессенджерах предусматривается регистрация по номеру мобильного телефона, что исключает анонимность.

СВЕДЕНИЯ О РЕАЛИЗАЦИИ

Работа приложения основана на клиент-серверном взаимодействии.

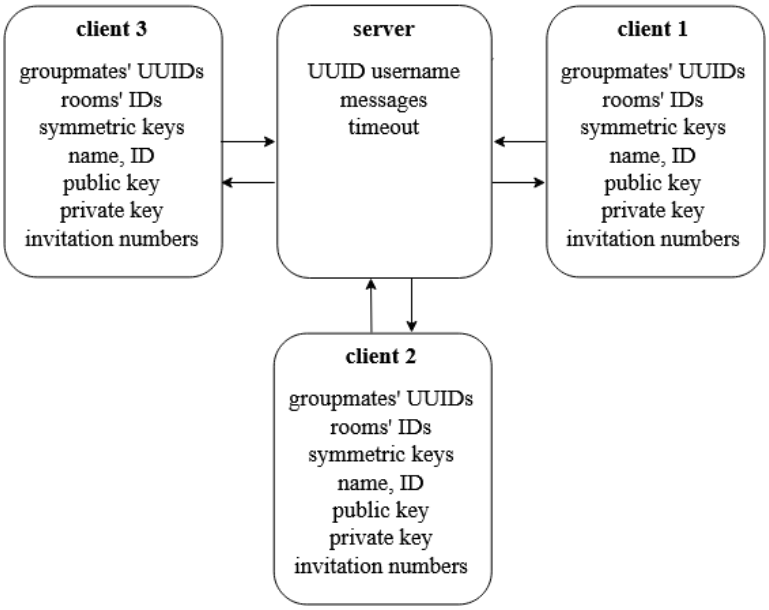


Рис. 1 Схема взаимодействия клиента и сервера

На сервере хранятся:

- таблица имен и идентификаторов пользователей;
- пакеты с сообщениями, которые еще не дошли до получателя и не были удалены по таймауту.

У клиента хранятся:

- таблица идентификаторов и имен пользователей, с которыми он общается;
- таблица идентификаторов комнат, в которых он находится, и их названий;
- секретные числа, отправленные другим пользователям для входа в чат;
- открытый и закрытый ключи асимметричного шифрования;
- общий ключ симметричного шифрования;
- собственные идентификатор пользователя и имя.

В нашем решении была реализована REST-архитектура [4], мы учли все ее обязательные принципы. Но мы не используем кэширование на стороне клиента, поскольку это снижает уровень секретности обмена сообщениями. Слои и код по

требованию в нашей реализации не требуются. Мы используем традиционный для REST-архитектуры протокол HTTP [4].

Отправка и принятие сообщений в приложении происходят в двух потоках:

1. в одном потоке формируются запросы на передачу сообщений, а также запросы на авторизацию и выход из приложения;
2. второй потом посылает запросы на наличие сообщений для клиента от сервера или других пользователей.

Перед отправкой сообщения вся информация в виде «ключ-значение» собирается в JSON [4] на стороне клиента и передается на сервер. На сервере сначала происходит разбор JSON-объекта, затем полученные данные обрабатываются сервером. Процесс обработки зависит от вида запроса. После этого сервер посылает ответ пользователю, его вид также зависит от запроса.

Для работы с JSON используется библиотека jackson.

Запрос с сообщениями содержит, собственно, сами сообщения, ID комнаты, в которую оно отправляется, идентификатор пользователя и идентификатор комнаты. Запрос на авторизацию содержит логин пользователя.

Основные запросы:

- /login <username> — запрос на авторизацию. Если логина нет в базе данных сервера, то он сохраняется в ней, и пользователю отправляется ответ с подтверждением авторизации. Если такой логин уже есть в базе данных, то пользователю отправляется ответ с кодом ошибки и предложением использовать другой логин;
- /logout — запрос на выход из приложения. Сервер удаляет данные о пользователе и отправляет сообщение с подтверждением выхода;
- /addMessage — запрос на добавление сообщения в личный список сообщений пользователя на сервере;
- /getMessages — запрос на получение с сервера списка сообщений для пользователя. После подтверждения получения, сообщения удаляются из списка.

ОСНОВНАЯ ФУНКЦИОНАЛЬНОСТЬ ЧАТА

Создание чата

Пусть Боб — первый участник и создатель группового чата. Боб:

1. Запускает программу.
2. Вводит свое имя пользователя. Если пользователь с таким именем уже существует, Боб получит от сервера сообщение об ошибке с предложением использовать другой никнейм.
3. После авторизации Боб создает комнату чата с помощью команды /createRoom и получает ее идентификатор.

Приглашение участника

1. Боб хочет пригласить Алису и безопасно передает ей случайно сгенерированное секретное число и свой идентификатор пользователя.

2. Когда Алиса входит на сервер, она отправляет Бобу запрос на получение его открытого ключа.
3. Боб отправляет открытый ключ Алисе.
4. Алиса зашифровывает на открытом ключе Боба секретное число, высланное им для доступа в групповой чат, и отправляет обратно.
5. Боб расшифровывает присланное сообщение и сравнивает секретное число с теми, что хранятся в его списке. Если числа совпали, он сообщает остальным участникам чата идентификатор Алисы. Если число неверное, он отправляет Алисе сообщение об ошибке.
6. После этого Боб зашифровывает с помощью секретного числа Алисы список пользователей, которые состоят в чате, идентификатор комнаты, ее название и отправляет данные Алисе. Секретное число удаляется.
7. Алиса сохраняет список пользователей, идентификатор комнаты и ее название.

Смена ключа

Смена ключа происходит в трех случаях: при входе в комнату нового участника, при выходе участника из чата и при истечении времени использования предыдущего ключа.

Процесс смены ключа:

1. Пусть Алиса — инициатор смены ключа (новый участник группы или случайно выбранный создателем чата инициатор).
2. Пользователи в группе пересылают свои открытые ключи RSA [5-6] Алисе.
3. Алиса генерирует случайное секретное значение, зашифровывает его на открытых ключах и рассылает сообщения адресатам.
4. Пользователи в группе расшифровывают присланные им секретные значения.
5. Вычисляется значение хэш-функции по алгоритму MD5 [7] от расшифрованного секретного значения и, таким образом, получается общий ключ для AES [8-9]. Такое решение обусловлено ограничениями, связанными с реализацией RSA. В будущем эта проблема будет исправлена.

Шифрование

1. Алиса получает общий закрытый ключ.
2. Она составляет сообщение и зашифровывает его общим ключом симметричного шифрования с помощью алгоритма AES.
3. Алиса вычисляет значение хэш-функции от конкатенации незашифрованного сообщения и общего секретного ключа.
4. Отправляет зашифрованное сообщение и данное значение хэш-функции участникам группы.
5. Участник группы, например, Боб, получает сообщение и расшифровывает его, после чего также вычисляет значение хэш-функции от конкатенации расшифрованного сообщения и общего секретного ключа. Если оба значения

хэш-функций совпадают, то можно считать, что целостность сообщения подтверждена.

6. Боб расшифровывает сообщение общим ключом.

Получение списка пользователей комнаты

1. Алиса вводит команду /list <room>.
2. На экран выводится список участников комнаты.

Выход из приложения

1. Алиса вводит команду /logout.
2. Всем пользователям, с которым она общалась, с ее стороны рассылается зашифрованное уведомление о выходе из приложения. Аналогичное сообщение отправляется на сервер.
3. Из таблиц пользователей, общавшихся с Алисой, удаляются данные о ней.
4. Из таблицы пользователей на сервере также удаляются данные об Алисе.

ОСОБЕННОСТИ ПРЕДСТАВЛЕННОГО РЕШЕНИЯ

В разработанном приложении невозможно открытие параллельной сессии, т.е. нельзя войти с разных устройств под одним логином одновременно. Конечно, с точки зрения безопасности это можно считать преимуществом, но со стороны простоты использования это является недостатком, так как параллельные сессии удобно использовать в некоторых случаях.

Так как закрытый ключ приглашенного пользователя удаляется сразу после подтверждения его входа, по данному приглашению будет невозможно присоединиться к группе второй раз. Это безусловное преимущество со стороны безопасности, которое одновременно вынуждает каждый раз получать новое индивидуальное приглашение.

Также в комнату нельзя войти по ссылке-приглашению. Это существенно повышает безопасность, так как ссылку-приглашение можно перехватить. Однако это можно считать и недостатком нашего решения — присоединиться к чату в нашем случае достаточно непросто.

После выхода из приложения вся история общения удаляется, а аккаунт пользователя удаляется. Это может быть не очень удобным решением, но в таком случае даже с доступом к устройству злоумышленник не сможет получить историю переписки.

Вошедший в комнату пользователь не может видеть сообщения, которые были отправлены до его входа.

Получение контроля над сервером не позволяет злоумышленнику прочитать сообщения, так как они передаются уже в зашифрованном виде. Ключи хранятся у пользователей, и доступа у сервера к ним нет.

Абсолютная анонимность — ни пользователи, ни сервер не располагают никакой информацией о человеке (например, номером телефона или реальным именем), который пользуется приложением.

ПЛАНЫ ПО РАЗВИТИЮ ПРОЕКТА

В будущем мы планируем продолжить работу над приложением, и ставим перед собой такие задачи, как:

- устранение уязвимости, связанной с пересылкой сообщений во время смены ключа;
- реализация интерфейса для десктопного приложения;
- адаптация приложения под Android;
- реализация длинной арифметики в алгоритмах шифрования;
- реализация возможности пересылки файлов разных форматов (картинки, аудио, видео, документы и т.д.);
- реализация возможности назначения администраторов комнаты, которые смогут удалять или блокировать пользователей;
- контроль потери соединения пользователями;
- реализация возможности выхода из определенной группы, не выходя при этом из приложения;
- реализация возможности совершения видеозвонков;
- оптимизация приложения по памяти и времени работы.

ЗАКЛЮЧЕНИЕ

Результатом работы над проектом TGmini является прототип консольного секретного чата с end-to-end шифрованием и возможностью создания секретных чатов для групп пользователей. Авторами работы был создан и реализован собственный протокол обмена зашифрованными сообщениями в групповых чатах.

При создании секретного чата TGmini авторы тезисов изучили язык программирования Java и основы ООП. Также были подробно разобраны алгоритмы классической криптографии, как асимметричной (RSA, протокол Диффи-Хеллмана [9]), так и симметричной (AES, RC4 [10], шифр Вернама[11]). Кроме того, велась работа с архитектурой REST, HTTP, JSON, URL и URI [4]. Для создания базы данных мы использовали встраиваемую СУБД SQLite [12]. Также использовался фреймворк Spring [13].

ЛИТЕРАТУРА

1. Официальная документация Telegram // URL: https://tlgrm.ru/docs
2. Официальная документация Signal // URL: https://signal.org/docs/
3. Официальная документация WhatsApp // URL: https://scontent.whatsapp.net/v/t61.22868-34/68135620_760356657751682_6212997528851833559_n.pdf/WhatsApp-Security-Whitepaper.pdf?_nc_sid=41cc27&_nc_ohc=qYaIlUD1xLoAX9yPvzQ&_nc_ht=scontent.whatsapp.net&oh=9b188a278311254cde2caeb79c98352b&oe=5F1DF0D3
4. Webber J.,Parastatidis S.,Robinson I. REST in Practice // Printed in the United States of

America, 2010.
5. Нильс Фергюсон, Брюс Шнайер. Практическая криптография = Practical Cryptography: Designing and Implementing Secure Cryptographic Systems. — М. : Диалектика, 2004. — 432 с. — 3000 экз. — ISBN 5-8459-0733-0, ISBN 0-4712-2357-3.Rivest R. The MD5 Message-Digest Algorithm // Internet Engineering Task Force, 1992. — 21 p. [Электронный ресурс] // URL: https://tools.ietf.org/html/rfc1321
6. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — М.: Триумф, 2002. — 816 с. — 3000 экз. — ISBN 5-89392-055-4
7. Rivest R., Rivest R. RFC 1321, The MD5 Message-Digest Algorithm (англ.): The MD5 Message-Digest Algorithm // Request for comments. — Internet Engineering Task Force, 1992. — 21 p. — ISSN 2070-1721 — doi:10.17487/RFC1321
8. Federal Information Processing Standards Publication 197 November 26, 2001 Specification for the ADVANCED ENCRYPTION STANDARD (AES) (англ.)
9. Мао, В. Современная криптография: теория и практика. : Пер. с английского. М.: Издательский дом «Вильямс», 2005. — 768 с. : ил. — Парал. тит. англ. Шифрование — асимметричные методы. Глава 8 («Шифрование с открытым ключом», «Обмен ключом без обмена ключом», «Криптографическая стойкость», «Задача Диффи — Хеллмана и задача дискретного логарифмирования»)
10. Scott R. Fluhrer, Itsik Mantin, Adi Shamir. Weaknesses in the key scheduling algorithm of RC4 // Lecture notes in computer science. — 2001. — Т. 2259. — С. 1—24.
11. Токарева,Н.Н Симметричная криптография. Краткий курс: учебное пособие // Новосиб.гос.ун-т.Новосибирск, 2012. 234 с.
12. Официальная документация SQlite // URL: https://sqlite.org/docs.html
13. Официальная документация Spring Framework // URL: https://github.com/spring-projects/spring-framework

Куратор исследования – Завалишина Елена Владимировна, магистрантка ФИТ НГУ, м.н.с. ИМ СО РАН, сотрудник лаборатории криптографии JetBrains Research.

КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ

Разработка генераторов и мутаторов данных для поиска информации на основе открытых источников

М. Г. Палян[1], Е. А. Кравец[2], М. И. Сергеев[3]

[1]Ереванский государственный университет,
[2]Новосибирский государственный технический университет,
[3]Гимназия №1 Г. Стерлитамак

E-mail**: palyan.m@gmail.com, katherinkravets@mail.ru, enhisir@gmail.com**

В настоящий момент поиск информации по открытым источникам является актуальной темой для многих областей исследований: получение и анализ информации, поиск пропавших людей, поиск конфиденциальных данных, сбор данных о ресурсах в компьютерных сетях, и другой информации. В большинстве случаев для сбора подобной информации требуется большое число ручных действий и аналитики со стороны исследователя. При этом, отличительной особенностью этих действий является потенциальная возможность автоматизации с целью упрощения анализа исследователем. Интерес к данным особенностям и возможность автоматизации действий стали мотивацией для разработки автоматизированного инструмента для поиска и обработки информации из открытых источников в Интернет.

Ключевые слова: генераторы и мутаторы данных, генерация почтовых адресов, генерация форматов телефонных номеров

Возникают ситуации, когда надо получить максимальное количество информации о человеке при помощи минимально возможных входных данных. В таком случае исследователи обычно обращаются к такой технологии как OSINT [1] (англ. Open Source Intelligence) - сбор информации на основе открытых источников. Но в тех случаях, когда полученной информации недостаточно, возникает необходимость самостоятельной генерации готовых данных на их основе.

Мини-модули «Simple Email Generator» [2] и «Phone Number Generator» [3] были разработаны для автоматизированный фреймворка «osint-framework» [4] с целью обеспечения быстрой генерации больших объемов данных на основе уже полученной ранее информации. Благодаря легкой масштабируемости данного фреймворка, каждый из данных модулей может быть запущен как индивидуально из консоли, так и внутри единой системы. Данная информация необходима для расширения спектра поиска среди различных представлений данных.

Мини-модуль «Simple Email Generator»

Мини-модуль «Simple Email Generator» реализован с помощью Python 3 и его стандартных средств, а также встроенных средств «osint-framework», таких как

BaseRunner и ScriptResponse. Основная программа работает следующим образом:

1. Программа получает на вход никнейм объекта исследования;
2. Создается экземпляр класса Runner, наследуемого от BaseRunner;
3. Внутри класса Runner создается экземпляр класса EmailGenerator;
4. Полученный никнейм делится на смысловые части методом EmailGenerator.divide(username);
5. С помощью метода EmailGenerator.generate(username) формируется список потенциальных email-адресов объекта исследования и передается главной программе
6. В качестве результата главная программа возвращает JSON-документ со следующей структурой:
   {“result”: result, “message”: message, “status”: status}
   Данная структура включает в себя набор из трех пар вида “ключ”: значение. Ключ “result” включает в себя конечный результат работы модуля - информацию, которая была получена путем генерации данных.

В рамках дальнейшего развития мини-модуля планируется создание более совершенной генерации данных, а также увеличения количества возможных и сокращения заранее неправильных email-адресов.

Мини-модуль «Phone Number Generator»

Для сбора информации очень часто требуется изменить вид ключей. Мини-модуль «Phone Number Generator» генерирует различные форматы номера телефона. Скрипт реализован с помощью Python 3 и библиотеки phonenumbers[5]. Это версия библиотеки Google, которая разбирает, форматирует и хранит международный номера телефонов, написана на Python. Она поддерживается на Python 2.5-2.7 и Python 3.x. Ее можно установить через систему управления пакетами pip.

Основная программа работает следующим образом:

1. На вход программа получает номер телефона в международном формате, либо локальный номер и название страны (например “RU”, “GB”).
2. С помощью функции phonenumbers.parse отделяет от номера код страны, локальную часть и ведущие нули, и для этого внутри есть специальная структура данных.
3. Полученный объект приводим в стандартный формат через функцию phonenumbers.format_number. Эта функция поддерживает только три формата – NATIONAL, INTERNATIONAL и E164.
4. Все остальные форматы генерируются функцией gen_all, которую реализовали мы.
5. Результат имеет следующую структуру:
   {“message”: message, “result”: result, “status”: status}
   Ключ “result” – массив, который состоит из номеров разных форматов

Конечный результат работы модуля можно использовать для поиска номеров на различных поисковых системах, рассылок, проверки наличия аккаунта пользователя в социальных сетях (если например сайт принимает только локальный формат номера для

регистрации, можно сгенерировать локальный формат из международного).

ЛИТЕРАТУРА И ИСТОЧНИКИ

1. https://ru.wikipedia.org/wiki/OSINT
2. https://github.com/osint-dev-team/osint-framework/tree/email_generator/src/scripts/convert/email_generator
3. https://github.com/osint-dev-team/osint-framework/tree/phone_number_generator/src/scripts/convert/phone_num_generator
4. https://github.com/osint-dev-team/osint-framework
5. https://pypi.org/project/phonenumbers/
6. Michael Bazzell. Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information.
7. Chris Kubecka. Down the Rabbit Hole An OSINT Journey: Open Source Intelligence Gathering for Penetration.

Кураторы исследования:

– Колегов Денис Николаевич, к.т.н., доцент кафедры компьютерной безопасности ТГУ, главный разработчик облачной платформы кибербезопасности компании Bi.Zone (Томск);

– Николаев Антон Анатольевич, студент кафедры компьютерной безопасности ТГУ, разработчик сервисов анализа защищенности Bi.Zone, главный разработчик фрэймворка Grinder (Томск).

Разработка модулей для сбора информации о людях из открытых источников

Е.К. Дубинская[1], С.С. Хлопина[1], О.С. Маркелов[1], Е.Р. Данченкова[2]

[1]Новосибирский государственный технический университет,

[2]МБОУ СШ №6 г. Смоленска

E-mail: **e.dubinskaya@g.nsu.ru, sophiasmile48@gmail.com,**

o.markelov@g.nsu.ru, lizadanchenkova@gmail.com.

В настоящий момент поиск информации по открытым источникам является актуальной темой для многих областей исследований: получение и анализ информации, поиск пропавших людей, поиск конфиденциальных данных, сбор данных о ресурсах в компьютерных сетях, и другой информации. В большинстве случаев для сбора подобной информации требуется большое число ручных действий и аналитики со стороны исследователя. При этом, отличительной особенностью этих действий является потенциальная возможность автоматизации с целью упрощения анализа исследователем. Интерес к данным особенностям и возможность автоматизации действий стали мотивацией для разработки автоматизированного инструмента для поиска и обработки информации из открытых источников в Интернет.

Ключевые слова: сбор информации по открытым источникам, анализ веб-приложений

Сбор информации на основе открытых источников (англ. OSINT, Open Source Intelligence) - это процесс поиска, сбора и анализа информации, полученной из различных открытых, общедоступных источников. К сбору информации на основе открытых источников относят такие дисциплины, как поиск критичных документов и файлов, находящихся в свободном доступе, поиск информации о людях и организациях, поиск пропавших людей, сбор информации о различных веб-приложениях и интернет-ресурсах, анализ открытых баз данных. Данные направления являются особенно актуальными в настоящий момент, так как области применения данных техник очень обширны: помощь правоохранительным органам, поисковые операции, поведенческий анализ преступников на основе открытой информации о них и многие другие.

В первую очередь, OSINT используются в интересах национальной безопасности, правоохранительных органов и бизнес-аналитики. С помощью модулей OSINT может быть составлено достаточно полное представление о человеке или, по крайне мере, могут быть найден ресурсы, по которым можно продолжить поиск. Например, очевидно, что если найдена страница в соцсети, на ней, если повезет, будут лежать открытые данные вроде даты рождения, университета, места работы, родственников, друзей, фотографий и тому подобного.

Стоит отметить, что так как сбор данных осуществляется с использованием open source ресурсов (в большинстве случаев не требующих регистрации пользователя или генерации собственных api-ключей), соответственно, практически любая задача может быть выполнена вручную(например, поиск человека по нику в соцсети), однако этому должен предшествовать поиск ресурсов, с помощью которых можно будет найти требуемую информацию, затем ручной ввод. Не всегда можно такой поиск осуществить быстро, так что наш блок, как и весь фреймворк делает акцент на автоматизации поиска необходимой информации. Так, например, вместо поиска человека по нику в соцсетях, необходимо будет только ввести nickname, а

фреймворк с помощью подключенного модуля осуществит поиск такого ника в популярных соцсетях (twitter, vk, ok, Instagram, github, reddit, tiktok, flickr и других).

В рамках летней школы по Криптографии и информационной безопасности при работе над проектом нашей командой были разработаны следующие модули:

1. **check_nickname** - данный модуль в качестве аргумента принимает nickname человека, при работе модуль проверяет в социальных сетях и форумах существование пользователя с таким же ником. На выходе модуль выдает ссылки на аккаунты с данным никнеймом.
2. **email_verifier** - данный модуль в качестве аргумента принимает email. В результате работы модуль определяет, существует ли такой email на самом деле. Такой модуль работает в паре с модулем, генерирующим пулл возможных email- адресов.
3. **region_check** - модуль в качестве аргумента принимает номер мобильного телефона. Модуль обрабатывает номер телефона и выдает пользователю регион, в котором зарегистрирован заданный номер и оператора связи

Для каждого из вышеперечисленных модулей написаны тесты.

В рамках дальнейшего развития фреймворка, наш проект может быть перенесен на сервер. Также возможно создание полноценного десктопного приложения, для чего можно реализовать графический интерфейс. Также возможно расширение возможностей нашего фреймворка, путем добавления новых модулей. OSINT - довольно-таки большая тема, так что в дальнейшем мы можем реализовать другие модули, такие как: проверка номера телефона в различных базах данных, спам рассылок по данному номеру и email, администрируемые каналы и паблики в социальных сетях, выдача сайтов с упоминанием номера телефона и почты, и т.д.

ЛИТЕРАТУРА

1. URL: requests.readthedocs.io , документация по библиотеке requests для языка python.
2. URL: urllib3.readthedocs.io , документация по библиотеке urllib3 для языка python.
3. URL: yarl.readthedocs.io , документация по библиотеке yarl для языка python.
4. Michael Bazzell. Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information.
5. Chris Kubecka. Down the Rabbit Hole An OSINT Journey: Open Source Intelligence Gathering for Penetration.

Кураторы исследования:
– Колегов Денис Николаевич, к.т.н., доцент кафедры компьютерной безопасности ТГУ, главный разработчик облачной платформы кибербезопасности компании Bi.Zone (Томск);

– Николаев Антон Анатольевич, студент кафедры компьютерной безопасности ТГУ, разработчик сервисов анализа защищенности Bi.Zone, главный разработчик фрэймворка Grinder (Томск).

Разработка фреймворка для поиска информации на основе открытых источников

Н.Д. Крюков[1], Т.Р. Касимов[2], Н.А. Проскурников[3], В.В. Черников[4], В.С. Никифоров[4], А.С. Шапоренко[1]

[1]Томский государственный университет,
[2]Новосибирский государственный технический университет,
[3]Президентский физико-математический лицей №239,
[4]Новосибирский государственный университет,

E-mail: **cravtos92@gmail.com, timurcat7@gmail.com, milaevpro@gmail.com, spykex3@gmail.com, v.nikiforov1@g.nsu.ru , andrew230820@gmail.com**

В настоящий момент поиск информации по открытым источникам является актуальной темой для многих областей исследований: получение и анализ информации, поиск пропавших людей, поиск конфиденциальных данных, сбор данных о ресурсах в компьютерных сетях, и другой информации. В большинстве случаев для сбора подобной информации требуется большое число ручных действий и аналитики со стороны исследователя. При этом, отличительной особенностью этих действий является потенциальная возможность автоматизации с целью упрощения анализа исследователем. Интерес к данным особенностям и возможность автоматизации действий стали мотивацией для разработки автоматизированного инструмента для поиска и обработки информации из открытых источников в Интернет.

Ключевые слова: сбор информации по открытым источникам, анализ веб-приложений

Сбор информации на основе открытых источников (англ. OSINT, Open Source Intelligence) - это процесс поиска, сбора и анализа информации, полученной из различных открытых, общедоступных источников. К сбору информации на основе открытых источников относят такие дисциплины, как поиск критичных документов и файлов, находящихся в свободном доступе, поиск информации о людях и организациях, поиск пропавших людей, сбор информации о различных веб-приложениях и интернет-ресурсах (процесс, также называемый RECON, сокр. от Reconnaissance), анализ открытых баз данных. Данные направления являются особенно актуальными в настоящий момент, так как области применения данных техник очень обширны: помощь правоохранительным органам, поисковые операции, поведенческий анализ преступников на основе открытой информации о них и многие другие.

Автоматизированный фреймворк “osint-framework” был разработан с целью обеспечения быстрого и эффективного поиска информации с помощью открытых источников, баз данных, веб-приложений и API-сервисов. Данный фреймворк является легко масштабируемым за счет системы изолированных модулей (каждый из которых может быть запущен индивидуально из консоли), и позволяет находить различную информацию благодаря обширной базе используемых ресурсов и баз данных. Текущее решение позволяет составить базовую аналитическую картину на базе открытой информации о каком-либо человеке или веб-ресурсе в сети Интернет, что позволяет сократить время рутинного поиска информации для исследователя.

Основной идеей фреймворка является разделение поисков на некоторые обособленные структуры поиска под названием “case” - таким образом, в общем случае поддерживается три категории поиска:

1. BaseCase - общий поиск. Данная структура используется в том случае, когда аргументы и входные данные поиска не могут быть разделены на меньшие, более конкретные категории. Является наиболее общим классом фреймворка и включает в себя следующие два подкласса: OsintCase, ReconCase. Является легко расширяемым за счет возможности создания дочерних классов для новых категорий поиска.
2. OsintCase - поиск информации о людях. Данная структура используется в том случае, когда в качестве входных данных выступает информация о конкретном человеке - такая, как, например, e-mail адрес, телефонный номер, имя пользователя в социальных сетях, полные ФИО и другая схожая информация.
3. ReconCase - поиск информации о различных веб-ресурсах. Данная структура используется в случае необходимости получения информации о конкретных веб-ресурсах по доменному имени или по IP-адресу.

Каждый из представленных выше классов или структур включает в себя соответствующий набор модулей, написанных на языке Python 3. Структура фреймворка позволяет запускать наборы модулей параллельно на любом наборе входных данных, соответствующих выбранной категории поиска - для каждого из классов поиска заданы некоторые наборы аргументов, которые могут быть переданы. Так, например, в случае поиска информации по конкретному человеку с использованием класса OsintCase могут быть переданы такие аргументы, как: “email”, “username”, “fullname”, “phone” и другие.

Параллельное исполнение модулей в текущей реализации фреймворка осуществляется за счет работы каждого модуля в отдельном системном потоке (единовременно), что позволяет расширить возможности модулей и делает их более гибкими - возможность независимого использования асинхронного кода без конфликтов с основной работой фреймворка, возможность использования различных библиотек и системных команд, и другие возможности, применимые для конкретной реализации модуля. Дальнейшее потенциальное расширение и масштабирование фреймворка видится возможным в использовании изолированных процессов вместо потоков, либо в масштабировании за счет системы изолированных docker-контейнеров с общей очередью задач.

Параллельное исполнение модулей возможно внутри одного класса поиска - таким образом, в случае набора задач включающих разные категории поиска (допустим, в случае последовательного поиска различных людей или анализа различных веб-ресурсов в следующей последовательности: OsintCase, OsintCase, ReconCase, OsintCase, …) параллельно модули будут исполняться только в рамках одного класса. Параллельное исполнение нескольких классов поиска возможно путем использования различных решений на уровне самого фреймворка - использование изолированных процессов, очередей задач для параллельного исполнения (с отслеживанием статуса задачи), или изолированных docker-контейнеров.

Набор аргументов для поиска информации не является строго декларированным в общем случае - любой из модулей некоторой категории может принимать на вход любое число необходимых аргументов, но для обеспечения эффективной работы каждый из классов поддерживает валидацию аргументов. Так, например, при использовании OsintCase класса, при получении аргументов, которые не входят в список разрешенных, скрипт будет пропущен.

В качестве примера можно привести следующую ситуацию: в случае ошибочного получения модулем по обработке IP-адреса из класса ReconCase имени пользователя

“username”, валидатор пропустит исполнение данного модуля по причине несоответствия разрешенных аргументов для класса ReconCase и полученного аргумента “username”. В общем случае, при необходимости получения любого набора аргументов без валидации, её можно игнорировать, либо использовать класс BaseCase для поиска по данным свободного формата.

Основной идеей для поиска информации в модулях данного фреймворка является принцип использования open source ресурсов: в большинстве случаев и модулей используются открытые источники, веб-сервисы, API-ресурсы и другие источники, не требующие регистрации пользователя или генерации собственных API-ключей. Тем не менее, в силу некоторых особенностей различных популярных сервисов, некоторые модули принимают на вход также API-ключи к различным сервисам (таким как, например, VK API, Telegram API и другим). В некоторых модулях также используется Selenium WebDriver, который благодаря автоматизации действий браузера позволяет получить нужную информацию с сервисов без использования API-ключей.

Структура каждого из существующих модулей позволяет запускать их отдельно, без использования внешнего фреймворка. Таким образом, каждый модуль можно запустить из консоли с некоторым набором передаваемых позиционных аргументов. Данный метод запуска обеспечивается описанием Runner функции, которая может включать в себя любой пре-процессинг входных данных, их обработку, валидацию и любые другие необходимые функции.

Так, например, модуль, возвращающий код HTTP статуса от веб-ресурса может быть запущен следующим образом:

python3 -m recon.get_host_status https://facebook.com/

Ответом от каждого из модулей является JSON-документ со следующей структурой:

{“result”: result, “message”: message, “status”: status}

Данная структура включает в себя набор из трех пар вида “ключ”: значение. Ключ “result” включает в себя конечный результат работы модуля - информацию, которая была получена из открытых источников и которая представляет ценность для исследователя в контексте поиска. Ключ “message” включает в себя развернутое сообщение о выполненной задаче - данное сообщение может включать в себя некоторые статистические или аналитические данные по полученным результатам. Ключ “status” включает в себя статус выполнения задачи, статус может принимать одно из трех значений для каждой задачи: “success” в случае успешного выполнения, “error” в случае ошибки модуля, “unknown” в случае если статус задачи не был выставлен или результаты не могут быть проанализированы в контексте исполняемой задачи.

В качестве примера данной структуры может послужить ответ от обозначенного ранее модуля “get_host_status” с аргументом для анализа “https://example.com/”, что выступает в качестве анализируемого веб-ресурса.

{“message”: “Got HTTP response status from https://example.com/”, “result”: 200, “status”: “success”}

Другие модули класса ReconCase выполняют следующие задачи:

Модуль **“allowed_methods”** принимает как аргумент URL адрес и проверяет доступность различных HTTP-методов. В результате проверок составляется список методов и ответов сервера на них. Из полученной информации можно сделать выводы, как можно взаимодействовать с сервером и, возможно, обнаружить ошибки в его конфигурации

(например, разрешение на использование небезопасных методов, таких как DELETE, TRACE или PUT).

Модуль **“shodan_favicon_hash”** принимает как аргумент имя хоста и вычисляет хеш-функцию murmurhash3 от favicon.ico файла (иконка веб-сайта). Это позволяет, используя поисковую систему Shodan, узнать реальный IP-адрес серверов, скрытых за прокси, или найти сервера с общей характеристикой, например, для проверки их на известные уязвимости.

Модуль **“ip_info”** принимает как аргумент IP-адрес и предоставляет следующую информацию: континент, страна, регион (область), город, район, почтовый индекс, широта, долгота, часовой пояс, провайдер и хостинг. Это было реализовано с помощью API веб-сайта IP-API.com, который для некоммерческого использования предоставляет свои услуги бесплатно и не требует получения ключа. Данный модуль может быть полезен при поиске людей.

Модуль **“get_title”** принимает в качестве аргумента URL адрес и возвращает его титульник, если таковой имеется. Реализовано это с помощью отправки GET-запроса, после которого сайт возвращает нам ответ в виде HTML. В данном ответе модуль пытается найти тег <title> и вывести его содержимое. Для этого используется регулярное выражение “<title>(.*)</title>”. В случае нахождения строки, которая удовлетворяет данному регулярному выражению, модуль вернет текст заключенный между <title> и </title>, иначе - None. По тексту внутри тега <title> пользователь получает дополнительную информацию о сайте, а также заголовок страницы.

Модуль **“cookie_checker”** принимает в качестве аргумента URL-адрес и возвращает все cookie, устанавливаемые страницей, с проставленными атрибутами “Path”, “Secure”, “HttpOnly”, “Prefix” и “Same-Site”. Реализовано это с помощью отправки GET-запроса, и анализа заголовков ответа, в которых и передаются cookie. Модуль сделан по подобию observatory.mozilla.org, в будущем возможен переход от имеющейся реализации к реализации с использованием API данного сервиса. Также возможно улучшение, проверяющее не один URL, а обходящее все доступные (или некоторую обозримую часть от них) страницы. Таким образом, будут проверены не только cookie с задаваемой страницы, но и cookie сайта в целом. Данный модуль полезен для начальной оценки защищенности веб-приложения.

Модули класса OsintCase выполняют следующие задачи:

Модуль **“check_nickname”** принимает на вход никнейм, по которому осуществляется поиск в различных социальных сетях, список которых находится в файле social_network.txt. Программа работает в многопоточном режиме, используя одновременно 10 потоков, что позволяет ускорить работу программы в несколько раз за счет того, что запросы к сайтам происходят не последовательно, а одновременно. Проверка на существование пользователя с заданным никнеймом осуществляется с помощью отправки GET-запроса на адрес “https:// + домен + / + nickname”. (все социальные сети из social_networks.txt используют именно такую систему адресации страниц пользователей). Данный модуль может быть полезен, если при поиске человека известен только его никнейм в какой-то социальной сети. Люди довольно часто используют один и тот же никнейм, поэтому данный модуль может помочь быстро обнаружить страницы человека в других социальных сетях.

Модуль **“email_verifier”** в качестве аргумента принимает адрес электронной почты и проверяет, существует ли данный адрес, используя библиотеку verify_email, которая эффективно проверяет доменное имя и отправляет команду ping на обработчик, чтобы проверить существование нашего адреса. В отличие от validate-email, email_verifier не просто проверяет правильность написанного адреса, но и смотрит на то, зарегистрирован ли пользователь с таким адресом ранее. Особенности: синтаксическая проверка, проверка MX

(Mail Exchange records), кэширование поиска доменов для повышения производительности, поддержка asyncio параллелизма.

В рамках дальнейшего развития фреймворка возможно создание REST API веб-сервиса, работающего на базе одного из веб-фреймворков, таких как Tornado или Aiohttp. В рамках использования данной утилиты как веб-сервиса предполагается возможность реализации очередей задач, где каждой текущей поисковой задаче (из класса BaseCase, OsintCase, ReconCase) будет присвоен уникальный идентификатор, благодаря которому можно будет забирать результаты при завершении сканирования, а также узнавать статус текущей задачи.

В качестве оптимизации и ускорения развертывания системы на других системах или вычислительных кластерах сервис может быть контейнеризирован с помощью docker-контейнеров на базе одного из легковесных для сборки образов, таких как Alpine Linux. Кроме того, данный способ дает возможность легкого масштабирования системы в дальнейшем.

Планируется разработка новых модулей, таких как, например, поиск в социальных сетях с помощью хеширования никнейма различными алгоритмами (MD5, MD4, SHA-1, SHA-256, SHA-512 и т.д.). Это может оказаться довольно полезным, т.к. некоторые социальные сети хранят хранят адрес страницы пользователя в виде “https:// + домен + / + hash(nickname)”. Еще один планируемый модуль - это проверка наличия аккаунтов в различных соцсетях, зарегистрированных на данный номер телефона. Возможно создание модуля для проверки открытых портов по заданному IP и предположения о работающих там сервисах.

ЛИТЕРАТУРА

1. https://docs.python.org/3/
2. https://requests.readthedocs.io/en/master/
3. https://selenium-python.readthedocs.io/
4. https://github.com/hajimes/mmh3
5. https://docs.aiohttp.org/en/stable/
6. https://docs.python.org/3/library/asyncio.html/
7. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
8. https://github.com/kakshay21/verify_email/
9. Michael Bazzell. Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information.
10. Chris Kubecka. Down the Rabbit Hole An OSINT Journey: Open Source Intelligence Gathering for Penetration.

Кураторы исследования:
– Колегов Денис Николаевич, к.т.н., доцент кафедры компьютерной безопасности ТГУ, главный разработчик облачной платформы кибербезопасности компании Bi.Zone (Томск);
– Николаев Антон Анатольевич, студент кафедры компьютерной безопасности ТГУ, разработчик сервисов анализа защищенности Bi.Zone, главный разработчик фрэймворка Grinder (Томск).

Список авторов:

1. **Аламов Владимир Александрович** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 5 курс специалитета
2. **Атутова Наталья Дмитриевна** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
3. **Ахтямов Данил Айдарович** – The Hebrew University of Jerusalem, магистрант 1-го курса, Иерусалим (Израиль)
4. **Базаров Андрей Алдарович** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
5. **Бахарев Александр Олегович** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
6. **Бонич Татьяна Андреевна** – Механико-математический факультет, Новосибирский государственный университет, 1 курс магистратуры
7. **Быков Денис Александрович** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
8. **Валитов Андрей Александрович** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
9. **Горяйнова Анастасия Павловна** – Институт математики и компьютерных наук, Тюменский государственный университет, 3 курс бакалавриата
10. **Данченкова Елизавета Руслановна** – Средняя общеобразовательная школа №6 г. Смоленска, 11 класс
11. **Диденко Андрей Антонович** – Областная специализированная школа лицей для детей одаренных в области математики, физики и информатики, 11 класс, г. Усть-Каменогорск (Казахстан)
12. **Доронин Артемий Евгеньевич** – Механико-математический факультет, Новосибирский государственный университет, 2 курс магистратуры
13. **Дубинская Екатерина Константиновна** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
14. **Евсюков Анатолий Павлович** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
15. **Жантуликов Булат Фаритович** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
16. **Желтова Кристина Анатольевна** – Институт информатики и телекоммуникаций, Сибирский государственный университет науки и технологий им. М.Ф. Решетнева, 3 курс бакалавриата
17. **Завалишина Елена Владимировна** – Факультет информационных технологий, Новосибирский государственный университет
18. **Запольский Максим Михайлович** – Механико-математический факультет, Новосибирский государственный университет, 3 курс бакалавриата
19. **Зюбина Дарья Александровна** – Факультет информационных технологий, Новосибирский государственный университет, 3 курс бакалавриата
20. **Ищукова Евгения Александровна** – Южный федеральный университет, доцент кафедры

Безопасности информационных технологий

21. **Карнаухова Виктория Олеговна** – Факультет информационных технологий, Алтайский государственный технический университет, 1 курс бакалавриата
22. **Касимов Тимур Рустамович** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 2 курс бакалавриата
23. **Ким Станислав Евгеньевич** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 3 курс бакалавриата
24. **Котельникова Анна Александровна** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата
25. **Кравец Екатерина Александровна** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 2 курс бакалавриата
26. **Крюков Никита Дмитриевич** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 1 курс специалитет
27. **Лаханский Алексей Андреевич** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
28. **Леонович Дарьяна Александровна** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 2 курс бакалавриата
29. **Ляпич Никита Сергеевич** – Муниципальное автономное общеобразовательное учреждение Лицей 6, г. Бердск, выпускник
30. **Маро Екатерина Александровна** – Южный федеральный университет, доцент кафедры Безопасности информационных технологий.
31. **Маркелов Олег Сергеевич -** Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
32. **Матеюк Илья Анатольевич** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
33. **Натарова Ксения Витальевна** – Факультет радиотехники и кибернетики, Московский физико-технический институт, 1 курс бакалавриата
34. **Никифоров Владислав Сергеевич** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
35. **Палян Марине Гургеновна** – Факультет информатики и прикладной математики, Ереванский государственный университет, 1 курс магистратуры. Ереван(Армения)
36. **Панферов Матвей Андреевич** – Механико-математический факультет, Новосибирский государственный университет, 1 курс магистратуры
37. **Парфенов Денис Романович** – Факультет информационных технологий, Новосибирский государственный университет, 3 курс бакалавриата
38. **Побединский Сергей Юрьевич** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 2 курс бакалавриата
39. **Помыкалов Савелий Витальевич** – Средняя общеобразовательная школа №2, 11 класс, г. Луховицы
40. **Проскурников Никита Андреевич** – Президентский физико-математический лицей 239, 11 класс, г. Санк-Петербург

41. **Разенков Семен Игоревич** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 2 курс специалитета
42. **Раимбеков Азим Русланович** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 3 курс специалитета
43. **Сафенрейтер Дмитрий Алексеевич** – Факультет информационных технологий, Новосибирский государственный университет 2 курс бакалавриата
44. **Семенова Екатерина Вадимовна** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 4 курс специалитета
45. **Сергеев Алексей Вячеславович** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
46. **Сергеев Матвей Игоревич** – Гимназия №1, 10 класс, г. Стерлитамак (респ. Башкоркостан)
47. **Скудина Виктория Викторовна** – Механико-математический факультет, Новосибирский государственный университет, 1 курс бакалавриата
48. **Софронова Дарья Алексеевна** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
49. **Сутормин Иван Александрович** – Механико-математический факультет, Новосибирский государственный университет, 4 курс бакалавриата
50. **Титова Ксения Максимовна** – Механико-математический факультет, Новосибирский государственный университет, 3 курс бакалавриата
51. **Трацевский Игорь Дмитриевич** – Институт математики и компьютерных наук, Тюменский государственный университет, 3 курс специалитета
52. **Хильчук Ирина Сергеевна** – Механико-математический факультет, Новосибирский государственный университет, НГУ, 4 курс бакалавриата
53. **Хлопина София Сергеевна** – Факультет прикладной математики и информатики, Новосибирский государственный технический университет, 1 курс бакалавриата
54. **Ходзицкий Артем Федорович** – Механико-математический факультет, Новосибирский государственный университет, 2 курс бакалавриата
55. **Филиппов Степан Дмитриевич** – Санкт-Петербургский государственный университет, 2-й курс
56. **Черников Василий Викторович** – Факультет информационных технологий, Новосибирский государственный университет, 2 курс бакалавриата
57. **Чхайло Иван Дмитриевич** – Институт математики и компьютерных наук, Тюменский государственный университет, 3 курс специалитета
58. **Шапоренко Андрей Сергеевич** – кафедра Компьютерной Безопасности, Томский государственный университет, 2-й курс
59. **Щербина Даниил Алексеевич** – Институт прикладной математики и компьютерных наук, Томский государственный университет, 3 курс специалитета
60. **Эйсвальд Юлия Игоревна** – Факультет информационных технологий, Новосибирский государственный университет, 1 курс бакалавриата