Летняя школа-конференция "Криптография и информационная безопасность" 2023

Сборник тезисов

Cryptographic Center (Novosibirsk)

Kovalevskaya North-West Centre of Mathematical Research

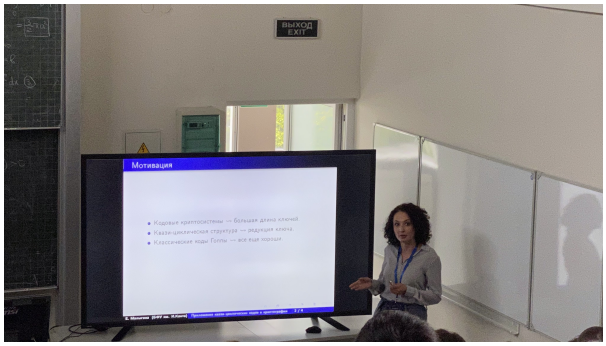
MATHEMATICAL CENTER IN AKADEMGORODOK

Novosibirsk State University *THE REAL SCIENCE

NSU Non stop University CRYPTO

Калининград 2023


ВЫХОД
EXIT


ВЫХОД
EXIT
Мотивация

Содержание

О школе

Летняя школа «Криптография и информационная безопасность» — традиционное мероприятие, организуемое НГУ и Международным математическим центром. В этом году Летняя школа организуется в сотрудничестве с БФУ им. И.Канта и на его базе.

Организаторами школы-конференции выступают Криптографический Центр (Новосибирск), Новосибирский государственный университет, Международный математический Центр в Академгородке, Северо-Западный центр математических исследований имени Софьи Ковалевской (БФУ им. И. Канта) и организаторы международной олимпиады NSUCRYPTO.

Даты проведения: 7 августа - 17 августа 2023.

Формат участия: очный.

Студенты принимали участие в лекциях, командной и индивидуальной работе в проектах, связанной с решением исследовательских задач в области криптографии и информационной безопасности, спортивных занятиях. Одно из важнейших событий школы-конференции – круглый стол по современным проблемам криптографии. Темы проектов разнообразные, особое внимание уделено постквантовой криптографии. В школе принимают участие ведущие разработчики РФ в данной области, а именно – команды, непосредственно участвующие в создании стандартов в этой области.

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

Руководители школы: - к.ф.-м.н. Токарева Наталья Николаевна, к.ф.-м.н., доцент Новосибирского государственного университета, руководитель Криптографического центра (Новосибирск), председатель международной олимпиады Non-Stop University CRYPTO;

– к.ф.-м.н. Малыгина Екатерина Сергеевна, доцент ОНК «Институт высоких технологий», научный сотрудник лаборатории «Защиты и обработки информации» НОМЦ «Северо-Западный центр математических исследований им. С.Ковалевской» БФУ им. И.Канта.

Лекторы и преподаватели школы:

- **БАХАРЕВ Александр Олегович** – преподаватель Новосибирского государственного университета;
- **ГРЕБНЕВ Сергей Владимирович** – ведущий криптограф-исследователь QApp, Российского квантового центра (Москва);
- **ЗАИКИН Олег Сергеевич** – к.т.н., ведущий научный сотрудник Института динамики систем и теории управления (Иркутск), специалист в области криптографии и SAT-вычислений, создатель лучшего SAT-решателя в 2019 году (по результатам международного конкурса SATrace);
- **КИРШАНОВА Елена Алексеевна** – доцент ОНК "Институт высоких технологий";
- **КОЛОМЕЕЦ Николай Александрович** –к.ф.-м.н., старший преподаватель Новосибирского государственного университета, специалист в области симметричной криптографии и криптоанализа;
- **КОНДЫРЕВ Дмитрий Олегович** – преподаватель Новосибирского государственного университета, специалист в области блокчейн-технологий;
- **КОСТОЧКА Светлана Владимировна** – спортивный тренер летней школы;
- **КУЦЕНКО Александр Владимирович** – преподаватель Новосибирского государственного университета, специалист в области постквантового криптоанализа;
- **МАЛЫГИНА Екатерина Сергеевна** – к.ф.-м.н., доцент ОНК «Институт высоких технологий», научный сотрудник лаборатории «Математические методы защиты и обработки информации» НОМЦ «Северо-Западный центр математических исследований им. С.Ковалевской» БФУ им. И.Канта;
- **МАРО Екатерина Александровна** – к.т.н., доцент Института компьютерных технологий и информационной безопасности, Южный федеральный университет;
- **МЕЛЬНИЧУК Евгений Михайлович** – – ассистент Института физико-математических наук и информационных технологий, БФУ им. Канта;
- **НОВОСЕЛОВ Семен Александрович** – младший научный сотрудник лаборатории "Математические методы защиты и обработки информации старший преподаватель с ученой степенью кандидата наук ОНК "Институт высоких технологий";
- **ТОКАРЕВА Наталья Николаевна** – к.ф.-м.н., доцент Новосибирского государственного университета, руководитель Криптографического центра (Новосибирск), председатель международной олимпиады Non-StopUniversity CRYPTO;
- **ЧИЖОВ Иван Владимирович** – к.ф.-м.н., доцент Московского государственного университета, специалист в области постквантовой криптографии;
- **ХИЛЬЧУК Ирина Сергеевна** – преподаватель Новосибирского государственного университета;

- **ЦАРЕГОРОДЦЕВ Кирилл Денисович** – специалист-исследователь, АО «НПК «Криптонит»;
- **ШАПОРЕНКО Александр Сергеевич** – преподаватель Новосибирского государственного университета.

Организационный комитет:

- Токарева Наталья Николаевна;
- Малыгина Екатерина Сергеевна;
- Кочеткова Валерия Кирилловна;
- Хильчук Ирина Сергеевна;
- Новоселов Семен Александрович;
- Калгин Константин Викторович.

Лекции:

1. Криптосистема RSA: устройство и практический криптоанализ. Часть 1 (Гребнев С.В.)
2. Криптосистема RSA: устройство и практический криптоанализ. Часть 2 (Гребнев С.В.)
3. Методы аутентификации пользователей и их уязвимости (Маро Е.А.)
4. Блокчейн: составные компоненты и принципы работы (Кондырев Д.О.)
5. Zero-knowledge proof и блокчейн. ZK-Rollups (Мельничук Е.М.)
6. Введение в SAT-криптоанализ (Заикин О.С.)
7. Квазигруппы в криптографии: введение (Царегородцев К.Д.)
8. Применение атак по побочным каналам на криптографические системы защиты информации (Маро Е.А.)
9. Введение в квантовые вычисления и квантовый алгоритм Шора целочисленной факторизации (Чижов И.В.)
10. Что такое постквантовая криптография и зачем она нужна? (Чижов И.В.)
11. Сложные задачи из теории линейных кодов в постквантовой криптографии (Чижов И.В.)
12. Постквантовая криптография: когда что-то пошло не так (Гребнев С.В.)
13. Теоретико-кодовые криптосистемы с открытым ключом (Чижов И.В)
14. Криптоанализ криптосистемы Мак-Элиса (Малыгина Е.С.)
15. Трудные задачи на решетках и их приложения в современных постквантовых стандартах (Киршанова Е.А.)
16. Булевы функции в криптографии (Шапоренко А.С.)
17. Механизмы анонимизации в сети Bitcoin: протокол Taproot и Coinjoin (Мельничук Е.М.)
18. SAT-криптоанализ криптографических хэш-функций и поточных шифров (Заикин О.С.)
19. Имитовставка: что это и как работает (Коломеец Н.А.)
20. Квазигруппы в криптографии: примеры криптосистем (Царегородцев К.Д.)

Программа конференции

17 августа 2023г., начало в 09:00

1. М.А. Белов, Н.Н. Сероштан, Ю.П. Леонтьева, Ю.В. Хорева, А.Н. Ибрагимова **"Построение булевых функций с оптимальной алгебраической иммунностью в сочетании с другими свойствами"** (куратор – И.С. Хильчук)

2. М.А. Арбейтер, К.П. Якушенокс **"Задача синдромного декодирования кодов общего положения"** (кураторы – И.В. Чижов, Н.А. Коломеец)

3. А.А. Бучнев **"Задача эквивалентности линейных кодов"** (кураторы – И.В. Чижов, Н.А. Коломеец)

4. В.В. Давыдов, А.П. Кирьянова **"SAT-криптоанализ криптографических хэш-функций - финалистов конкурса NIST"** (куратор – О.С. Заикин)

5. Д.А. Соколова, Д.А. Быков **"SAT-криптоанализ некоторых поточных и блочных шифров"** (кураторы – О.С. Заикин, Е.А. Маро)

6. И.Д. Иогансон, А.Г. Леевик, К.И. Зименко **"Разработка эвристических методов криптоанализа постквантовых криптосистем"** (кураторы – А.В. Куценко, А.О.Бахарев)

7. Е.А. Рубан, А.В. Быстрых, Е.А. Галушка, Е.В. Новикова, С.А. Жданкина **"Изучение свойств криптографических протоколов"** (кураторы – С.В. Гребнев, И.С. Хильчук)

8. Р.Т. Еремеев, С.И. Ларионова, В.А. Далевич, В.А. Башкиров, Р.С. Кругляков, Е.Д. Фролов **"Обеспечение приватности данных в смарт-контрактах"** (куратор – Д.О. Кондырев)

9. А.А. Блохин, В. Д. Боязитов, А.С. Евсеев, Д.В. Енин, А.А. Ерохина, Ф.Е. Пивоваров, Р.Д. Тогумбаев **"Реализация системы анонимных платежей на базе сети Ethereum с использованием технологии доказательств с нулевым разглашением"** (куратор – Е.М. Мельничук)

10. А.Ф. Хуцаева, А.В. Исайчева **"Анализ и реализация протокола конфиденциальных активов"** (куратор – Е.М. Мельничук)

11. А.А. Голованов, С.А. Ковалева, Ле Куок Зунг **"Векторные булевы функции с аффинно эквивалентными компонентами"** (куратор – Н.А. Коломеец)

12. А.С. Шапоренко, Я.Е. Дрожжина, А. М.Мамаджанова **"Решение проблемы масштабируемости блокчейна с помощью ZK-rollups"** (куратор – Д.О. Кондырев)

13. А.А. Голяшов, В.О. Гурьянов, А.А. Мухортова, О.Ю. Немова **"Изучение свойств подстановок, порождаемых сдвиговыми преобразованиями"**(куратор – К.Д. Царегородцев)

14. А.А. Кунинец, Е.А. Калинина, К.П. Якушенокс, А.А. Нецветайлов **"Приложение квазициклических кодов в криптографии"** (куратор – Е.С. Малыгина)

15. Н.А. Якунин **"Криптографические свойства булевых функций"** (куратор – А.С. Шапоренко)

16. Г.Д. Малютин, А.Д. Рудзянский, Н.А. Глущенко **"Анализ утечек по энергопотреблению для различных реализаций симметричных блочных алгоритмов шифрования"** (куратор – Е.А.Маро)

17. Д.А. Котов, А.Д. Рудзянский **"Уменьшение времени работы постквантовых криптосистем, основанных на решётках"** (кураторы – А.В, Куценко, А.О. Бахарев)

18. Д.А. Попков, К.А. Лернер, К.А. Волков, А.Б. Мудрич, Ю.Ф. Болтнев**"Практический криптоанализ RSA"** (кураторы – С.В, Гребнев, А.С. Шапоренко)

19. А.А. Чубань, Н.А. Попов, А.И. Сацута **"Оптимизация вычисления целочисленного квадратного корня в смартконтракте пула ликвидности"** (куратор – Е.М. Мельничук)

20. **Закрытие Летней школы.**

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Изучение свойств криптографических протоколов

С. А. Жданкина[1], Е. В. Новикова[2], А. В. Быстрых[2], Е. А. Рубан[2], Е. А. Галушка
[1]Дальневосточный государственный университет путей сообщения
[2]Национальный исследовательский Томский государственный университет
E-mail: s-zhdankina01@mail.ru, novikova.ev99@yandex.ru, tri_de@inbox.ru, egor-ruban@mail.ru, katyafrmrussia@gmail.com

Аннотация

Целью работы является исследование свойств безопасности криптографических протоколов. Для исследования выбраны протоколы выработки общего ключа, а именно отечественный протокол Лимонник-3 [2] и базовая версия протокола SIGMA [9], входящего в набор протоколов для обеспечения защиты данных, передаваемых по межсетевому протоколу IP, IPsec. В качестве метода исследования использован формальный анализ протоколов программными продуктами верификации протоколов. В работе изучены следующие программы для проверки свойств безопасности протоколов: Scyther, Avispa, Verifpal. По результатам верификации указанных протоколов несоблюдение указанных свойств безопасности не выявлено.
Ключевые слова: *протоколы совместной выработки ключа, scyther, verifpal, avispa, Лимонник-3, SIGMA*

Развитие методов криптографии обеспечивает увеличение сохранности и конфиденциальности данных. Однако существуют возможности утечки информации, которые происходят не в силу математических слабостей криптографических методов, а в силу особенностей передаваемых сообщений и содержащейся в них информации. Некоторые уязвимости могут возникнуть при обмене данными между сторонами, происходящими по правилам выбранного сторонами протокола.

Определение 1. Протокол - описание распределенного алгоритма, в процессе выполнения которого два (или более) участника последовательно выполняют определенные действия и обмениваются сообщениями [1].

Безопасность протокола выражается в обеспечении гарантий выполнения таких свойств, характеризующих безопасность, как доступность, конфиденциальность, целостность и др. Для обеспечения функций безопасности, отвечающих этим свойствам, в протоколах применяют специальные конструкции. Протокол, обеспечивающий поддержку хотя бы одной из функций безопасности, называют защищенным [1].

Существуют разные классы протоколов с различными целевыми назначениями. Протоколы, относящиеся к одному из них - классу протоколов выработки общего ключа, рассмотрены в данной работе.

Определение 2. Протоколы выработки общего ключа - это протоколы, предназначенные для выработки ключа, известного лишь двум абонентам, как правило, не имеющим заранее распределенной ключевой информации. Выработанный ключ может быть использован для обеспечения конфиденциальности канала связи между этими абонентами.

Наиболее известным представителем семейства протоколов открытого распределения ключей является схема Диффи-Хеллмана. У данного протокола имеется серьезный недостаток, позволяющий осуществить атаку типа "человек посередине". Эта атака становится возможной благодаря

отсутствию в протоколе механизмов аутентификации. Эту проблему решают протоколы аутентифицированной выработки общего ключа, проверка свойств безопасности которых является целью данного исследования.

Для проверки свойств безопасности криптографических протоколов существует специальное программное обеспечение. Использование таких инструментов обусловлено следующими причинами:

1. автоматизированные средства способствуют уменьшению объема ручной работы;
2. благодаря использованию вычислительных мощностей ЭВМ уменьшается время, отводимое на проверку протоколов;
3. автоматизированные программные продукты сокращают количество ошибок, обусловленных человеческим фактором.

В работе изучаются следующие программы для проверки свойств безопасности протоколов: Scyther, Avispa, Verifpal. Во всех рассматриваемых средствах автоматизированной верификации протоколов все криптографические методы шифрования, хэширования и пр. считаются абсолютно стойкими.

Инструменты анализа протоколов

Avispa - инструмент для автоматизированного анализа безопасности криптографических протоколов. Данный инструмент использует для описания протоколов язык HLPSL (High Level Protocol Specification Language) [6].

Свойства безопасности:

- secrecy_of - сохранение в секрете информации, разделяемой внутри группы участников, от остальных участников;
- authentication_on - строгая аутентификация участников по некоторой информации,
- weak_authentication_on - слабая аутентификация агентов по некоторой информации.

Scyther — это инструмент, предназначенный для формального анализа протоколов безопасности, их свойств безопасности и потенциальных уязвимостей. Scyther использует язык SPDL (Simple Protocol Definition Language), который имеет C/Java-подобный синтаксис. Свойства безопасности[4][5]:

1. Aliveness - При выполнении этого свойства, протокол гарантирует, что после завершения обмена сообщениями инициатор протокола может удостоверится в в личности своего собеседника.
2. Weak agreement - является усилением свойства Aliveness, при котором выполняется аутентификация инициатора перед собеседником.
3. В данном примере выполняется свойство Aliveness для B, так как только B мог расшифровать Na, однако злоумышленник мог подменить идентификатор A в первом сообщении, заставив B думать, что B общается со злоумышленником, что не соответствует свойству weak agreement.

4. Secret / SKR – Указывает на то, что не существует такой последовательности обмениваемых сообщений, при которых активный атакующий может получить секретную информацию.

5. Niagree - Протокол гарантирует инициатору неинъективное согласие с собеседником по поводу сообщения M если Алиса считает, что общалась в Бобом, Боб считает, что общался с Алисой и они договорились по поводу сообщения M.Свойство проедпологает защиту от комбинированных атак, в которых сообщения содержат информацию взятую из других сообщений.

6. Nisynch - протокол гарантирует участникам, что они общались друг с другом, а также то, что они договорились по поводу всех передаваемых данных. Свойство проедпологает защиту от атак, в которых используются сообщения образованные не в рамках текущей сессии.

Verifpal - программное обеспечение для проверки безопасности криптографических протоколов. Язык Verifpal предназначен для описания протоколов на высоком уровне абстракции, при этом он является достаточно точным и выразительным для формального моделирования[8].

Свойства безопасности[8]:

1. confidentiality - свойство, при котором проверяется возможность активного злоумышленника расшифровать секретный ключ, если он может выдавать себя за инициатора или его собеседника.

2. authentication (Bob → Alice) gbs- свойство, при котором моделируется сценарий, в котором подписанный участником предварительный ключ мог быть подписан активным злоумышленником, использовавшим закрытый ключ подписи, который он контролирует. Затем активный злоумышленник может подменить открытый ключ долговременной подписи участника своим собственным, когда он отправляется к инициатору протокола, что приведет инициатора к успешной проверке подписи под вредоносным открытым ключом.

3. authentication (Alice → Bob) - свойство, гарантирующее невозможность злоумышленника выдать себя участнику протокола за инициатора путем атаки "человек посередине".

4. authentication (Bob → Alice) - свойство, гарантирующее невозможность злоумышленника выдать себя инициатору протокола за собеседника путем атаки "человек посередине".

5. confidentiality - свойство, предпологающее невозможность злоумышленника выдавать себя за собеседника участика протокола для всех участников.

Анализ протоколов

Лимонник-3

Лимонник-3[2] (Л-3) — протокол выработки общего ключа с возможностью использования двух различных эллиптических кривых и с двусторонней аутентификацией при помощи ключа схемы Диффи-Хеллмана. Схематичное изображение Л-3 приведено на рисунке (рис.1).

Стороны заранее договариваются о параметрах используемых эллиптических кривых: P_A, P_B, c_A, c_B. Затем в процессе взаимодействия в рамках протокола обмениваются идентификаторами ID_A, ID_B, сертификатами $Cert_A$, $Cert_B$ с долговременными открытыми ключами Sa и Sb, заверенными удостоверяющим центром, открытыми сеансовыми ключами Ka и Kb. Стороны также

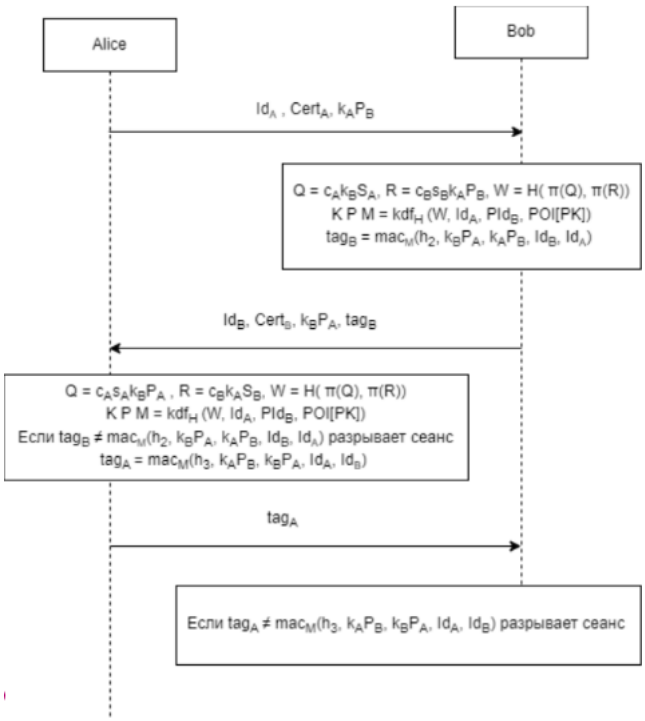


Рис. 1: Схема протокола Лимонник-3

вычисляют две метки подтверждения ключа: tag_A и tag_B, посылают каждый свое значение и сверяют пришедший от собеседника с вычисленным для него значением. В результате в случае удачной аутентификации у сторон формируется общий секретный ключ.

Формальный анализ данного протокола не выявил уязвимостей для выбранных при анализе требований безопасности. В [3] проводился анализ протокола для другого набора требований, критических уязвимостей также не было выявлено.

SIGMA

SIGMA (SIGn-and-MAc)[9] - семейство протоколов выработки общего ключа, представляющих общий подход к построению аутентифицированных протоколов Диффи-Хеллмана с использованием сочетания механизмов цифровой подписи и кода аутентификации сообщений.

Схематичное изображение самой простой формы SIGMA (без защиты идентичности) приведено на рисунке (рис. 2).

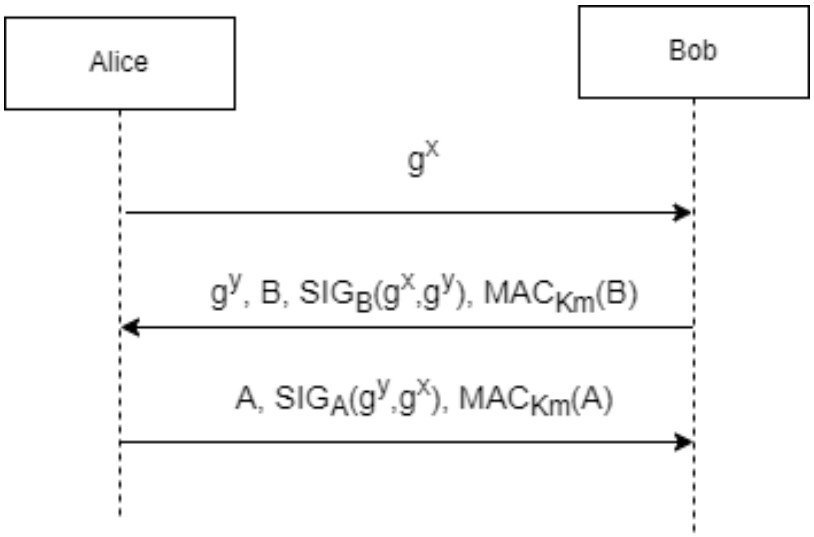


Рис. 2: Схема протокола SIGMA

Стороны обмениваются сеансовыми открытыми ключами g^x, g^y, своими идентификаторами A и B, подписанными сеансовыми открытыми ключами (каждый своей), кодом аутентификации сообщений с использованием общего секретного ключа, выработанного по схеме Диффи-Хеллмана $K_m = g^{xy}$.

Формальный анализ данного протокола не выявил уязвимостей для выбранных при анализе требований безопасности. Результаты об автоматизированном анализе рассмотренной конкретной версии SIGMA нам не встречались. Формальный анализ протокола SIGMA в [7] без автоматизированных средств показал, что базовая конструкция SIGMA и ее варианты защищены в рамках теоретико-сложной модели безопасности.

ЛИТЕРАТУРА

[1] Черемушкин, А. В. Криптографические протоколы: основные свойства и уязвимости // ПДМ. Приложение. 2009. №2.

[2] Гребнев, С. В. О криптографических свойствах схемы выработки общего ключа // Безопасность информационных технологий. 2019. №26(2). С. 6-20.

[3] A. M. Semenov, Analysis of Russian key-agreement protocols using automated verification tools, Mat. Vopr. Kriptogr., 2017, Volume 8, Issue 2, 131–142.

[4] G. Lowe, "A hierarchy of authentication specifications,"Proceedings 10th Computer Security Foundations Workshop, Rockport, MA, USA, 1997, pp. 31-43, doi: 10.1109/CSFW.1997.596782.

[5] Cas Cremers and Sjouke Mauw. Operational Semantics and Verification of Security Protocols. Information Security and Cryptography. Springer, 2012

[6] The AVISPA team, The High Level Protocol Specification Language. [Электронный ресурс] // URL: http://www.avispa-project.org/ (дата обращения: 16.08.2023).

[7] Canetti, R., and Krawczyk, H., "Security Analysis of IKE's Signature-based Key Exchange Protocol", Crypto 2002. LNCS Vol. 2442. Full version in: Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2002/120.

[8] Nadim Kobeissi - Verifpal User Manual first edition [Электронный ресурс] // URL :https://verifpal.com/software/ (дата обращения: 16.08.2023).

[9] Krawczyk, H. (2003). SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In: Boneh, D. (eds) Advances in Cryptology - CRYPTO 2003. CRYPTO 2003. Lecture Notes in Computer Science, vol 2729. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45146-4_24

Кураторы исследования –

Гребнев Сергей Владимирович – ведущий криптограф-исследователь QApp, Российский квантовый центр (г. Москва);

Хильчук Ирина Сергеевна – преподаватель Новосибирского государственного университета.

КРИПТОГРАФИЧЕСКИЕ БУЛЕВЫ ФУНКЦИИ

Векторные булевы функции с аффинно эквивалентными компонентами

А. А. Голованов[1], С. А. Ковалева[1], К. З. Ле[2]
[1]Университет ИТМО, г. Санкт-Петербург
[2]Национальный исследовательский университет МЭИ, г. Москва
E-mail: agolovanov2403@gmail.com, kovaleva.sofya2708@yandex.ru, dunglq@yandex.ru

Аннотация

В данной работе была исследована возможность построения взаимно однозначной векторной булевой функции, все компоненты которой аффинно экивалентны некоторой заданной булевой функции f. Показано, что при большом числе переменных это не всегда возможно, выдвинута гипотеза о существовании таких f среди квадратичных сбалансированных функций от чётного числа переменных.

Ключевые слова: *булевы функции, векторные булевы функции, аффинная эквивалентность, квадратичная функция.*

Определения

Булева функция — функция $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$. Любую булеву функцию можно представлять (и единственным образом) с помощью операций сложения, умножения и константы 1. Такое представление называется *алгебраической нормальной формой (АНФ)* или *полиномом Жегалкина*:

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

где при каждом k все индексы $i_1, \ldots, i_k$ различны и параметры $a_0, a_{i_1}, \ldots, a_{i_k}$ принимают значения 0 или 1.

Булевы функции f и g от n переменных называются *аффинно эквивалентными*, если существует двоичная невырожденная матрица A размера $n \times n$ и вектор b из $\mathbb{F}_2^n$, такие что

$$g(x) = f(xA \oplus b) \text{ для всех } x \in \mathbb{F}_2^n. \tag{2}$$

Весом булевой функции f является количество её ненулевых значений, он обозначается через $wt(f)$. Булева функция f от n переменных называется *сбалансированной*, если $wt(f) = 2^{n-1}$.

Функция вида $\mathbb{F}_2^n \to \mathbb{F}_2^m$ называется *векторной булевой функцией*. Векторная булева функция F от n переменных представима в виде $F(x) = (f_1(x), \ldots, f_m(x))$, где f_i — булева функция от n переменных. Булевы функции $f_1, \ldots, f_m$ называются *координатными*. *Компонентами* F (*компонентными функциями*) называют все непустые линейные комбинации её координатных функций, т. е. $\langle v, F \rangle$, где $v \in \mathbb{F}_2^n \setminus \{0\}$.

Описание задачи

Конструкция 1. Вход: сбалансированная булева функция f от n переменных. Выход: векторная функция $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$, все компонентные функции которой аффинно эквивалентны f.

Известно, что F взаимно однозначна тогда и только тогда, когда каждая её компонентная функция сбалансирована. Таким образом, порождаемая конструкцией F всегда взаимно однозначна.

Цель работы — исследовать возможность по заданной сбалансированной $f : \mathbb{F}_2^n \to \mathbb{F}_2$ построить F с помощью конструкции 1. Отметим, что конструкция нетривиальна. Нетрудно взять координатные функции F, аффинно эквивалентные f. Однако, их суммы уже могут не быть аффинно эквивалентными f. Примеры f, по которым всегда можно построить F по конструкции 1:

- аффинная f, не являющаяся константой;
- мономиальная f [7].

Используя результаты [5, 6], можно доказать следующую теорему.

Теорема 1. *Существует n_0, такое что для всех $n \geq n_0$ существует сбалансированная булева функция f от n переменных, по которой нельзя построить F конструкцией 1. При нечётном n можно взять $n_0 = 15$.*

Квадратичные функции

Так как количество сбалансированных булевых функций большое ($C_{2^n}^{2^{n-1}}$ для функции от n переменных), мы работали с подклассом функций, а именно с классом квадратичных функций.

Определение 1. Булева функция является *квадратичной*, если степень её полинома Жегалкина равна 2. Квадратичная булева функция имеет вид

$$f(x_1, \ldots, x_n) = \left(\bigoplus_{i \neq j} a_{ij} x_i x_j \right) \oplus \left(\bigoplus_{i=1}^{n} b_i x_i \right) \oplus a_0,$$

где $a_0, a_{ij}, b_i \in \mathbb{F}_2$, $\bigoplus_{i \neq j} a_{ij} x_i x_j$ называется *квадратичной частью*; $\bigoplus_{i=1}^{n} b_i x_i$ — *линейной частью*.

Определение 2. Любую квадратичную функцию с помощью аффинных преобразований можно привести к виду $x_1x_2 \oplus x_3x_4 \oplus \ldots \oplus x_{2r-1}x_{2r} \oplus \bigoplus_{i=2r+1}^{n} b_i x_i$. Число r называется *рангом* квадратичной булевой функции. Функции с разным рангом не являются аффинно эквивалентными.

Пример $r = 1$: $f(x_1, x_2, x_3) = x_1x_2 \oplus x_3$; $f(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3$ т. к. $f(x_1, x_2, x_3) = g(x_1, y_1) = x_1y_1$ при замене $y_1 = x_2 \oplus x_3$.

Пример $r = 2$: $f(x_1, x_2, x_3, x_4) = x_1x_2 \oplus x_3x_4$.

По результатам исследования была сформулирована следующая гипотеза.

Гипотеза 1. При любом чётном $n \geq 4$ существует сбалансированная квадратичная функция от n переменных, по которой нельзя построить F конструкцией 1.

Проведение работы

3 переменные

Рассмотрим квадратичные булевые функции при 3 переменных на свойства, описанные ранее. Ранг функций при 3 переменных равен 1.

- x_1x_2, x_1x_3, x_2x_3;
- $x_1x_2 \oplus x_1x_3 = x_1(x_2 \oplus x_3) = x_1y_1$ (замена $y_1 = x_2 \oplus x_3$). Аналогично для $x_1x_2 \oplus x_2x_3$ и $x_1x_3 \oplus x_2x_3$;
- $x_1x_2 \oplus x_2x_3 \oplus x_3x_1 = x_1x_2 \oplus x_2x_3 \oplus x_3x_1 \oplus x_3 \oplus x_3 = x_1x_2 \oplus x_2x_3 \oplus x_3x_1 \oplus x_3^2 \oplus x_3 = x_2(x_1 \oplus x_3) \oplus x_3(x_1 \oplus x_3) \oplus x_3 = (x_2 \oplus x_3)(x_1 \oplus x_3) \oplus x_3 = y_1y_2 \oplus x_3$.

x_3	x_2	x_1		f_3	f_2	f_1
0	0	0	0 →	f_{30}	f_{20}	f_{10}
0	0	1	1 →	f_{31}	f_{21}	f_{11}
0	1	0	2 →	f_{32}	f_{22}	f_{12}
0	1	1	3 →	f_{33}	f_{23}	f_{13}
1	0	0	4 →	f_{34}	f_{24}	f_{14}
1	0	1	5 →	f_{35}	f_{25}	f_{15}
1	1	0	6 →	f_{36}	f_{26}	f_{16}
1	1	1	7 →	f_{37}	f_{27}	f_{17}

Векторная функция строилась по $f(x_1, x_2, x_3) = x_1x_2 \oplus x_3$. Были найдены некоторые функции F с использованием программной среды SageMath. При проведении работы для случая $n = 3$, первая координата f_1 была зафиксирована, а именно, взята $f_1 = f$. Из результата выполнения программы были найдены 48 пар (f_2, f_3), удовлетворяющие заданным требованиям (с точностью до инверсии значений).

4 переменные

При $n = 4$ функции имеют ранг 0 (аффинные) или 1, 2 (квадратичные). При этом только квадратичные функции ранга 1 являются сбалансированными.

Теорема 2. *При $n = 4$ не существует сбалансированных квадратичных функций, по которым можно построить F конструкцией 1.*

Для $n \geq 4$, можно использовать неориентированный граф для описания квадратичной части булевых функций [4]. Если существует слагаемое x_ix_j, тогда вершины i и j соединены ребром. *Графово эквивалентными* называем функции, описанные изоморфными графами.

Для анализа также было использовано следующее понятие. *Производной булевой* функции $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ по направлению $a \in \mathbb{F}_2^n$ называется $f'_a(x) = f(x) \oplus f(x \oplus a)$.

В ходе перебора функции были разделены на классы следующим образом. Для каждой функции было подсчитано d_0 — количество направлений, в которых производная функции тождественна 0, d_1 — тождественна 1, и d — не константа. Тройку (d_0, d_1, d) мы назвали *поведением производной* и обозначили ей соответствующий класс.

Оказалось, что графовое представление функции от 4 переменных однозначно соответствует поведению производной функции, что в свою очередь однозначно соответствует рангу функции. Результаты перебора представлены в таблице 1.

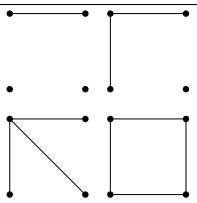
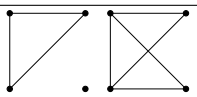
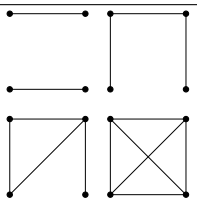
Ранг	Поведение производной	Графовое представление	Количество
1	$(4, 0, 12)$		25
	$(2, 2, 12)$		10
2	$(1, 0, 15)$		28

Таблица 1: Найденные свойства квадратичных булевых функций при $n = 4$

ЛИТЕРАТУРА

[1] Н.Н. Токарева, *Симметричная криптография. Краткий курс.*, учебное пособие. Новосибирск, 2012.

[2] О.А. Логачёв, А.А. Сальников, В.В. Ященко, *Булевы функции в теории кодирования и криптологии*. Москва, 2004.

[3] И.А. Панкратова, *Булевы функции в криптографии*, учебное пособие. Томск, 2014.

[4] Е.П. Корсакова, *Классификация графов квадратичных бент-функций от шести переменных* // Дискретный анализ и исследование операций, 2013, Т.20, N.5, С.45–57.

[5] K.U. Schmidt, *Asymptotically optimal Boolean functions*. J. Comb. Theory, Ser. A, 2019, V.164, P.50–59.

[6] S. Maitra, P. Sarkar, *Modifications of Patterson-Wiedemann functions for cryptographic applications* // IEEE Transactions on Information Theory, 2002, V.48, No.1, P.278–284.

[7] A.M. Youssef, S.E. Tavares, *Affine equivalence in the AES round function* // Discrete Applied Mathematics, 2005, V.148, No.2, P.161–170.

Куратор исследования

Коломеец Н.А., канд. физ.-мат. наук, ст. преп. кафедры теоретической кибернетики НГУ.

Криптографические свойства булевых функций

Н. А. Якунин[1], А. С. Шапоренко[2]
[1] ГАУ Ко ОО школа-интернат лицей-интернат
[2] Новосибирский государственный университет
E-mail: flashastroll300@gmail.com, a.shaporenko@g.nsu.ru

Аннотация

За годы использования булевых функций в системах шифрования были сформулированы математические требования, которые накладываются на булевы функции, для противодействия различным криптографическим атакам. В рамках проекта были реализованы программы для проверки криптографических свойств. Были проверены свойства для малых чисел переменных $n = 3, 4$. Для 8 и 10 переменных были получены уравновешенные булевы функции с высокими параметрами нелинейности равными 112 и 480 соответственно. Выдвинута гипотеза о возможности построения уравновешенных булевых функций от n переменных со значением нелинейности $2^{n-1} - 2^{n/2}$ и другими "хорошими"криптографическими свойствами.

Ключевые слова: *булевы функции, криптографические свойства, уравновешенные функции, высокая нелинейность.*

Функция $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ называется *булевой функцией* от n переменных. *Весом Хэмминга* $wt(x)$ вектора $x \in \mathbb{Z}_2^n$ называется количество ненулевых координат в данном векторе:

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

Булева функция f от n переменных называется *уравновешенной*, если $wt(f) = 2^{n-1}$. *Расстояние Хэмминга* $dist(f, g)$ между двумя булевыми функциями f, g от n переменных вычисляется следующим образом: $d(f, g) = |\{x \in \mathbb{Z}_2^n : f(x) \neq g(x)\}|$.

Каждую булеву функцию f от n переменных можно единственным образом представить в виде *полинома Жегалкина*:

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

где при каждом k индексы $i_1, ..., i_k$ различны и в совокупности пробегают все k-элементные подмножества $\{1, ..., n\}$, а коэффициенты $a_{i_1}, ..., a_{i_k}, a_0$ принимают значение 0 или 1.

Алгебраической степенью (степенью) $\deg(f)$ функции f называется количество переменных в самом длинном слагаемом ее полинома Жегалкина, при котором коэффициент не равен нулю. Функция степени не выше 1 называется *аффинной*. Аффинную функцию от n переменных также можно представить в виде $\ell_{a,b} = \langle x, a\rangle \oplus b$, где $a \in \mathbb{Z}_2^n$ и $b \in \mathbb{Z}_2$, а $\langle x, y\rangle = x_1y_1 \oplus \ldots \oplus x_ny_n$.

Булевы функции f и g от n переменных *аффинно эквивалентны*, если существуют невырожденная квадратная матрица A порядка $n \times n$ и вектор $b, \in \mathbb{Z}_2^n$ такие, что $g(x) = f(Ax \oplus b)$.

Производной булевой функции f от n переменных по направлению $y \in \mathbb{Z}_2^n$ называется функция $D_yf(x) = f(x) \oplus f(x \oplus y)$.

Определение 1. Критерий распространения $PC(k)$ порядка k – для любого ненулевого вектора $y \in \mathbb{Z}_2^n$ веса не более k, где $1 \leqslant k \leqslant n$, функция $D_yf(x) = f(x) \oplus f(x \oplus y)$ уравновешена. При $k = 1$ говорим, что функция f удовлетворяет строгому лавинному критерию.

Определение 2. Булева функция f имеет линейную структуру, если существует ненулевое направление $y \in \mathbb{Z}_2^n$ такое, что $D_y f(x) \equiv const$.

Определение 3. Булева функция f от n переменных называется корреляционно-иммунной порядка r, $1 \leqslant r \leqslant n$, если для любой ее подфункции $g = f_{i_1,\ldots,i_r}^{a_1,\ldots,a_r}$, полученной фиксацией r переменных, выполняется $wt(g) = \frac{wt(f)}{2^r}$. Максимальный порядок корреляционной-иммунности для булевой функции f обозначим cor(f).

Определение 4. *Нелинейностью* N_f булевой функции f от n переменных называется расстояние Хэмминга от данной функции до множества всех аффинных функций, а именно

$$N_f = \min_{a \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2} d(f, \ell_{a,b}),$$

где $a \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2$ и $\ell_{a,b}(x) = \langle a, x \rangle \oplus b$.

Определение 5. Булева функция от четного числа переменных n называется *бент-функцией*, если ее значение нелинейности достигает наибольшего значения $2^{n-1} - 2^{\frac{n}{2}-1}$ [1].

Малое число переменных

Изучив основные криптографические свойства булевых функций, а также методы их определения, мы решили сравнить различные свойства булевых функций для малого количества переменных ($n = 3, 4$) и найти количество функций с "хорошими"свойствами.

$n = 3$

уравновешенность	степень	N_f	cor(f)	$PC(k)$	линейные структуры	число функций
нет	3	1	0	0	нет	128
да	1	0	1	0	есть	6
нет	2	2	1	2	есть	8

$n = 4$

уравновешенность	степень	N_f	cor(f)	$PC(k)$	линейные структуры	число функций
нет	3	4	1	0	нет	192
да	2	4	1	1	есть	24
да	2/3	4	0	1	нет	1056
нет	2	6	0	4	нет	896

Анализируя полученные данные из таблицы, мы можем сделать вывод: булевой функции с *"идеальными"* свойствами не существует и выбор булевых функций сильно зависит от рассматриваемых задач.

Перебирая булевы функции от $n = 2, 3, 4$ переменных, мы обнаружили интересный факт: если функция имеет максимальную степень, то она не имеет линейных структур.

Предложение 1. *Пусть f – булева функция от n переменных и $deg(f) = n$. Тогда f не имеет линейных структур.*

Доказательство. Известно, что если функция имеет линейную структуру, то она аффинно эквивалентна булевой функции с линейной или фиктивной переменной [2]. Но поскольку f имеет максимальную степень, любая функция g аффинно эквивалентная f имеет в полиноме Жегалкина моном $x_1x_2\dots x_n$. □

Уравновешенные функции с высокой нелинейностью

Поскольку вес бент-функций от n переменных равен $2^{n-1} \pm 2^{n/2-1}$, бент-функции не являются уравновешенными. В ходе работы над проектом мы старались получить уравновешенные функции с высоким показателем нелинейности путем модификации бент-функций. Для этого мы использовали конструкцию бент-функций, которые имеют некоторую заданную аффинную функцию $\ell(x)$ своей производной, которая была представлена в [3].

Предложение 2. *[3] Пусть $\ell \neq const$ – аффинная функция от $n \geq 4$ переменных, существенно зависящая от x_2, такая, что $\ell(x) = \ell(x \oplus y)$ для некоторого $y(y_1 = 1)$ и для любого $x \in \mathbb{Z}_2^n$. Булева функция*

$$g(x) = ((D_{\bar{y}}f_0(\bar{x}) \oplus D_{\bar{y}}f_1(\bar{x}) \oplus 1)\ell(x) \oplus D_{\bar{y}}f_1(\bar{x}))x_1 \oplus$$
$$\oplus(f_0(\bar{x}) \oplus f_1(\bar{x}))\ell(x) \oplus f_1(\bar{x}),$$

где

$$\bar{z} = (z_3, \dots, z_n), \text{для } z_k \in \mathbb{Z}_2,$$

$$f_0, f_1 - \text{бент-функции от } n-2 \text{ переменных},$$

является бент-функцией от n переменных.

Идея построения уравновешенных функций с высокой нелинейностью следующая: из двух бент-функций f_0 и f_1, максимально нелинейной оставить f_0, а в качестве f_1 брать произвольную булеву функцию.

Для 8 и 10 переменных было проверено, что если

$$f_1(\bar{x}) = x_3 \oplus x_4 \oplus \dots \oplus x_n \oplus x_3x_4 \oplus x_5x_6 \oplus \dots \oplus x_{n-3}x_{n-2} \oplus c,$$

$$l(x) = x_2 \oplus \dots \oplus x_n \oplus d,$$

где $c, d \in \mathbb{Z}_2$, то для любого направления $\bar{y}$ полученные булевы функции будут уравновешенными с параметрами нелинейности $N_f = 112$ и 480 соответственно. В качестве бент-функций f_0 для получения функций от 8 переменных были рассмотрены все представители классов аффинной эквивалентности бент-функций от 6 переменных, а для 10 переменных - представители классов аффинной эквивалентности бент-функций от 8 переменных степени $\leqslant 3$, а также бент-функция $f_0(\bar{x}) = x_3x_7 \oplus x_4x_8 \oplus x_5x_9 \oplus x_6x_{10} \oplus x_7x_8x_9x_{10}$ степени 4. При этом все функции не имели линейных структур, удовлетворяли лавинному критерию, а степень не превышала $n/2$. Для некоторых других ℓ и f_1, такие значения нелинейности не сохраняются.

Справедливо будет выдвинуть следующую гипотезу.

Гипотеза 2. Пусть $\ell = x_2 \oplus \dots \oplus x_n \oplus d$ – аффинная функция от $n \geq 4$ переменных, где $d \in \mathbb{Z}_2$. Булева функция

$$g(x) = ((D_{\bar{y}}f_0(\bar{x}) \oplus D_{\bar{y}}f_1(\bar{x}) \oplus 1)\ell(x) \oplus D_{\bar{y}}f_1(\bar{x}))x_1 \oplus$$
$$\oplus(f_0(\bar{x}) \oplus f_1(\bar{x}))\ell(x) \oplus f_1(\bar{x}),$$

где

$$\bar{z} = (z_3, \dots, z_n), \text{для } z_k \in \mathbb{Z}_2, \bar{y} \in \mathbb{Z}_2^{n-2},$$

f_0 – бент-функции от $n-2$ переменных,

$$f_1(\bar{x}) = x_3 \oplus x_4 \oplus \ldots \oplus x_n \oplus x_3x_4 \oplus x_5x_6 \oplus \ldots \oplus x_{n-3}x_{n-2} \oplus c, \text{ для } c \in \mathbb{Z}_2,$$

является уравновешенной булевой функцией от n переменных с параметром нелинейности

$$N_f = 2^{n-1} - 2^{n/2},$$

которые удовлетворяют лавинному критерию и не имеют линейных структур.

ЛИТЕРАТУРА

[1] Tokareva, N. Bent Functions, Results and Applications to Cryptography, Acad. Press. Elsevier, 2015.

[2] Панкратова И. А. Булевы функции в криптографии: Учебное пособие, Томск: Томский гос. ун-т, 2014.

[3] Shaporenko, A. Derivatives of bent functions in connection with the bent sum decomposition problem, Des. Codes Cryptogr. 91, 1607 – 1625, 2023.

Кураторы исследования –
старший преподаватель НГУ, Шапоренко Александр Сергеевич.

Построение булевых функций с оптимальной алгебраической иммунностью в сочетании с другими свойствами

М. А. Белов[1], А. Н. Ибрагимова[1], Ю. П. Леонтьева[2], Н. Н. Сероштан[2], Ю. В. Хорева[1], И. С. Хильчук[3]

[1]Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

[2]Южный федеральный университет

[3]Новосибирский государственный университет

E-mail: belov.maksim.alekseevich@mail.ru, a_linalii@mail.ru, izavodnova@sfedu.ru, seroshtan@sfedu.ru, juliakhorevva@yandex.ru, i.khilchuk@g.nsu.ru

Аннотация

В работе рассматриваются алгоритмы перехода от булевых функций малых размерностей с оптимальными криптографическими характеристиками к функциям от большего числа переменных. Анализируется возможность сохранить и повысить показатели криптографических свойств при таких переходах.

Ключевые слова: *булева функция, алгебраическая иммунность, корреляционная иммунность, нелинейность, алгебраическая степень*

Булевы функции являются неотъемлемой частью симметричного шифрования, которое, в свою очередь, играет важную роль в современной информационной безопасности компьютерных систем. Основными факторами стойкости криптосистемы к атакам являются свойства булевых функций, которые показывают возможность противостоять атакам того или иного типа. Например, функции с высокими показателями алгебраической иммунности и нелинейности гарантируют стойкость шифра к алгебраическим и линейным криптоатакам, соответственно. Однако функций, которые имели бы максимальные характеристики по всем свойствам, не существует, отчего возникает необходимость искать компромисс — функции с **достаточно хорошими** характеристиками. Прямой перебор всего множества булевых функций сложно осуществить даже для сравнительно небольшого числа переменных: число различных булевых функций от n равно 2^{2^n}, то есть уже для случая $n = 5$ имеется 2^{32} различных функций, из-за чего функции чаще строят аналитически, итеративно или ищут с использованием методов машинного обучения.

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

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

где при каждом k все индексы $i_1, \ldots, i_k$ различны и параметры $a_0, a_{i_1}, \ldots, a_{i_k}$ принимают значения 0 или 1.

Степенью булевой функции называется число переменных в самом длинном ненулевом слагаемом АНФ. Для стойкости шифра необходимо, чтобы функции обладали высокой алгебраической степенью. Функции, степень которых не превосходит 1, называются *аффинными*.

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

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

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

Нелинейность $nl(f)$ булевой функции f от n переменных определяется как расстояние Хэмминга от нее до множества всех аффинных функций [1]. Высокая нелинейность функции обеспечивает стойкость шифра к линейному криптоанализу.

Алгебраической иммунностью $AI(f)$ 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 $(f(x) + 1)g(x) = 0$ [1]. Высокая алгебраическая иммунность обеспечивает стойкость шифра к алгебраическому криптоанализу.

Булева функция $f(x)$ от n переменных называется *сбалансированной*, если принимает каждое из значений 0 и 1 ровно 2^{n-1} раз [1]. Сбалансированность позволяет ослабить статистическую зависимость между входом и выходом функции.

В работе мы исследовали итеративное построение уравновешенных булевых функций, обладающих оптимальной алгебраической иммунностью и высокой нелинейностью. Впоследствии при повышении размерности проверили также корреляционную иммунность. Целью нашей работы является поиск алгоритма перехода из меньшего количества переменных булевой функции к большим с улучшением криптографических характеристик.

Для перехода от функций от n переменных к функциям от $n + 1$ переменных использовали два метода: преобразование входной последовательности и преобразование вектора значений функции. Производили последовательные переходы до $n = 5$, на каждом этапе выбирая функции с лучшими характеристиками и способы построения, давшие больше таких функций.

При преобразовании входной последовательности из функции $f(x_1, \ldots, x_n) = f(x^1), \ldots, f(x^{2^n})$ получаем функцию $f(x_1, \ldots, x_n, x_{n+1}) = f(x^1), \ldots, f(x^{2^n}), f(\pi(x^1)), \ldots, f(\pi(x^{2^n}))$, где π — перестановка координат вектора x, $x^i \in \mathbb{Z}_2^n, i = 1, \ldots, 2^n$. Из множества функций от 3 переменных выбрали 56 уравновешенных функций с оптимальной алгебраической иммунностью (их нелинейность равна 2). В случае, когда перестановка π являлась транспозицией двух координат вектора x, все полученные с помощью таких транспозиций функции обладали оптимальной алгебраической иммунностью и нелинейностью не ниже 4 (при использовании каждой из транспозиций 4 из полученных функций обладали также корреляционной иммунностью первого порядка). Если перестановка не оставляет на месте ни одной координаты, то корреляционно-иммунных среди полученных функций нет, все функции обладают оптимальной алгебраической иммунностью и нелинейностью не ниже 4. От полученных 280 функций осуществили переход к функциям от 5 переменных:

- Транспозиция двух координат вектора x приводит к следующему распределению характеристик: 4 функции обладают максимальной алгебраической иммунностью и $nl(f) = 10$. 12 функций корреляционно-иммунны порядка 1, $nl(f) = 8$. Пример функции, полученной данным способом ($AI(f) = 3, nl(f) = 10$):

$$f(x) = x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_2x_3x_4 \oplus x_2x_4x_5 \oplus x_1x_2 \oplus x_1x_3 \oplus x_4x_5 \oplus x_3 \oplus x_4 \oplus x_5$$

- Перестановки, меняющие местами три координаты вектора x, дают от 96 до 106 функций с оптимальной алгебраической иммунностью и нелинейностью, равной 10. Каждая из таких перестановок позволяет получить 12 корреляционно-иммунных порядка 1 функций, их нелинейность равна 8. Пример функции, полученной данным способом ($AI(f) = 3, nl(f) = 10$):

$$f(x) = x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_1x_2x_3 \oplus x_1x_2x_5 \oplus x_1x_3x_4 \oplus x_1x_4x_5 \oplus x_2x_3x_4 \oplus$$
$$\oplus x_2x_4x_5 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_5 \oplus x_4x_5 \oplus x_3$$

- Перестановки, не оставляющие на месте ни одну координату вектора x, прозволяют получить от 106 до 112 функций с оптимальной алгебраической иммунностью и нелинейностью, равной 10. Каждая из таких перестановок позволяет получить 12 корреляционно-иммунных порядка 1 функций, их нелинейность равна 8. Пример функции, полученной данным способом ($AI(f) = 3, nl(f) = 10$):

$$f(x) = x_1x_2x_3x_4 \oplus x_1x_2x_4x_5 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_3x_5 \oplus x_1x_4x_5 \oplus x_2x_3x_4 \oplus$$
$$\oplus x_2x_3x_5 \oplus x_1x_2 \oplus x_1x_3 \oplus x_3x_5 \oplus x_4x_5 \oplus x_3$$

Для того, чтобы в векторном представлении функции совершить переход от функции n переменых к функции $n + 1$ переменных, необходимо увеличить длину исходного вектора в 2 раза. Если вектор начальной функции имеет вид $f(x_1, \ldots, x_n) = y_1, \ldots, y_{2^n}$, то при переходе получим $f(x_1, \ldots, x_n, x_{n+1}) = y_1, \ldots, y_{2^n}, \sigma(y_1, \ldots, y_{2^n})$, где σ — одно из преобразований: тождественное, инверсия, зеркальное отображение, циклический сдвиг. Мы рассматривали все последовательные переходы для $n = 2, \ldots, 5$. Из множества функций от 2 переменных мы выбрали уравновешенные и применили вышеперечисленные преобразования. Тождественное преобразование, инверсия, зеркальное отображение, циклический сдвиг на 2 при переходе от 2 к 3 переменным дали функции с $AI(f) = 1$, что не является оптимальным показателем для функции 3 переменных. С помощью циклического двига на 1 позицию влево и вправо из исходных 6 функций получили 12 функций 3 переменных, 8 из которых обладали оптимальной алгебраической иммунностью и $nl(f) = 2$. В дальнейшем мы использовали только циклический сдвиг. Далее из 8 функций с предыдущего шага мы получили 16 функций 4 переменных, каждая из которых обладала оптимальной алгебраической иммунностью, нелинейностью, равной 4, а так же половина из них были корреляционно-иммунными порядка 1. Пример такой функции ($AI(f) = 2, nl(f) = 4, CI(f) = 1$):

$$f(x) = x_1x_2 \oplus x_1x_4 \oplus x_2x_4 \oplus x_3 \oplus 1$$

В результате перехода к функциям от 5 переменных мы получили 32 функции, 28 из которых обладали $AI(f) = 3$, что является оптимальным показателем для функции 5 переменных, и $nl(f) = 10$. Пример такой функции $AI(f) = 3, nl(f) = 10$:

$$f(x) = x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_1x_4x_5 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_5 \oplus x_2x_3 \oplus x_2x_5 \oplus x_3x_5 \oplus x_2 \oplus x_3 \oplus x_4$$

В работе были проанализированы два способа перехода к булевым функциям от большего числа переменных до $n = 5$. Способ, опирающийся на циклический сдвиг вектора значений исходной функции, охватывает меньшее количество функций, но чаще позволяет получить функции с оптимальными характеристиками. Перестановки входной последовательности функции порождают большее количество функций, однако реже приводят к функциям с оптимальными показателями криптографических свойств. Интересно, что оба способа построения позволили нам получить корреляционно-иммунные булевы функции с максимальной алгебраической иммунностью от 4 переменных, количество которых равно 392 из 2^{16} [5].

ЛИТЕРАТУРА

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

[2] Cusick T.W., Stănică P.: Cryptographic Boolean Functions, second edition. Academic Press, San Diego (2017).

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

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

[5] Зюбина Д. А., Хильчук И. С., Корнилов А. П., Malanda S.N. О корреляционно-иммунных функциях с максимальной алгебраической иммунностью // Летняя школа-конференция "Криптография и информационная безопасность" 2021, сборник тезисов

Кураторы исследования –
аспирантка ММФ НГУ, м.н.с. Международного Математического центра в Академгородке, Хильчук Ирина Сергеевна

АЛГЕБРАИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

Анализ сдвиговых преобразований в квазигруппах

А. А. Голяшов[1], В. О. Гурьянов[2], А. А. Мухортова[2], О. Ю. Немова[2]
[1]Балтийский Федеральный Университет им. И. Канта
[2]Национальный Исследовательский Ядерный Университет "МИФИ"
E-mail: anton.golyashov@gmail.com, majorlenox@mail.ru, alyonka.mukhortova@gmail.com, olyanemova36@gmail.com

Аннотация

Для сохранения формата в зашифрованом тексте при условии множества открытых текстов относительно малого размера могут использоваться подстановки на основе квазигрупповых сдвигов. В данной работе проводится эксперимент по исследованию характеристик подстановок, сгенерированных при помощи квазигрупповой операции на соответствие характеристикам случайных подстановок. В ходе исследования проведены статистические тесты на основе критерия Стьюдента и критерия согласия хи-квадрат. В числе выбранных метрик присутствуют кратчайшая длина цикла, количество неподвижных точек, количество инверсий, четность подстановки, количество циклов, порядок подстановки и количество рекордов. В результате данной работы были получены способы генерации подстановок и их соответствие случайным подстановкам. Лучший результат показал способ генерации квазигрупповой операции на основе случайной биекции.

Ключевые слова: *Квазигруппа, статистический тест, подстановки, сдвиговые преобразования, правильные семейства булевых функций.*

Введение и постановка задачи

Пусть задано множество сообщений $\mathcal{M}$ произвольного вида. Необходимо построить алгоритм шифрования, который бы сохранял формат сообщения, то есть результат зашифрования ct некоторого сообщения $m \in \mathcal{M}$ имел бы тот же формат, что и исходное сообщение: $ct \in \mathcal{M}$.

Блочные шифры не могут обеспечить данное свойство для любого открытого текста, поскольку что множество $\mathcal{M}$ может быть сильно меньше, чем множество блоков, которые блочный шифр принимает на вход.

Для решения данной задачи могут быть использованы псевдослучайные подстановки на основе квазигрупп.

Целью данной работы является исследование параметров сдвиговых преобразований в квазигруппах на соответствие параметрам случайной подстановки.

Статистистические тесты для подстановок на малых множествах

Существующие статистические батареи тестов, например, NIST SP800-22, DIEHARD, TESTU01, предназначены для тестирования нулевой гипотезы, заключающейся в случайном независимом равновероятном распределении битов. Для поставленной задачи эти тесты не подходят, так как рассматриваются подстановки на малых множествах. При таких условиях отличимость набора значений подстановки от случайной двоичной строки соответствующей длины станет заметна

на любом подвекторе значений подстановки, длина которого равна корню из размера множества (согласно парадоксу дней рождения). Также применение существующих батарей тестов невозможно из-за ограниченных длин последовательностей значений подстановки.

Из-за невозможности использования стандартных методов исследование проводилось при помощи различных статистик подстановок и их оценки с помощью статистического критерия Стьюдента и критерия согласия хи-квадрат.

Используемые статистики и их распределения

Для случайных подстановок известны предельные распределения некоторых их статистик. Ниже приведены (предельные либо точные) распределения статистик, которые использовались в данной работе:

1. вероятность наличия ровно k неподвижных точек в подстановке

$$P[k] = \frac{1}{k!} \cdot \frac{!(n-k)}{(n-k)!},$$

   где за $!(n-k)$ обозначено количество беспорядков (подстановок без неподвижных точек) на множестве размера $n-k$,

2. вероятность наличия ровно k циклов в подстановке

$$P[k] = \frac{1}{n!} \cdot \left[\begin{array}{c} n \\ k \end{array} \right],$$

   где за $\left[\begin{array}{c} n \\ k \end{array} \right]$ обозначено число Стирлинга первого рода с параметрами (n, k),

3. вероятность того, что длина кратчайшего цикла подстановки больше k

$$P[> k] \xrightarrow[n\to\infty]{} e^{-(1+\frac{1}{2}+\ldots+\frac{1}{k})}, \tag{3}$$

4. рекорд - наибольшее из всех предыдущих значений. Вероятность наличия k рекордов

$$P[k] = \frac{1}{n!} \left[\begin{array}{c} n \\ k \end{array} \right],$$

5. математическое ожидание порядка подстановки [2],

$$\log \mu_n \xrightarrow[n\to\infty]{} c\sqrt{\frac{n}{\log n}}, \text{ где}$$

$$c = 2\sqrt{(2\int_0^\infty \log\log(\frac{e}{1-e^{-t}})dt)},$$

6. числа i и j образуют в перестановке инверсию, если $i > j$, но i расположено раньше j. Количество перестановок с k инверсиями - коэффициент при x^k многочлена

$$P_{n-1}(x) = (1+x)(1+x+x^2)\ldots(1+x+\ldots+x^{n-1}),$$

7. вероятность того, что перестановка чётная

$$P = \frac{1}{2}.$$

Распределения статистик (1) - (4) приведены в книге [1]. Для длины кратчайшего цикла используется формула при асимптотическом приближении $n \to \infty$ для повышения скорости вычисления при больших n. Рассматривалось сравнение форумулы (1) с экспоненциальной производящей функцией для вероятностей $P[> k]$, откуда извлекали n-ый коэффициент,

$$P_{=k}(x) = \frac{e^{-(x+\ldots+\frac{x^{k-1}}{k-1})} - e^{-(x+\ldots+\frac{x^k}{k})}}{1-x}. \tag{4}$$

На рис. 1 приведены результаты проверки возможности замены точных формул, полученных в соответствии с формулой (2), на асимптотическое приближение (1) при малых значениях n. По результатам воспользовались в проведении экспериментов формулой (1) вместо вычислительно сложной (2).

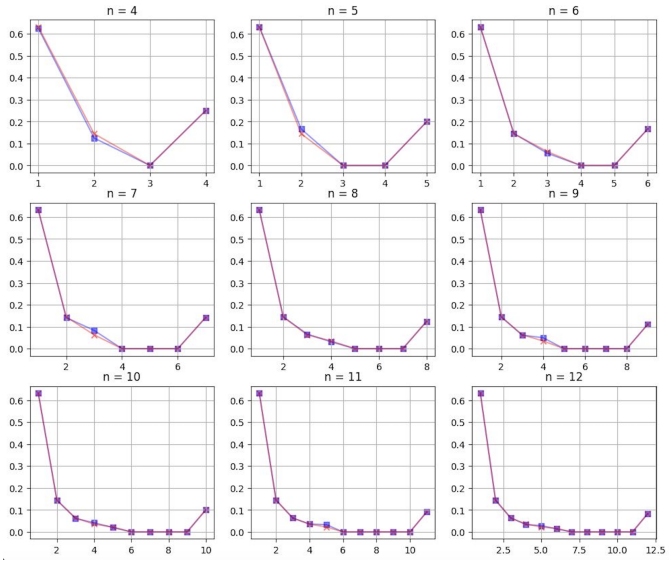


Рис. 1: Сравнение распределений длин кратчайшего цикла подстановки при $n = \overline{4, 12}$

Статистическое тестирование гипотез

Для оценки статистических критериев использовался критерий согласия χ-квадрат, который является универсальным и позволяет проверить гипотезы о соответствии эмпирического распределения предполагаемому теоретическому распределению при большом объеме выборки.

Оцениваемые признаки подстановки с помощью χ-квадрат:

- Количество неподвижных точек (FP);
- Количество циклов (NoC);
- Кратчайшая длина цикла (SC);
- Рекорды (Rec);

Критерий χ-квадрат применим, если известно некоторое теоретическое распределение величины. Для статистик, где известно только одно значение (обычно среднее), был использован t-критерий Стьюдента. Этот критерий выбран исходя из большого размера выборки и близкого к нормальному распределению велилчин в ней.

Оцениваемые признаки подстановки с помощью t-критерия Стьюдента:

- Чётность (Evn);
- Порядок (Ord);
- Количество инверсий (Inv);

Исследование избыточности тестов

Для исследования избыточности тестов построена корреляционная матрица признаков (рис.2).

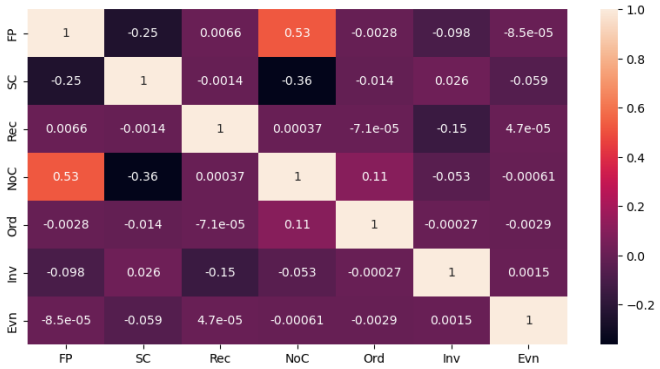


Рис. 2: Корреляционная матрица статистик

Из построенной корреляционной матрицы очевидно, что подобранные признаки имеют слабую корреляцию, поэтому избыточные тесты отсутствуют. Корреляция количества неподвижных точек и количества циклов заметно выше, чем у остальных величин, но все еще незначительна.

Квазигруппы

Общие определения

Определение 1. Квазигруппа — множество Q с заданной на нём бинарной операцией

$$\circ\colon Q \times Q \to Q,$$

со следующим свойством: для любых $a, b \in Q$ существуют единственные $x, y \in Q$, такие что:

$$a \circ x = b, \qquad y \circ a = b.$$

Другими словами, операции **левого** L_a и **правого** R_a умножения (сдвигов)

$$L_a\colon Q \to Q,\ L_a(x) = a \circ x,$$

$$R_a\colon Q \to Q,\ R_a(y) = y \circ a,$$

являются биекциями на Q.

Определение 2. Квазигруппы $(Q, \circ)$ и $(Q', *)$ называются **изотопными**, если существуют взаимно-однозначные отображения $\pi, \sigma, \tau \colon Q \to Q'$, такие что

$$x \circ y = \tau^{-1}(\pi(x) * \sigma(y)).$$

Определение 3. Семейство булевых функций

$$F = (f_1(x_1, \ldots, x_n), f_n(x_1, \ldots, x_n))$$

называется правильным, если для любых двух различных двоичных наборов $x = (x_1, \ldots, x_n)$ и $y = (y_1, \ldots, y_n)$, $x \neq y$, выполняется следующее условие:

$$\exists i : x_i \neq y_i, f_i(x) = f_i(y)$$

Замечание 1. Заметим, что из определения правильного семейства следует, что любая функция f_i из правильного семейства F не может существенно зависеть от одноименной переменной x_i: если бы f_i зависела существенно от x_i, то по определению существенной зависимости нашлись бы два набора, различающихся только в i-й компоненте, на которых f_i принимает различные значения:

$$\exists x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n :$$

$$f_i(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f_i(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n),$$

что противоречит условию правильности.

Способы генерации квазигрупп

Зададим квазигруппу на множестве $Q = \{0, 1\}^n$ с помощью правильного семейства булевых функций.

Рассмотрим следующее задание квазигрупповой операции:

$$x \circ y = z \leftrightarrow z_i = x_i \oplus y_i \oplus f_i(\pi_1(x_1, y_1), ..., \pi_n(x_n, y_n)).$$

.

В работе были исследованы квазигрупповые операции на основе двух различных семейств функций.

1. $F = (f_1(x_1, ..., x_n), ..., f_n(x_1, ..., x_n))$, где f_i — любые функции, такие что $f_i = f(x_1, ..., x_{i-1})$, которые зависят только от $x_1, ..., x_{i-1}$.
2. $F = (f_1(x_1, ..., x_n), ..., f_n(x_1, ..., x_n))$, где f_i — функции вида

$$f_i = x_1 \oplus ... \oplus x_{i-1} \oplus \bigoplus_{i<j, i, j \neq k}^{n} x_i x_j.$$

Приведем еще один способ задать квазигруппу. Пусть $(G, \cdot)$ — некоторая группа, а α, β, γ — биекции на множестве G. Тогда можно рассмотреть операцию $\circ$ на G:

$$x \circ y = \gamma^{-1}(\alpha(x) \cdot \beta(y)).$$

Множество G с таким образом введенной операцией $\circ$ является квазигруппой, изотопной G.

Результаты исследования

Описание эксперимента

В ходе эксперимента проведена проверка гипотезы о равнораспределенности "структурированных"подстановок, полученных как сдвиг на случайный элемент слева, в множестве всех возможных подстановок.

В эксперименте осуществлены

- генерация квазигруппы размера 2^{10},
- выбор случайного элемента квазигруппы,
- порождение перестановки, являющейся левым сдвигом на выбраннный элемент.

В рамках работы проведено 4 эксперимента с различным способом генерации квазигрупп.

Групповые и квазигрупповые операции реализованы как функции от двух переменных:
$x * y = *(x, y)$

1. Квазигрупповая операция порождается с помощью функции высшего порядка:

$$F(\alpha, \beta, \gamma, \cdot) = \circ : \ \circ(x, y) = \gamma(\cdot(\alpha(x), \beta(y))) = \gamma(\alpha(x) \cdot \beta(y)))$$

   В качестве $\cdot$ использовалось побитовое исключающее или (xor), α, β, γ порождались с помощью функции numpy.random.permutation, но использовались неоднократно.

2. Квазигрупповая операция порождается с помощью функции высшего порядка, случайно выбирающей булевы функции π_i и возвращающей функцию $\circ(x, y)$:

$$\circ(x, y) = \circ(x_1, x_2, ...x_n, y_1, y_2, ...y_n) = f(\pi_1(x_1, y_1), ..., \pi_n(x_n, y_n)),$$

   где $f(a_1, a_2, ...a_n) = (f_1(a_1, a_2, ...a_n), f_2(a_1, a_2, ...a_n), ...f_n(a_1, a_2, ...a_n))$,
   f_i принадлежит семейству правильных функций

$$f_k = x_1 \oplus ... \oplus x_{i-1} \oplus \bigoplus_{i<j, i,j \neq k}^{n} x_i x_j$$

3. Квазигрупповая операция порождается с помощью функции высшего порядка:

$$F(\alpha, \beta, \gamma, \cdot) = \circ : \ \circ(x, y) = \gamma(\cdot(\alpha(x), \beta(y))) = \gamma(\alpha(x) \cdot \beta(y)))$$

   В качестве $\cdot$ использовалась операция сложения по модулю 2^{10}. α, β, γ порождались с помощью функции numpy.random.permutation, но использовались неоднократно

4. Квазигрупповая операция порождающая с помощью функции высшего порядка, случайно выбирающей булевы функции π_i и f_i и возвращающей $\circ$:

$$\circ(x, y) = \circ(x_1, x_2, ...x_n, y_1, y_2, ...y_n) = f(\pi_1(x_1, y_1), ..., \pi_n(x_n, y_n))$$

$$f(a_1, a_2, ...a_n) = (f_1(), f_2(a_1), ...f_n(a_1, a_2, ...a_{n-1})).$$

Результаты экспериментов

В результате 4-х экспериментов выявлено 2 способа, дающие параметры подстановок наиболее приближенные к случайным.

Лучшие результаты получены в 1-м и 3-м экспериментах. При генерации заданным способом все параметры оказались приближены к случайным.

По результатам 2-го и 4-го эксперимента параметре перестановок оказали далеки от случайных. В 4-м эксперименте наиболее приближенным к случайным перестановка оказался прааметр длины кратчайшего цикла. Такие способы генерации не дают ожидаемых результатов, их использование не рекомендуется для дальнейшего исследования и использования.

Выводы и направления дальнейших исследований

Проведенные эксперименты показывают, что не любой способ генерации перестановки с помощью квазигрупп позволяет получить подстановки, параметры которых близки к случайным.

По итогам двух экспериментов по всем метрикам наблюдается заметное расхождение с параметрами случайной перестановки. Такие результаты могут возникать из-за недостаточного рассеяния за счет сдвига на 1 элемент влево. Для лучшего результата возможно увеличение количества сдвигов и комбинация левых и правых сдвигов.

Однако также в 2-х экспериментах получены результаты, которые максимально приближены к случайным подстановкам, что дает основания для дальнейших исследований подстановок, полученных на основе квазигрупповых сдвигов.

В дальшейших исследованиях планируется:

- реализовать более сложные методы генерации квазигрупп;
- рассмотреть другие возможные статистики случайных подстановок и соответствующие им статистические тесты;
- реализовать процедуры генерации квазигрупп и проверки статистических гипотез на более высокопроизводительных языках программирования;
- обобщить результат на множества большего размера.

ЛИТЕРАТУРА

[1] R. Sedgewick, P. Flajolet An introduction to the analysis of algorithms. – Addison-Wesley Longman Publishing Co., Inc., 1996.

[2] W. M. Y. Goh, E. Schmutz The expected order of a random permutation //Bulletin of the London Mathematical Society. – 1991. – Т. 23. – №. 1. – С. 34-42.

Куратор исследования

Старший специалист-исследователь Лаборатории Криптографии АО «НПК «Криптонит», Царегородцев Кирилл Денисович.

КРИПТОАНАЛИЗ

Анализ утечек по энергопотреблению для различных реализаций симметричных блочных алгоритмов шифрования

Г. Д. Малютин[1], А. Д. Рудзянский [2], Н. А. Глущенко[1]
[1]Южный Федеральный Университет
[2]Национальный Исследовательский Универсистет "Высшая Школа Экономики"
E-mail: malyutin@sfedu.ru, adrudzyanskiy@edu.hse.ru, glushchenko@sfedu.ru

Аннотация

Данная работа посвящена исследованию методов выявления побочных каналов утечки на примере утечек по энергопотреблению для симметричных блочных алгоритмов шифрования. В работе проведено моделирование трасс энергопотребления устройства при выполнении шифров ГОСТ Р 34.12.2015 (n=64 "Магма"и n=128 "Кузнечик") на базе эмулятора ELMO и выявлены инструкции, по статистическим тестам потенциально содержащие учетки по энергопотреблению. Также описан механизм последующего применения к сформированным трассам энергопотребления метода дифференциального анализа на основе средства DPA-simulator.

Ключевые слова: *побочный канал утечки информации, атаки по побочным каналам, симметричные блочные шифры, эмуляция работы криптографических средств защиты информации, средство поиска статистической утечки по энергопотреблению ELMO*

Применение эмуляторов для моделирования утечек по энергопотреблению криптографических средств

Использование эмуляторов энергопотребления криптографических устройств в сочетании с выполнением набора статистических тестов позволяет исследовать реализации криптографических алгоритмов с возможностью выявления побочных каналов утечки. В качестве примеров программных средств для моделирования утечек по энергопотребелению можно упомянуть такие средства, как Riscure's inspector toolbox, Riscasim, Virtualyzr, Profiled model based power simulator, Silk, Ascold, Savrasc и ELMO. Последовательность шагов моделирования энергопотребления описана в работе [1] и структурно представлена на рисунке 1.

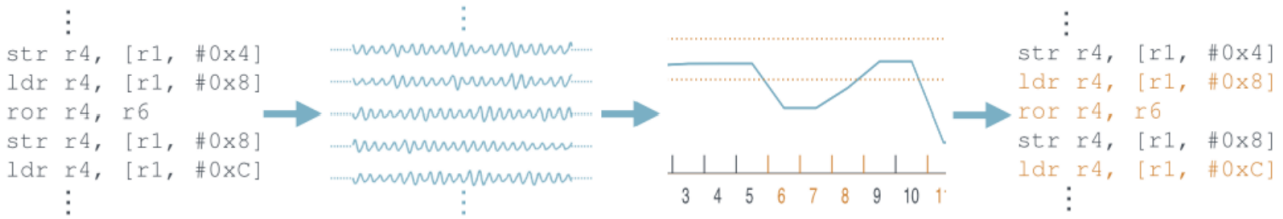


Рис. 1: Общая схема моделирования энергопотребления устройств для обнаружения утечек по побочным каналам.

Для проведения моделирования статистических утечек в рамках исследовательского проекта выбрано средство ELMO [2], которое предназначено для эмулирования работы процессоров семейства ARM Cortex M0 в ходе выполнения симметричных блочных шифров (исходный код средства ELMO доступен на GitHub). Важное отличие эмулятора ELMO от аналогов заключается в том, что оно не использует фиксированные предположения о модели энергопотребления процессора.

ELMO используем в качестве механизма эмулирования работы процессора Thumbulator - эмулятор набора инструкций ARM Cortex M0 с открытым исходным кодом [3]. Thumbulator, как и другие процессоры средней сложности, транслирует ассемблерный код в машинные инструкции. Таким образом, путем ввода произвольных последовательностей ассемблерных инструкций можно достаточно точно воспроизвести поток данных ядра микропроцессора. Thumbulator является эмулятором уровня инструкций и не зависит от периферийных устройств, включая подсистему памяти.

Однако утечка из подсистемы памяти имеет значение в контексте реализации криптографии с контрмерами против утечек через побочные каналы. Поэтому была добавлена упрощенная модель, представляющая реализацию архитектуры шины, соединяющей память с процессором. Разработчики предполагают два 32-разрядных шин - одна для чтения и другая для записи. При этом значение шины изменяется только при выполнении операции чтения/записи.

Генерирование трасс энергопотребления представляет собой процесс соотнесения потока данных на уровне инструкций, генерируемого эмулятором, с прогнозами энергопотребления для каждой инструкции модели. Главной целью ELMO является улучшение стандартных моделей, которые использовались во многих предыдущих симуляторах, путем введения специальных моделей, учитывающих потенциально сложные зависимости данных, включая конвейеризацию инструкций или взаимодействие между соседними проводами в схеме. Модели, интегрированные в ELMO, созданы по принципу „серого ящика“, то есть они не опираются на подробные сведения о конкретной реализации архитектуры, а используют общедоступные базовые возможности. ELMO предоставляет модели для 21 инструкции Thumb, которые широко используются в реализациях симметричной криптографии на базе бюджетных устройств ARM Cortex M0. Все инструкции моделируются независимо и разделены на 5 отдельных групп:

1. Инструкции ALU (`adds`, `adds #imm`, `ands`, `eors`, `movs`, `movs #imm`, `orrs`, `subs`, `subs #imm`, `cmp`, `cmp #imm`);
2. Инструкции сдвига (`lsls`, `lsrs`, `rors`);
3. Инструкции загрузки (`ldr`, `ldrb`, `ldrh`);
4. Инструкции сохранения (`str`, `strb`, `strh`);
5. Инструкция умножения (`muls`).

Построение модели энергопотребления выполняется следующим образом:

1. Измеряется энергопотребление устройства при выполнении различных последовательностей ассемблерного кода на случайных входных данных.
2. Тактовый цикл, соответствующий целевой инструкции, определяется путем анализа и сжимается до одного зависимого переменного значения (например, выбирается пик; другой вариант может включать суммирование всех точек цикла).
3. Определяются возможные переменные, включая входные биты, переходы между последовательными входами, предыдущие и последующие инструкции, а также различные взаимодействия между ними.
4. Путем итерационного построения линейных регрессионных моделей для каждой из 5 групп представительных инструкций добавляются группы переменных, при этом каждый раз удаляются или сохраняются переменные в соответствии с результатами теста на значимость.

5. Для платы NXP модели были расширены для включения подсистемы памяти через расстояние Хэмминга между текущим и предыдущим состояниями на каждой шине, что отслеживается в расширенном потоке данных.

Описание исследуемых симметричных блочных шифров и результаты моделирования утечек

Встроенные в ELMO примеры выявления утечки по энергопотреблению для реализаций криптографического алгоритма Advanced Encryption Standard

Эмулятор ELMO включает в себя несколько базовых примеров поиска утечек для различных реализаций симметричного криптографического алгоритма шифрования Advanced Encryption Standard (AES) [9], являющегося международным стандартом симметричного блочного шифрования.

В рамках проекта, прежде чем реализовывать и исследовать на базе эмулятора отечественные криптографические алгоритмы, были проанализированы тестовые реализации выполнения шифра AES в среде эмулятора ELMO. Эмулятор ELMO содержит в себе несколько реализаций шифра AES. В данном проекте была использована реализация MbedAES из проекта MbedTLS [6], который является легковесной реализацией протокола Transport Layer Security (TLS) для встраиваемых систем.

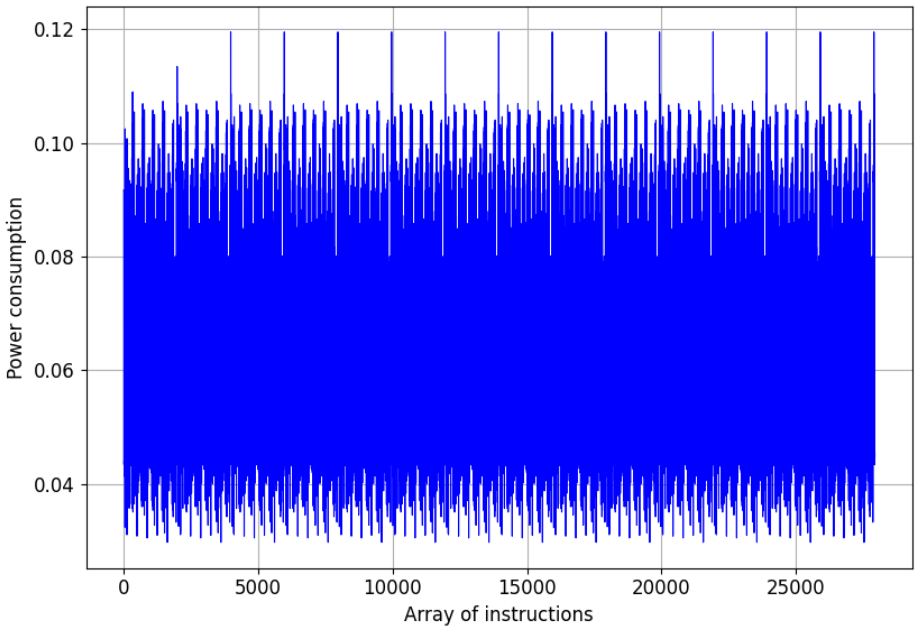


Рис. 2: Сгенерированная в среде ELMO трасса энергопотребления при выполнении алгоритма MbedAES.

На рисунке 2 представлена одна из 200 сформированных трасс энергопотребления шифра MbedAES в среде ELMO. Из графического представления смоделированной трассы энергопотребления видно, что выполнение групп инструкций одного раундового преобразования составляет 2471 инструкцию (число инструкций между соответствующими 14-тью пиками графика трассы на рисунке 2).

Выявление утечки по энергопотреблению для криптографического алгоритма Магма

Целью анализа возможности реализации атак по побочным каналам является выявление наличия утечки информации путем исследования статистической зависимости энергопотребления криптографического устройства от значений обрабатываемой защищаемой информации, таких как входные векторы и секретный ключ. Если все входные векторы приводят к одному и тому же энергопотреблению (среднему значению, полученному из нескольких измерений при фиксированных параметрах), то утечка информации через побочный канал энергопотребления не будет обнаружена. Однако, если графики энергопотребления устройства различаются при разных входных векторах, требуется разработать модель энергопотребления устройства для дальнейшего анализа.

В рамках поставленной задачи, был проведен анализ на выявление утечки энергопотребления алгоритма Магма [4] для ядра ARM Cortex M0. Были получены оценки общего количества инструкций и количества инструкций, содержащий статистические утечки по энергопотреблению, для различного числа раундов (Таблица 2). Было выполнено исследование, направленное на изучение корреляции между количеством раундов алгоритма Магма и изменением количества инструкций, содержащих статистические утечки. Результаты указывают на отсутствие прямой зависимости между этими параметрами.

Таблица 2: Результаты моделирования утечек по побочным каналам для различного количества раундов шифра Магма.

Кол-во раундов	Общее кол-во инструкций	Инструкции с утечками по энергопотреблению
1	701	194
3	2293	697
5	3875	1032
7	5485	1596
9	7077	1888
11	8669	2181
13	10261	2181
15	11853	2943
17	13445	2733
19	15037	3464
21	16629	2891
23	18221	4383
25	19813	3361
27	21405	4454
29	22997	3744
31	24593	5973
32	25389	5309

На рисунке 3 представлена одна трасса энергопотребления полнораундового шифра Магма, полученная в результате применения эмулятора ELMO для обнаружения утечки. Как видно из графического представления смоделированной трассы энергопотребления, сначала выполняются служебные инструкции инициализации параметров шифрования, после чего запускаютстя группы инструкций раундового преобразования (соответствующие 32 пика после $1.1 \cdot 10^4$ инструкции).

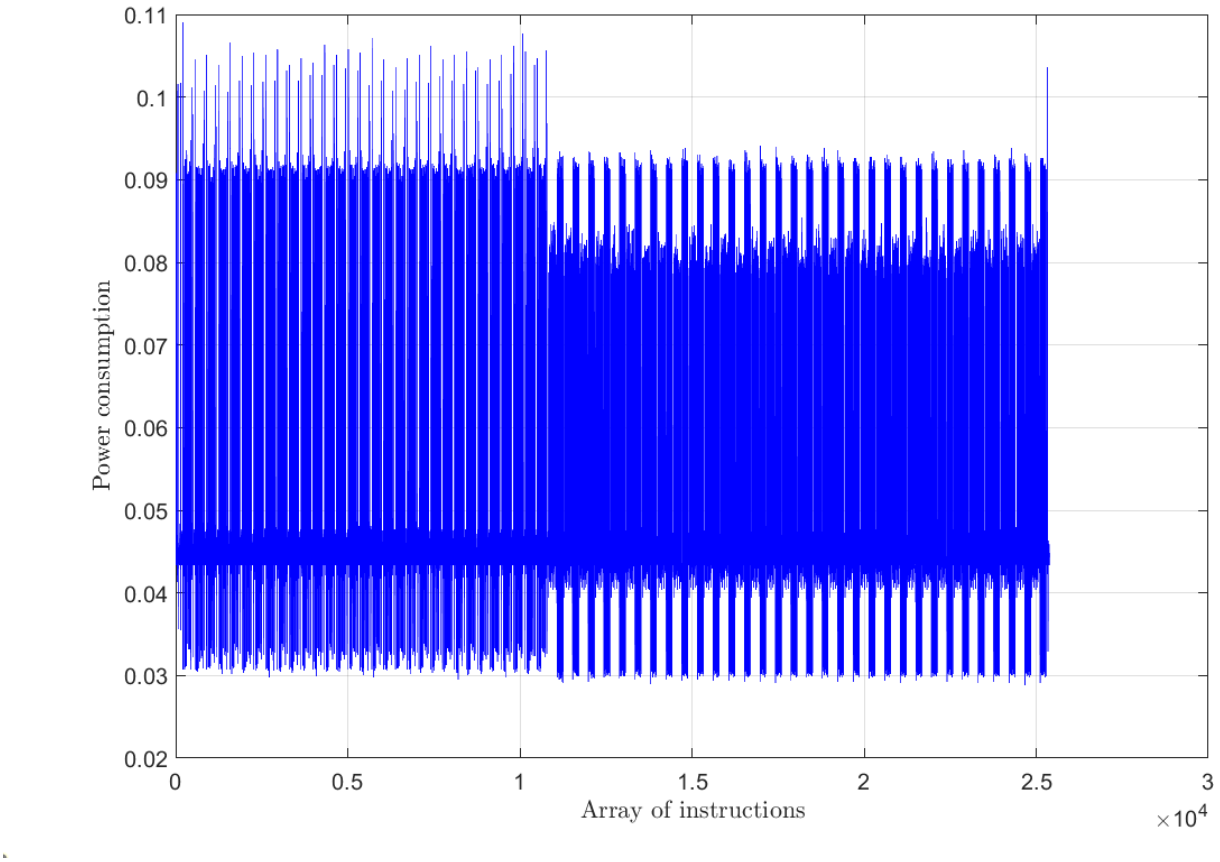


Рис. 3: Сгенерированная в среде ELMO трасса энергопотребления при выполнении полнораундового алгоритма Магма.

На рисунке 4 показан результат выполнения статистического поиска утечки на основе t-теста в ELMO. Значение параметра t вычисляется по формуле 5:

$$t = \frac{mean_{fix} - mean_{rand}}{\sqrt{\frac{var_{fix}}{N_{fix}} + \frac{var_{rand}}{N_{rand}}}} \tag{5}$$

Пороговым значением, по которому делается вывод о наличии утечки, зафиксировано значение $t = |4.5|$, в соответствии с рекомендациями стандарта CSA ISO/IEC 17825-2018 "Information technology - Security techniques - Testing methods for the mitigation of non-invasive attack classes against cryptographic modules"[5] по использованию Test Vector Leakage Assessment (TVLA) [7], [8] .

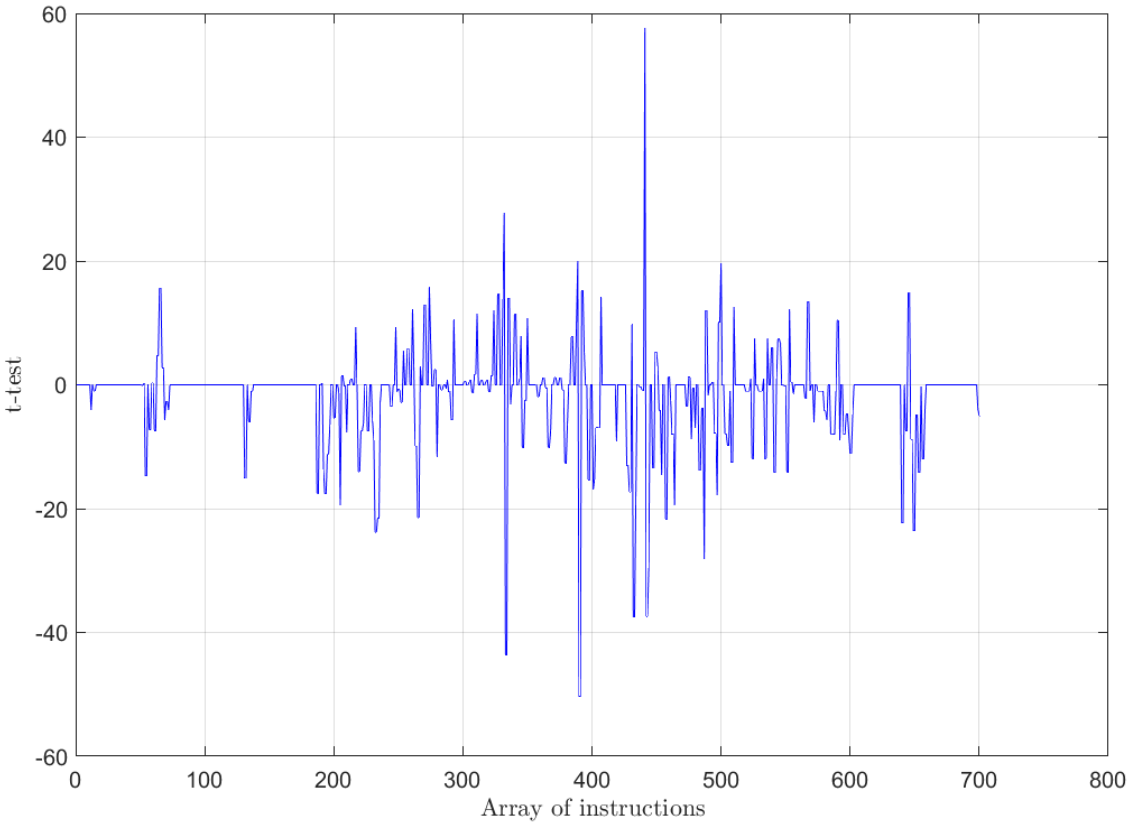


Рис. 4: Результаты применения статистического t-теста в среде ELMO для одного раунда шифра Магмы.

Инструкция, соответствующая наибольшему пику на рисунке 4: *0x0800018A: 0x2A10 cmp r2, #0x10*, в которой осуществляется сравнение значения в регистре $r2$ с константой $0x10$.

Выявление утечки по энергопотреблению для криптографического алгоритма Кузнечик

В 2015 году на территории Российской Федерации утверждён национальный стандарт симметричного шифрования ГОСТ Р 34.12-2015, описывающий блочных шифр Кузнечик [4].

В рамках данного проекта был реализован на языке Си базовый вариант одного раунда данного шифра. По итогам выполнения одного раунда шифра Кузнечик на эмуляторе ELMO была получена трасса энергопотребеления, представленная на рисунке 5.

Также был проведён анализ на базе теста `FixedvsRandom`. По формуле 5 была вычислена $t-$мера для одного раунда, график изменения значений t показан на рисунке 6.

Результаты анализа:

- Число раундов - 10;
- Общее число трасс - 200;
- общее число цислов/инструкций - 381598;
- общее число инструкций, содержащих статистическую зависимость (утечку) 168562 (44.2%).

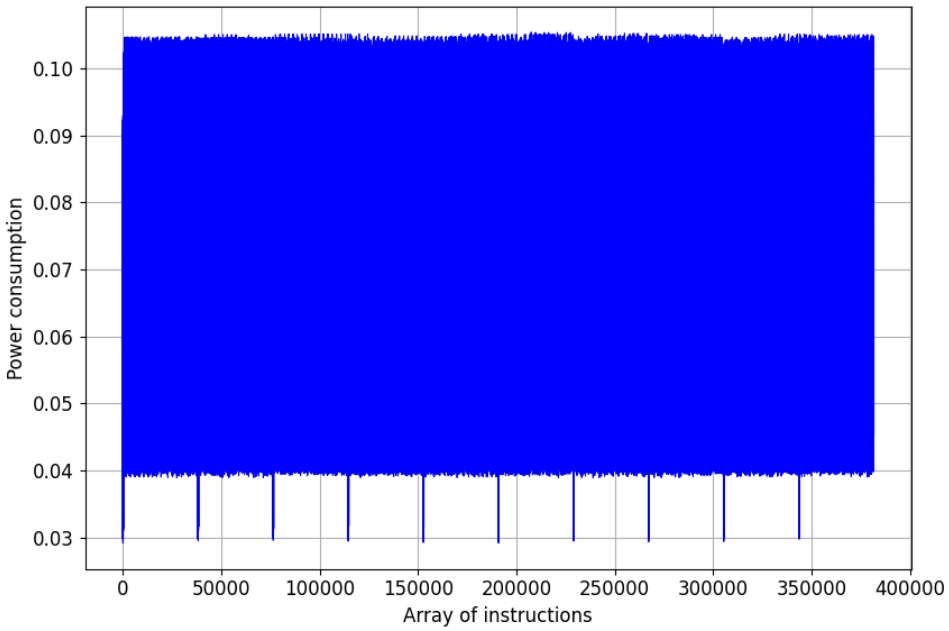


Рис. 5: Сгенерированная в среде ELMO трасса энергопотребления при выполнении десяти раундов шифра Кузнечик.

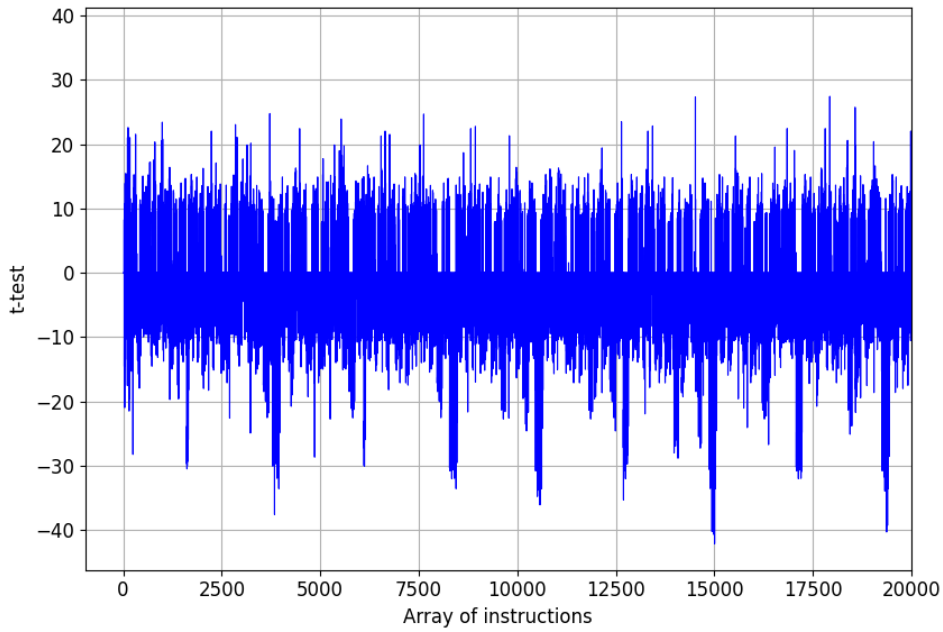


Рис. 6: Часть результата применения статистического t-теста в среде ELMO для десяти раундов шифра Кузнечик.

Метод дифференциального анализа побочных каналов по энергопотреблению (средство DPA Simulator)

DPA (Differential Power Analysis) Simulator представляет собой инструмент, предназначенный для моделирования атак по побочным каналам на симметричные блочные алгоритмы шифрования методом дифференциального анализа. DPA Simulator [10] позволяет исследователям задавать различные сценарии (модели) атак по побочным каналам и производить оценку уязвимости моделируемых криптографических систем. Данное программное обспечение эмулирует выполнение криптографического алгоритма на линейке Cortex-M с использованием библиотеки libthumb2sim, генерирует трассы энергопотребления и позволяет проводить их анализ на основе модели расстояния Хэмминга.

При дифференциальных атаках по энергопотреблению первого порядка изучаются имеющиеся трассы энергопотребления и в каждый момент времени вычисляются отдельные статистические характеристики этих сигналов. В случае атак более высокого порядка анализируются совместные статистические свойства потребления энергии на протяжении нескольких периодов выборки. Для дифференциального анализа энергопотребления порядка n используется n различных выборок значений энергопотребеления, которые соответствуют n разным промежуточным значениям выполнения алгоритма шифрования.

Атаки по энергопотреблению основываются на предположении, что различные криптографические операции требуют для их выполнения различное количество потребляемой устройством энергии, в том числе в зависимости от данных, обрабатываемых в конкретной инструкции. Анализ разности изменений энергопотребеления может содержать информацию о статистической зависимости (утечке), связанной со значениями внутренних состояний криптографического средства [11].

Процесс атаки на основе дифференциального анализа трасс энергопотребления включает в себя следующие шаги:

1. Проведение достаточного количества операций шифрования на фиксированном ключе.
2. Анализ графиков энергопотребления для каждой криптографической операции.
3. Фиксирование выходного значения для точки атаки (point of interest) и использование данного значения в качестве разделителя трасс энергопотребления на группы.
4. Вычисление вероятности для различных значений битов (блоков) ключа (начиная с least significant bit (LSB)), используя сформированную модель и статистику утечки.
5. Поиск максимального среднего значения по группам трасс энергопотребеления и определение соответствующего вероятного значения бита ключа.
6. Определение для каждого последующего бита (блока) ключа наиболее вероятного значения.

Успешность атаки методом дифференциального анализа побочных каналов зависит от множества факторов, таких как: типы используемых криптографических преобразований, структура алгоритма, объем доступных к анализу данных о трассах энергопотребеления, уровень шума при измерениях энергопотребления.

ЛИТЕРАТУРА

[1] David McCann, Elisabeth Oswald, and Carolyn Whitnall. 2017. Towards practical tools for side channel aware software engineering: grey box' modelling for instruction leakages. In Proceedings of the 26th USENIX Conference on Security Symposium (SEC'17). USENIX Association, USA, 199–216.

[2] Statistical leakage simulator for the ARM M0 family ELMO. URL: https://github.com/sca-research/ELMO

[3] D. Welch. Thumbulator. URL: https://github.com/dwelch67/thumbulator.git

[4] ГОСТ Р 34.122015 Информационная технология. Криптографическая защита информации. Блочные шифры. // https://tc26.ru/standard/gost/GOST_R_3412-2015.pdf

[5] CSA ISO/IEC 17825-2018 Information technology - Security techniques - Testing methods for the mitigation of non-invasive attack classes against cryptographic modules.

[6] Github — Mbed-TLS/mbedtls. — An open source, portable, easy to use, readable and flexible TLS library, and reference implementation of the PSA Cryptography API. URL: https://github.com/Mbed-TLS/mbedtls

[7] Goodwill G., Jun B., Jaffe J. and Rohatgi P. «A testing method-ology for side-channel resistance validation», NIST Non-Invasive At-tack Testing Workshop, Sept. 2011 // http://csrc.nist.gov/newsevents/non-invasive-attack-testing-workshop/papers/08Goodwill.pdf

[8] Cooper J., DeMulder E., Goodwill G., Jaffe J., Kenworthy G. and Rohatgi P. «Test vector leakage assessment (tvla) methodology in practice», International Cryptographic Module Conference, 2013 // http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf

[9] Dworkin, M. , Barker, E. , Nechvatal, J. , Foti, J. , Bassham, L. , Roback, E. and Dray, J. (2001), Advanced Encryption Standard (AES), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.FIPS.197 (Accessed August 16, 2023)

[10] Johannes Bauer. Dpa-simulator // https://github.com/ URL: https://github.com/johndoe31415/dpa-simulator //

[11] Masaya Yoshikawa and Toshiya Asai. DPA Attacks Simulator against Cryptography System on Algorithm Design Phase // https://www.iaeng.org/ URL: https://www.iaeng.org/publication/WCECS2011/WCECS2011_pp792-796.pdf

Куратор исследования –
к.т.н., доцент Института компьютерных технологий и информационной безопасности Южного федерального университета, Екатерина Александровна Маро.

Практический криптоанализ RSA

К.А. Лернер[1], Д.А. Попков[1], К.А. Волков[1], Ю.Ф. Болтнев[1], А.Б. Мудрич[2]
[1]БФУ им. Канта
[2]НИУ ИТМО
E-mail: ksyhalerner@gmail.com, dmitry-popkov@mail.ru, mr.voki@yandex.ru, iboltnev@kantiana.ru, mudrich_ab@niuitmo.ru

Аннотация

В данной работе представлены результаты исследования основных алгоритмов атаки на криптографическую систему RSA - асимметричный алгоритм шифрования с открытым ключом. Для проверки устойчивости криптосистемы использовались алгоритмы прямого перебора (BruteForce), метод факторизации Ферма, атака Винера, метод факторизации модулей RSA (Batch GCD), p-1 метод Полларда, метод Ленстры, метод факторизации модуля RSA с помощью неподвижной точки. Каждый из представленных алгоритмов был реализован программно на языке SageMath и Python и протестирован на открытом ключе, представляющем собой произведение двух простых чисел. Криптографическая система RSA показала свою устойчивость ко всем видам протестированных атак, при этом даже в случае успешной реализации подбора ключа алгоритм требовал значительного времени выполнения или соблюдения специальных условий для реализации (например, алгоритм Полларда выполняется только на числах, у которых p-1 хорошо раскладывается на множители).

Ключевые слова: *RSA, криптографическая система с открытым ключом, атака.*

Введение

Целью проекта является исследование процедуры экспресс-анализа открытого ключа криптосистемы RSA [1]. Команде было предложено рассмотреть известные алгоритмы атаки на криптосистему RSA: метод прямого перебора (BruteForce), метод факторизации Ферма, атака Винера, метод факторизации модулей RSA (Batch GCD), p-1 метод Полларда, метод Ленстры, метод факторизации модуля RSA с помощью неподвижной точки. Задачей номер один стала программная реализация данных алгоритмов на языках программирования SageMath и Python, и получение замеров времени их работы. Вторая задача — анализ стойкости криптосистемы RSA на данные алгоритмы атак. Тестирование основной части алгоритмов на открытом ключе требует значительного времени выполнения, поэтому с целью экономии вычислительных ресурсов замеры производились на неполном объеме прохода циклов для подбора ключа и полученное время полной работы алгоритма имеет приблизительный характер.

Описание алгоритмов атаки

Метод прямого перебора (BruteForce)

Данный метод относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий. Именно с этой проблемой и сталкивается данный метод вовремя реализации атаки на криптосистему RSA. При изучении учебных примеров данный метод показывает себя весьма положительно, но как только дело касается применения на настоящих

модулях сертификатов сайтов, время, затраченное на подборку двух простых множителей p и q, может исчисляться от нескольких месяцев до десятков лет[4].

Метод факторизации Ферма

В основе метода лежит теорема о представлении числа в виде разности двух квадратов: для нечетного n > 1 существует однозначное разложение на множители $n = a \cdot b$ и соответствующее ему представление в виде разности квадратов $n = x^2 - y^2$. Формулы задают следующие соотношения: $x = \frac{a+b}{2}, y = \frac{a-b}{2}$ при $a = x + y, b = x - y$.

Реализация метода заключается в поиске целых чисел, удовлетворяющих условию представления числа в виде разности квадратов, описанной в теореме выше. По условию теоремы $x^2 - n$ является квадратом. Сначала находится наименьшее число, при котором $x^2 - n$ принимает положительные значения. Для каждого последующего значения $n^{\frac{1}{2}} + k$ вычисляется его квадрат и проверяется, является ли число точным квадратом. Если число является точным квадратом, то выполняется условие теоремы о представлении числа в виде разности двух квадратов: $n = x^2 - y^2 = (x-y)(x+y)$. В случае, если число является тривиальным (в разложении присутствует сомножитель = 1), то n является простым числом.

Значение выражения на последующих шагах вычисляется на основе значений, полученных ранее по формуле: $(k+1)^2 - n = k^2 + 2k + 1 - n = (k^2 - n) + 2k + 1$.

Наиболее эффективно алгоритм работает при n, полученным в результате умножения двух близких сомножителей.

Атака Винера

Атака Винера основывается на следующей теореме.
Теорема Винера: Пусть задана криптосистема RSA с параметрами $N = p \cdot q$, $d \cdot e \equiv 1 (\text{mod } \phi(N))$, $q < p < 2q$ $d < \frac{1}{3} N^{\frac{1}{4}}$ Тогда ключ d эффективно вычислим.
Алгоритм атаки Винера:
ВВОД: $N, e, M, C = M^e (\text{ mod } N)$
ВЫВОД: $p, q, d : ed \equiv 1 \text{ mod } (\phi(N))$
1.Положить $\alpha = \frac{e}{N}$.
2.Разложить α в цепную дробь: α = $[q_0, q_1, ..., q_i, ..., q_s]$, $s < 2log_2 N$.
3.Для i = 1, 2, ... вычислить:
3.1.$P_i, Q_i : \delta_i = \frac{P_i}{Q_i}$ — i-я подходящая дробь к α.
3.2.Если $C^{Q_i} \equiv M (\text{ mod } N)$, то вернуть $d = Q_i, k = P_i$.
Затем рассчитывается $\phi(N) = \frac{e \cdot d - 1}{k}$.
Последним шагом нужно найти корни квадратного уравнения: $x^2 - ((N - \phi(N)) + 1) \cdot x + N = 0$, где корни уравнения будут искомым разложением n на множители.

Batch GCD

Этап I. В мире, где лучшая из доступных процедур умножения является квадратичной (т.е. умножение двух чисел длины n занимает $\Omega(n^2)$ времени), вычисление гигантского произведения будет квадратичным по длине $N = O((mn)^2)$, независимо от того, как мы секционируем и фрагментируем входные данные. Этап II. После того, как мы вычислили гигантское произведение N с помощью бинарного дерева, нам необходимо вычислить $\gcd(N_i^2, N)$ для всех i, но наивное решение этой задачи снова потребует квадратичного времени. Заметим, что $\gcd(N_i^2, N) = \gcd(N_i^2, \text{ mod } N_i^2)$,

поэтому вместо вычисления m GCD, где одно число является огромным, мы сначала выполняем m модульных сокращений, а затем m экземпляров 2n-разрядных GCD. Мы будем использовать простое уравнение $N \bmod a = (N \bmod ab) \bmod a$, что предполагает использование бинарного дерева. Этап III. После построения дерева остатков можно вычислить $G_i = \gcd(N_i^2, N \bmod N_i^2)$. Для большинства $G_i = N_i$, это означает, что N_i соизмерим со всеми остальными модулями в наборе данных. Для некоторых значений N_i можно сразу отнести к $N_i < G_i < N_i^2$, так как в этом случае $\gcd(\frac{G_i}{N_i}, N_i)$ нетривиально. Это позволит устранить те N_i, для которых один из простых коэффициентов является общим с каким-либо другим модулем. Остается ничтожное меньшинство N_i, для которых все их простые коэффициенты появляются где-то еще, и, следовательно, $G_i = N_i^2$. Для этих чисел, поскольку их мало, можно просто вычислить их GCD со всеми факторизованными N_i.

p-1 метод Полларда

Метод очень быстро находит простые делители малой и средней величины (до 20-25 десятичных цифр).
Алгоритм Полларда:
1. Вычислим $m =$ НОК $(1, 2, ..., B)$
2. Пусть $a = 2$
3. Вычислим $x = a^m - 1 \bmod n; p =$ НОД (x, N)
4. Если $p \neq 1$ или n, то выводим p и завершаем работу.
5. Если $a < 20$, то $a = a + 1$ и перейти к шагу 3. Иначе завершить работу и выбрать другое B.

Метод факторизации Ленстры на эллиптических кривых

Метод является обобщением предыдущего, но основан на предположении о гладкости мощности множества точек случайной эллиптической кривой, определенной $\bmod N$.

Дано положительное целое число n и граница b, этот алгоритм пытается найти нетривиальный множитель p для N. Вычислим m равное наименьшему общему кратному чисел от 1 до b. Выберем случайное $a \in \mathbb{Z}\backslash n\mathbb{Z}$, такое что $4a^3 + 27 \in (\mathbb{Z}\backslash n\mathbb{Z})^*.P = (0, 1)$ - это точка эллиптической кривой над $\mathbb{Z}\backslash n\mathbb{Z}$. Вычисляем mP используя алгоритм умножения точки на число. Если в некоторой точке мы не можем вычислить сумму точек, то мы вычисляем НОД этого знаменателя с n. Если НОД является нетривиальным делителем, выводим его. Если каждый знаменатель взаимно прост с n, выводим "Fail".

Если $mP = O$, то вычисляем НОД $(x_1 - x_2, n)$ или НОД $(2y_1, n)$, если $1 <$ НОД $(x_1 - x_2, n) < n$, то выводим и завершаем. Иначе снова выбираем случайное a. [2]

Метод факторизации модулей с помощью неподвижной точки

Использование неподвижных точек в криптосистеме RSA

В RSA существуют неподвижные точки, которые при шифровании совпадают с оригинальным сообщением: $F^e \equiv F \bmod n$. Это равносильно системе

$$\begin{cases} F^e \equiv F \bmod p \\ F^e \equiv F \bmod q. \end{cases}$$

Если НОД$(F,n) \neq 1$, то НОД$(F,n) = p$ или q; Если НОД$(F,n) = 1$, то $F^{e-1} \equiv 1 \bmod n$. Это равносильно системе

$$\begin{cases} F^{e-1} \equiv 1 \bmod p, F^{e-1} \equiv 1 \bmod q. \end{cases}$$

$F \bmod p$ - корень уравнения $x^{e-1} - 1 = 0$ в $\mathbb{Z}_p$. Для решения таких уравнений и нахождения неподвижных точек существует следующий алгоритм.

Алгоритм нахождения неподвижных точек криптосистемы RSA

Вход: p, q, e (p, q – простые, НОД$(e, \phi(n)) = 1, n = p \cdot q$)
Выход: B – количество неподвижных точек; [F1, F2,..] – список неподвижных точек.
1) Вычисление количества неподвижных точек. B $= (1+\delta_p)\cdot(1+\delta_q)$, где $\delta_p =$ НОД$(e-1, p-1); \delta_q =$ НОД$(e-1, q-1)$
2) Вычисление g_p, g_q, где g_p – образующая по модулю p; g_q – образующая по модулю q.
3) Вычисление w_p и w_q, где w_p – примитивный корень из единицы степени δ_p по модулю p; w_q – примитивный корень из единицы степени δ_q по модулю q. $w_p = g_p^{\frac{(p-1)}{\delta_p}} \bmod p$; $w_q = g_q^{\frac{(q-1)}{\delta_q}} \bmod q$
4) Вычисление P_i, Q_j, - неподвижных точек по mod p и mod q. $P_i = w_p^i \bmod p, i = 0...\delta_p - 1$, P_{δ_p}= 0; $Q_j = w_q^j \bmod q, j = 0...\delta_q - 1, Q_{\delta_q} = 0$
5) Решение системы сравнений:

$$I_p - \text{ решение системы } \begin{cases} x \equiv 1 \bmod p, \\ x \equiv 0 \bmod q. \end{cases}$$

$$I_q - \text{ решение системы } \begin{cases} x \equiv 0 \bmod p, \\ x \equiv 1 \bmod q. \end{cases}$$

$I_p = q' \cdot q \bmod n,$ где $q' = q^{-1} \bmod p = q^{p-2} \bmod p$
$I_q = p' \cdot p \bmod n,$ где $p' = p^{-1} \bmod q = p^{q-2} \bmod q$
6) Вычисление неподвижных точек RSA.
$F_{ij} = I_p \cdot P_i + I_q \cdot Q_j,$ где $i = 0...\delta_p; j = 0...\delta_q$
F_{ij} фактически является решением системы

$$\begin{cases} x \equiv P_i \bmod p, \\ x \equiv Q_j \bmod q. \end{cases} \tag{6}$$

Неудачный выбор открытого ключа

Как видно из предыдущего алгоритма, количество неподвижных точек всегда не менее 9, из них три тривиальные: F = 0, 1, -1 При специально построенном открытом ключе e все точки могут быть неподвижными.
Теорема Если $e = j \cdot$ НОК $(p-1, q-1) + 1$, где $j = 0, 1, 2, ..(\phi(n)/$ НОК $(p-1, q-1) - 1)$, то для всех $M \in (0, n-1)$ имеем $M^e = M \bmod n$.
Замечание. Количество таких e в условиях теоремы равно $\phi(n)/$ НОК $(p-1, q-1) =$ НОД $(p-1, q-1)$.

Использование неподвижных точек для криптоанализа RSA

Идея метода: криптоаналитик ищет неподвижные точки F методом перебора, затем находит НОД(F, n), НОД$(F^r + 1, n)$, НОД$(F^r - 1, n)$ при некотором r. В большинстве случаев он получит в результате p или q. Заметим, что НОД находится по алгоритму Евклида, т. е. не требует разложения на n множители. Алгоритму требуется $O(log\ n)$ итераций.
Как отмечено в алгоритме, неподвижные точки можно представить формулой: $F_{ij} = I_p \cdot P_i + I_q \cdot Q_j$
Случай 1: Если $P_i = 0$ то неподвижная точка находится из решения системы:

$$\begin{cases} x \equiv 0 \bmod p, \\ x \equiv Q_j \bmod q. \end{cases}, \text{ следовательно, НОД}(F, n) = p.$$

Случай 2: Если $P_i^r = 1$. Тогда $\begin{cases} x \equiv 1 \bmod p, \\ x \equiv Q_j \bmod q. \end{cases}$, следовательно, НОД$(F^r - 1, n) = p$

Случай 3: Если $P_i^r = -1$. Тогда $\begin{cases} x \equiv -1 \bmod p, \\ x \equiv Q_j \bmod q. \end{cases}$, следовательно, НОД$(F^r + 1, n) = p$

В качестве показателя r берется один из делителей $e - 1$. Аналогичные рассуждения можно провести и для Q_j.
Пример. $p = 1361, q = 29803, e = 127,\ n = p \cdot q = 1361 \cdot 29803 = 40561883$. Вычисляются $\delta_p =$ НОД $(127-1, 1361-1) = 2; \delta_q =$ НОД $(127-1, 29803-1) = 6, I_p = 33558178, I_q = 7003706$. Криптоаналитик методом перебора находит неподвижную точку $F = 27300300$. Для нее выполняется $F = 27300300 = 1 \cdot I_p + 752 \cdot I_q$. Тогда НОД$(F-1, n)$=НОД$(27300300-1, 40561883) = 1361 = p$. Отсюда $q = \frac{n}{p} = 29803$. Криптоаналитик получает p и q и взламывает криптосистему.

Временные показатели генерации данных

Для замеров скорости работы выполнения алгоритмов использовался персональный компьютер со следующими характеристиками: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz, 6 ядер, 12 потоков, с оперативной памятью 16 ГБ на 2133 ГГц.

Полученные результаты

Для каждого из описанных алгоритмов была подготовлена его программная реализация на языках SageMath и Python. Результаты работы алгоритмов и данные, на которых они были протестированы представлены в таблице ниже.

Таблица 1. Результаты работы алгоритмов

Алгоритм	Входные данные	Время выполнения	Полученные результаты
Метод прямого перебора (BruteForce)	Модуль 2048 бит	12 часов	Проверено 3,9 тыс. чисел, принудительно завершен без получения успешного результата
Метод факторизации Ферма	Модуль 2048 бит.	25 часов	Проверено более 24 млрд. чисел, принудительно завершен без получения нужного результата
Атака Винера	Учебный пример	<1 сек	Согласно теореме Винера битовая длина секретного ключа d должна быть в 4 раза меньше длины модуля n
Batch GCD	Модуль 2048 бит	12 часов	Принудительно завершен без получения нужного результата
p-1 метод Полларда	Учебный пример	<1 сек	Результат выполнения алгоритма зависит от значения выбранного показателя гладкости B
Метод факторизации Ленстры на эллиптических кривых	Учебный пример	<1 сек	Удачное завершение, если N имеет простой делитель p, для которого Z_p состоит только из небольших простых чисел
Метод факторизации модуля RSA с помощью неподвижной точки	Учебный пример	<1 сек	Успешное завершение, если известна неподвижная точка

Вывод

По результатам проведенного тестирования криптографическая система RSA показала свою устойчивость к основным видам атак. Для успешного проведения атаки требуется длительное время выполнения алгоритма, что делает данные методы нецелесообразными, или обязательное соответствие входных данных определенным условиям, из-за чего сужается область применения алгоритма при проведении атаки.

ЛИТЕРАТУРА

[1] С. Коутинхо Введение в теорию чисел. Алгоритм RSA / С. Коутинхо — . — Москва: Постмаркет, 2001 — 328 .

[2] О. Н. Василенко Теоретико числовые методы в криптографии / О. Н. Василенко — . — Москва: МЦНМО, 2003 — 326 с.

[3] Сонг Ю. Ян Тестирование на простоту и факторизация в криптографии с открытым ключом / Сонг Ю. Ян — . — : Springer, 2009 — 386 с.

[4] Christof Paar, Jan Pelzl Understanding Cryptography: A Textbook for Students and Practitioners / Christof Paar, Jan Pelzl — 1-е. — : Springer, 2010 — 390 с.

Кураторы исследования –

Гребнев Сергей Владимирович – ведущий криптограф-исследователь QApp, Российский квантовый центр (г. Москва);

Шапоренко Александр Сергеевич – старший преподаватель НГУ (г. Новосибирск);

SAT-РЕШАТЕЛИ В КРИПТОГРАФИИ

SAT-криптоанализ поточного шифра Trivium и легковесных блочных шифров Simon и Speck

Д. А. Соколова[1], Д. А. Быков[2]
[1]Колледж информационных технологий
[2]Новосибирский государственный университет
E-mail: sokolovad46@gmail.com, den.bykov.2000i@gmail.com

Аннотация

Целью данного проекта является алгебраический криптоанализ ослабленных версий поточного шифра Trivium, а также легковесных шифров Simon и Speck [1]. Данные задачи были сведены к задаче поиска выполняющего набора булевых формул (SAT). Были построены SAT-кодировки шифра Trivium с помощью программного средства CBMC. Используя SAT-решатель KisSat [2] были решены ослабленные задачи криптоанализа, в которых примерно половина из 288 бит искомого заполнения генератора ключевого потока была известна. Для Simon и Speck были выбраны SAT-решатели KisSat и MergeSat [3]. В качестве подхода к предобработке SAT-задачи, описывающей поиск ключей шифрования для произвольно выбранной пары открытый текст и шафротекст, в работе предложено использовать средство Bosphorus [4]. Проведен поиск выполняющих наборов на SAT-решателях для двух реализаций конвертации задачи из алгебраической нормальной формы (АНФ) представления в конъюнктивную нормальную форму (КНФ): anf2cnf [5] и Bosphorus.

Ключевые слова: *Алгебраический криптоанализ, поточный шифр, блочный шифр, Speck, Simon, Trivium, SAT-решатель*

Введение

Алгебраический криптоанализ — вид криптоанализа, в котором по известному алгоритму шифрования строится система алгебраических уравнений [6]. Путем подстановки в такую систему уравнений значений переменных, которые соответствуют шифртексту и (если необходимо) открытому тексту, можно получить модифицированную систему уравнений, которая соответствует задаче поиска секретного ключа. Существуют различные подходы к решению систем алгебраических уравнений. Одним из них является построение соответствующей булевой формулы в конъюнктивной нормальной форме (КНФ) и использование SAT-решателей. Для некоторых случаев SAT-решатели позволяют найти выполняющий набор КНФ, т.е. набор значений переменных, который обращает КНФ в значение Истина. Из этого набора можно эффективно получить искомый секретный ключ. Такой подход называется SAT-криптоанализом.

В данном исследовании рассматриваются три шифра: поточный шифр Trivium [7] и легковесные блочные шифры Simon и Speck [1]. Каждый из этих шифров в полном виде слишком сложен для практического криптоанализа, поэтому формируются их ослабленные версии, к которым применяется SAT-криптоанализ. Опираясь на результаты конкурса SAT-решателей «SAT Competition 2022» [9], для проведения исследования были выбраны финалисты конкурса — KisSat и MergeSat. Для решения SAT-задач по шифру Trivium был использован только KisSat, а для Simon и Speck были применены оба решателя.

SAT-криптоанализ легковесных блочных шифров Simon и Speck

Шифр `Simon`

Легковесный блочный шифр Simon разработан для эффективной программной и аппаратной реализации. Схема шифра Simon представляет собой LRX-структуру, как показано на рисунке 1. Криптографические алгоритмы данной структуры используют следующие операции: логические (побитовое AND, &); операции циклического сдвига; сложение по модулю 2 (побитовое XOR, $\oplus$). Блочный шифр Simon имеет n-битовое слово и 2n-битовый блок и обозначается: Simon2n, где параметр n может принимать значения – 16, 24, 32, 48 или 64. Алгоритм шифрования Simon2n с длиной ключа mn-бит обозначается - Simon2n/mn. Параметры шифра Simon показаны на рисунке 2. Параметры системы нелинейных уравнений и SAT-задачи, описывающие преобразование шифра Simon приведены в таблице 3. Результаты применения SAT-решателей представлены в таблице 4.

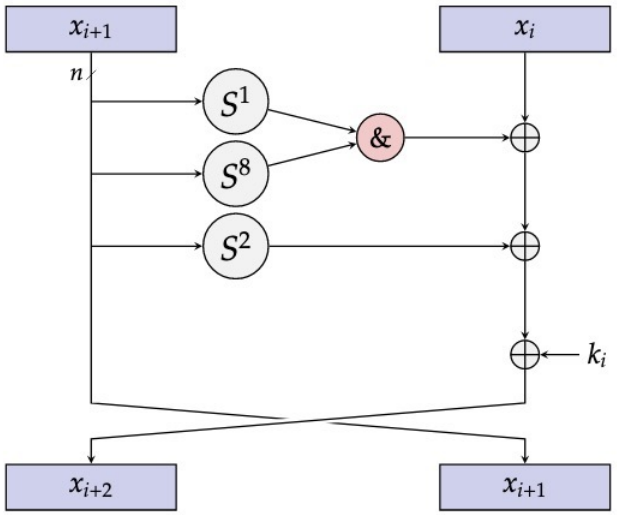


Рис. 1: Описание раундовых преобразований шифра Simon

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 шифра Simon

Ключ шифрования задается в виде первых m ключей длины n бит каждый. Последующие раундовые ключи вычисляются по формуле:

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+1}, \text{ for } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+2}, \text{ for } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1}) (S^{-3} k_{i+3} \oplus k_{i+1}), \text{ for } m = 4. \end{cases} \quad (7)$$

где c - константа, z_j — фиксированная периодическая последовательность.

В проекте рассматривется уменьшенная версия алгоритма Simon с параметрами $2n = 32$, $m = 2$. В настоящей работе был также использован конвертор Bosphorus для шифров Simon и Speck. Bosphorus – средство, позволяющее упростить исходную задачу, представленную в АНФ. Bosphorus поддерживающее работу с описанием упрощаемых задач как в АНФ, так и КНФ, также использваться как препроцессор для SAT-задачи. Bosphorus использует итеративную архитектуру, представленную на рисунке 3, которая выполняет следующий набор шагов, либо до тех пор, пока не закончится время, либо до фиксированной точки:

1) Замена переменных и распространение констант в АНФ.

2) Запуск ограниченной расширенной линаризации (XL) и ввод обратной единицы и двоичных XOR.

3) Запуск ограниченного ElimLin.

4) Преобразование в КНФ, запуск SAT-решителя для ограниченного количества конфликтов.

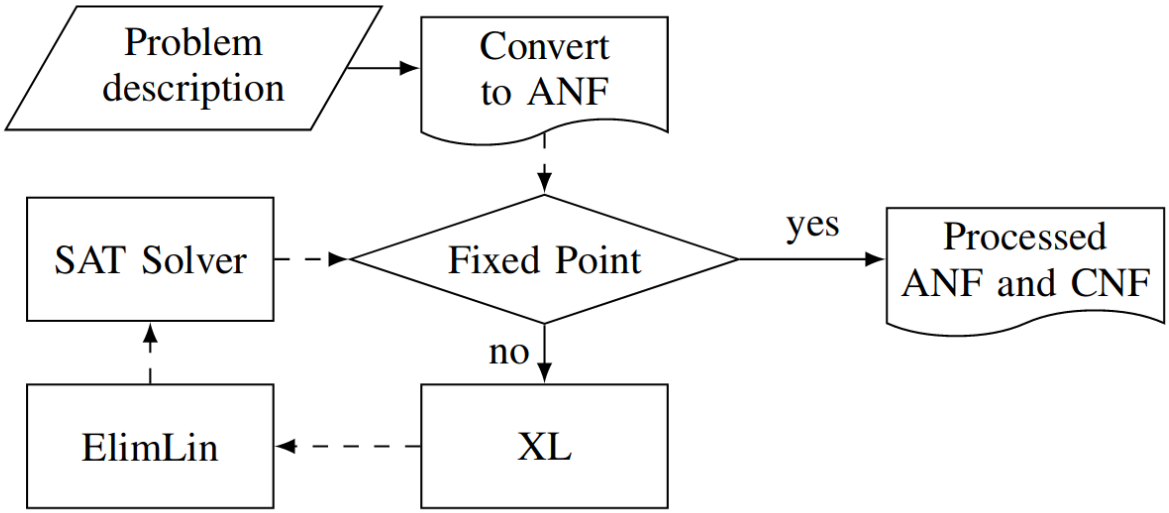


Рис. 3: Набор шагов, выполняемый архитектурой Bosphorus

Таблица 3: Параметры системы нелинейных уравнений и SAT-задачи, описывающих преобразования шифра Simon.

Параметры	Кол-во уравнений	Кол-во неизв.	Параметры КНФ (конвертор anf2cnf)	Параметры КНФ (конвертор Bosphorus)
T = 7	112	112	320 лит., 3632 клоз	889 лит., 24250 клоз
T = 8	128	128	368 лит., 4448 клоз	855 лит., 24438 клоз
T = 9	144	144	480 лит., 6448 клоз	1364 лит., 38622 клоз
T = 10	160	160	560 лит., 8096 клоз	2008 лит., 59908 клоз
T = 11	176	176	640 лит., 9648 клоз	2077 лит., 61686 клоз

Шифр `Speck`

Легковесный блочный шифр `Speck` имеет ARX-структуру (add-rotate-xor), показанную на рисунке 4. Speck применяет операции сложения по модулю 2 (побитовое XOR, $\oplus$); сложения по модулю 2n, $\oplus$; циклический сдвиг левой части на -α бит и правой части на β бит. Параметры отображены на рисунке 5. Параметры системы нелинейных уравнений и SAT-задачи, описывающие преобразо-

Таблица 4: Результаты применения SAT-решателей для анализа преобразований шифра Simon.

Параметры	Время KisSat (anf2cnf), с	Время KisSat (Bosphorus), с	Время MergeSat (anf2cnf), с	Время MergeSat (Bosphorus)
T = 7	1507.10	4376.56	48.02	1361.63

вание шифра Speck приведены в таблице 5. Результаты применения SAT-решатей представлены в таблице 6.

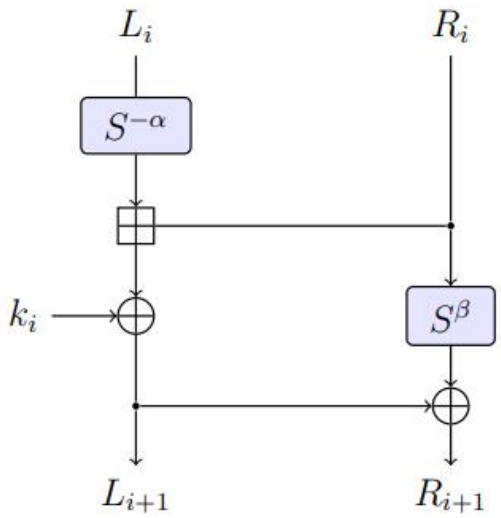


Рис. 4: Описание раундовых преобразований шифра Speck

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. 5: Параметры шифра Speck

В проекте рассматривается уменьшенная версия алгоритма Speck с параметрами $2n = 32, m = 2$.

SAT-криптоанализ поточного шифра Trivium

`Trivium` — синхронный поточный шифр, ориентированный на аппаратную реализацию [7], один из победителей eSTREAM. Он специально проектировался как шифр с простой структурой относительно других поточный шифров. Он способен генерировать до 2^{64} бит ключевого потока. Параметры шифра указаны в таблице , причём внутренне состояние разбито на 3 регистра сдвига снелинейной обратной связью длинами 93, 84 и 111 бит.

Такт работы задается следующими уравнениями:

$$a_i = c_{i-66} + c_{i-111} + c_{i-110} \cdot c_{i-109} + a_{i-69}$$
$$b_i = a_{i-66} + a_{i-93} + a_{i-92} \cdot a_{i-91} + b_{i-78}$$
$$c_i = b_{i-69} + b_{i-84} + b_{i-83} \cdot b_{i-82} + c_{i-87}$$

Таблица 5: Параметры системы нелинейных уравнений и SAT-задачи, описывающих преобразования шифра Speck.

Параметры	Кол-во уравнений	Кол-во неизв.	Параметры КНФ (конвертор anf2cnf)	Параметры КНФ (конвертор Bosphorus)
T = 3	500	176	1460 лит., 11020 клоз	1460 лит., 1460 клоз
T = 4	782	320	2492 лит., 17380 клоз	3386 лит., 59123 клоз
T = 5	1032	416	3312 лит., 23184 клоз	4929 лит., 93724 клоз
T = 6	1282	512	4132 лит., 28988 клоз	6490 лит., 127004 клоз

Таблица 6: Результаты применения SAT-решателей для анализа преобразований шифра Speck.

Параметры	Время KisSat (anf2cnf), с	Время KisSat (Bosphorus), с	Время MergeSat (anf2cnf), с	RAM KisSat (anf2cnf), Мбайт	RAM KisSat (Bosphorus), Мбайт
T = 3	0.03	0.005	0.09	3240	1052
T = 4	8.77	43.89	52.92	10488	18252

Таблица 7: Параметры Trivium

Параметры	
Ключ	80 бит
IV	80 бит
Состояние	288 бит

Инициализация начальных значений:

$$(a_{-1245}, \ldots, a_{-1153}) = (0, \ldots, 0, k_0, \ldots, k_{79})$$
$$(b_{-1236}, \ldots, b_{-1153}) = (0, \ldots, 0, IV_0, \ldots, IV_{79})$$
$$(c_{-1263}, \ldots, c_{-1153}) = (1, 1, 1, 0, \ldots, 0)$$

После 1152 тактов работы шифр начинает выдавать выходные биты по правилу: $z_i = c_{i-66} + c_{i-111} + a_{i-66} + a_{i-93} + b_{i-69} + b_{i-84}$

Рассматривается упрощенный вариант `Trivium`, в котором опущена стадия инициализации, то есть первые 1152 такта. При этом вместо заданного начального заполнения внутреннего состояния ключем и IV рассматривается произвольное заполнение регистра. Отметим, что полном шифре после стадии инициализации заполнение внутреннего состояния близко к случайному.

Для генерации КНФ использовалось CBMC [7] — средство верификации ограниченных моделей для программ на языках C и C++. Конкретнее, была задействована версия 5.89. Для поиска выполняющих наборов полученных КНФ использовался SAT-решатель KisSat [2] версии 3.0.

Ослабленная версия шифра, генерирующая 300 бит выходного потока, была представлена в виде программы на языке C. Далее по ней с помощью CBMC можно сгенерировать КНФ в формате Dimacs, связывающую 288 бит начального заполнения внутреннего состояния с 300 битами выходного потока. Отметим, что для представления бита в программе используется тип `__CPROVER_bool`, вводимый CBMC; а для генерации фиктивного свойства, которое CBMC будет пытаться доказать, используется `__CPROVER_assert(0, '' '')`. Означить некоторые переменные мы можем как напрямую в полученной КНФ, добавлением соответствующего однолитерального дизъюнкта, так и добавлением в исходную программу на C выражения `__CPROVER_assume(var == value)`. Нами использовался второй вариант.

Для проверки корректности полученной КНФ было взято несколько пар вход-выход шифра без стадии инициализации, означены соответствующим образом переменные и на полученной КНФ запущен SAT-решатель. Так как означивание переменных соответствует реальным парам вход-выход шифра, то правильная КНФ с означенными переменными должна быть выполнима и проверка этого факта происходит практически мгновенно.

SAT-криптоанализ может основываться на атаке на открытых текстах: по известному выходному потоку требуется получить начальное состояние генератора. Из этой задачи можно получить следующую подзадачу: сколько (первых) бит начального заполнения регистра нужно знать, чтобы с помощью SAT-решателя можно было за разумное время вычислить оставшиеся.

Для её решения был написан следующий скрипт.

```
1. Принимает на вход timeout -- ограничение на время работы SAT-решателя
2. Генерирует случайный 288-битный вход и вычисляет соответствующую 300-битную
   гамму ослабленного Trivium
3. Генерирует C программу ослабленного Trivium с означенным входом и выходом
4. С помощью CBMC из C программы генерирует соответствующую CNF
5. k = 288
6. Пока SAT-решатель может решить CNF за timeout:
  7. Убирает из CNF означивание бита входа k
  8. k = k - 1
9. Возвращает k + 1
```

Таким образом, возвращается число первых бит входа, которые нужно означить (при означенном выходе), чтобы SAT-решать мог решить КНФ за отведенное время `timeout` на данном случайном входе.

На 5 запусках, чтобы уложиться в `timeout` $= 1800$ с., потребовалось следующее число означенных первых бит входа: 139 138 138 139 139. В таблице 8 показаны последние 10 времен работы SAT-решателя для одного такого запуска.

Число означиваний	148	147	146	145	144	143	142	141	140	139
Время, с.	0.22	0.50	3.63	11.13	1.18	49.49	69.77	177.25	130.10	508.16

Таблица 8: Зависимость времени решения от числа означиваний первых бит входа

Отметим, что так как означивались первые биты, то в самых сложных для SAT-решателя задачах был полностью известен первый регистр и часть второго, а третий регистр был полностью неизвестен.

ЛИТЕРАТУРА

[1] ISO/IEC 29167-21:2018 Information technology — Automatic identification and data capture techniques — Part 21: Crypto suite SIMON security services for air interface communications

[2] Biere A., Fleury M. Gimsatul, IsaSAT and KisSat entering the SAT Competition 2022 // Proceedings of SAT Competition. 2022. pp. 10-11.

[3] Mergesat SAT Solver // https://github.com/conp-solutions/mergesat

[4] Bosphorus // https://meelgroup.github.io/post/bosphorus/

[5] Algebraic Normal Form to Conjunctive Normal Form Converter // https://github.com/Daeinar/anf2cnf-converter

[6] Bard G. Algebraic Cryptanalysis. 2009.

[7] Christophe De Canniere. Trivium: A stream cipher construction inspired by block cipher design principles. In Proc. of the 9th International Conference on Information Security (ISC), volume 4176 of Lecture Notes in Computer Science, pages 171–186, 2006.

[8] Clarke E. M. , Kroening D., Lerda F. A Tool for Checking ANSI-C Programs // In Proceedings of TACAS. 2004. pp. 168-176.

[9] Balyo T., Heule M. J. H. Iser M., Järvisalo M., Suda, M. Proceedings of SAT Competition 2022: Solver and Benchmark Descriptions. Department of Computer Science, University of Helsinki. Vol. B-2022-1. 2022.

Кураторы исследования –
к.т.н., ведущий научный сотрудник ИДСТУ СО РАН, Олег Сергеевич Заикин;
к.т.н., доцент Института компьютерных технологий и информационной безопасности Южного федерального университета, Екатерина Александровна Маро.

SAT-криптоанализ криптографических хэш-функций BLAKE и GRØSTL

В.В. Давыдов[1], А.П. Кирьянова[1], О.С. Заикин[2]
[1]Университет ИТМО
[2]ИДСТУ СО РАН
E-mail: vvdavydov@itmo.ru, anastacia.kiryanova@itmo.ru, zaikin.icc@gmail.com

Аннотация

В работе представлен SAT-криптоанализ криптографических хэш-функций BLAKE и GRØSTL. Были реализованы CBMC-кодировки, получены конъюнктивные нормальные формы функций сжатия, а также получены новые результаты при поиске прообраза для неполнораундовых версий функций сжатия.

Ключевые слова: *криптографическая хэш-функция, криптоанализ, атака нахождения прообраза, SAT, конъюнктивная нормальная форма.*

Применение SAT-решателей для анализа стойкости криптографических хэш-функций

Одним из необходимых шагов при создании криптографической схемы, алгоритма или протокола является проведение первоначального криптоанализа для оценки стойкости предложенной схемы. В случае криптографических хэш-функций криптоанализ сводится либо к задаче поиска коллизий, либо к обращению хэш-функции (задача поиска прообраза). Данные задачи, в свою очередь, могут быть сведены к задаче выполнимости булевых функций (Boolean satisfiability problem, SAT) [1]. Впервые использовать SAT-криптоанализ было предложено в 1989 году [2].

Решить задачу SAT в распознавательном варианте для заданной булевой формулы означает определить, выполнима ли эта булева формула, т.е. существует ли такой набор значений ее переменных, при которых итоговое значение формулы примет истинное значение. В поисковом варианте SAT необходимо найти соответствующий выполняющий набор если булева формула выполнима. В 1971 году американским учёным Стивеном Куком было введено понятие NP-полной задачи и доказано, что SAT в распознавательном варианте для булевых формул, записанных в конъюнктивной нормальной форме (КНФ), является NP-полной [3]. На сегодняшний день для решения задачи выполнимости булевых функций на практике используются SAT-решатели. В большинстве случаев такие решатели предназначены для решения задач верификации, планирования производства и поиска комбинаторных структур, поэтому они не адаптированы под задачи криптоанализа. Тем не менее, даже на текущем этапе их использование показывает эффективность применительно к задачам криптоанализа. Ежегодно проводится соревнование таких решателей. В данной работе для проведения криптоанализа и поиска прообразов функций сжатия выбранных криптографических хэш-функций использовался SAT-решатель kissat [4], в котором реализован алгоритм CDCL (Conflict-Driven Clause Learning) [5], в свою очередь основанный на DPLL [6].

Алгоритм поиска прообраза криптографической хэш-функции с помощью программы CBMC (средство для проверки наличия заданных свойств программы на языке C) [7] и SAT-решателя kissat выглядит следующим образом:

1. Написать алгоритм криптографической хэш-функции (обычно упрощённый) на C-подобном языке и подать на вход программе CBMC. Для анализа стойкости можно использовать только функцию сжатия выбранной хэш-функции. На выходе программа выдаст КНФ.

2. Проверить корректность полученной КНФ, подставив пару заранее известных значений «вход-выход». Если КНФ корректна, SAT-решатель выдаст ответ «SATISFIABLE». Если некорректна, то будет получен ответ «UNSAT».

3. Подать полученную корректную КНФ в SAT-решатель, предварительно написав условия для поиска прообразов. В таком случае в CBMC-кодировке означивается выход, при этом вход остаётся неизвестен. Если КНФ решаема, то решатель выдаст значения необходимых переменных. КНФ может и не решаться, тогда необходимо построить другую КНФ, например, уменьшив число раундов исследуемой хэш-функции.

Таким образом, эвристически уменьшая число раундов, можно исследовать криптографическую стойкость хэш-функции на практике.

Краткое описание выбранных криптографических хэш-функций

BLAKE

В работе рассматривается версия BLAKE-256. Данный вариант оперирует 32-битными словами и возвращает хэш длиной 256 бит. Алгоритм состоит из трёх основных этапов:

1. Инициализация начального состояния. Параметры можно найти в [8], раздел 3.1.1.

2. Применение функции сжатия.

3. Режим итерации.

В данной работе проводится анализ криптографической стойкости функции сжатия. Как было доказано, стойкость хэш-функции основывается на сложности обращения используемой в ней функции сжатия [9, 10].

Функция сжатия BLAKE получает на вход следующие параметры:

- 256 бит значения результата $h = (h_0||\ldots||h_7)$;
- 512 бит значения сообщения $m = (m_0||\ldots||m_{15})$;
- 128 бит значения соли $s = (s_0||s_1||s_2||s_3)$;
- 64 бита значения счётчика $t = (t_0||t_1)$.

Начальные значения h_i заполнены на этапе инициализации; результат выполнения функции сжатия находится в полученном векторе h.

Далее функция обращения состоит из инициализации, функции итерации раунда (общее число раундов – 14) и «финализации» (на данном этапе происходит сложение по модулю два некоторых значений). На стадии инициализации формируется массив 4×4 значений, состоящих из h_i, $s_i \oplus u_i$, а также сложения по модулю два значений счётчика с константами. Затем на стадии итерации раунда производится выполнение восьми функций G_i. Эти функции являются «ядром» функции сжатия и подробно описаны в [8], раздел 3.1.2.2. При анализе стойкости основное внимание в работе уделяется исследованию этих функций.

GRØSTL

Grøstl (Groestl) – это итеративная криптографическая хэш-функция, основной особенностью которой является заимствованные у блочного шифра AES структуры перестановок и S-блоки. В работе рассматривается вариант Grøstl-256, возвращающий хэш-значение длины 256 бит. Упрощённый алгоритм хэш-функции:

1. Инициализация начального состояния [11]. Вектор инициализации задаётся как

$$\text{iv}_{256} = \texttt{00 ... 00 01 00}$$

2. Применение функции сжатия.

3. Выходное преобразование.

Так как в работе анализируется 256-битная версия хэш-функции, то длина блока $\ell = 512$, длина сообщения задаётся параметром N. Функция сжатия состоит из двух перестановок P и Q: $f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h$, где h – chaining function, m – сообщение. В конце используется функция преобразования, возвращающая последние 256 бит значения $(P(x) \oplus x)$.

Как уже было описано выше, криптографическая стойкость хэш-функции основывается на стойкости её функции сжатия, поэтому основное внимание при работе с Grøstl уделяется именно этой функции. Функция сжатия задаётся двумя ℓ-битовыми перестановками P и Q, которые в свою очередь имеют раунды:

- `AddRoundConstant`: необходимо сделать P и Q различными и независимыми друг от друга;
- `SubBytes`: здесь идёт обращение к S-блоку AES, параметры и реализация которого уже хорошо изучены;
- `ShiftBytes` или `ShiftBytesWide`: так же добавляет различия между результатами P и Q, чтобы было сложнее найти какие-либо зависимости, а также для обеспечения лавинного эффекта;
- `MixBytes`: основано на коде, исправляющем ошибки, со свойством MDS.

Основное внимание уделяется раунду `SubBytes`, который можно разделить на 10 подраундов RND512P и 10 подраундов RND512Q, которые можно разделить ещё на 16 подраундов COLUMN, по 8 на каждую перестановку.

Полученные результаты

Для рассмотренных во втором разделе хэш-функций были выбраны открытые реализации на языке C, выделены их функции сжатия. Затем для получения КНФ было использовано средство проверки ограниченной модели CBMC, а для решения полученных булевых уравнений – SAT-решатель kissat.

Вычислительная платформа

При проведении экспериментов использовались следующие аппаратные и программные средства:

- Тестирование проводилось на 12-ядерном процессоре AMD Ryzen 3900X;
- Версия CBMC – 5.89;
- Версия kissat – 3.0;
- Для каждой КНФ запускался kissat на 1 ядре из 12.

BLAKE

Для данной криптографической хэш-функции за основу была взята официальная реализация Jean-Philippe Aumasson [12]. Для анализа была выделена функция сжатия `blake256_compress`. С помощью добавления в CBMC-кодировку команды `__CPROVER_assert(0,"test");` был сгенерирован .cnf файл.

Далее проводилась проверка корректности полученной конъюнктивной нормальной формы. Были проверены два варианта работы: при полностью нулевом и полностью единичном входе. Для этого был посчитан выход функции сжатия в C-программе, а затем дано указание CBMC, что в КНФ эти переменные инициализированы именно так. Проверку корректности программа прошла успешно.

Далее был проведён анализ обратимости BLAKE. Для этого было снижено число раундов, использующихся в функции сжатия. В работе [13] был получен результат по обращению одного из четырнадцати раундов BLAKE. Для одного раунда для известного выхода хэш-функцию удалось обратить за 0.02 секунды. Для двух раундов хэш-функцию за разумное время обратить не удалось, поэтому было принято решение разбить второй раунд на несколько последовательных функций и проанализировать возможность обращения с помощью SAT-решателя первого раунда и части второго раунда.

В одном раунде восемь раз выполняется функция G с заранее установленными параметрами. В каждой функции G, в свою очередь, присутствует восемь последовательных действий. Таким образом, можно условно принять, что в одном раунде производятся 64 действия. Следует обратить внимание, что сами действия не являются равноценными по сложности. В данной работе не анализировалась сложность каждого действия. В таблице 9 показано время нахождения прообраза для заданного числа раундов для нулевого хэша, в таблице 10 – для единичного хэша. Запись «n m/64» означает, что при анализе было запущено n полных раундов и m из 64 действий из второго раунда (то есть одно действие выполнения функции G).

По полученным результатам можно сделать несколько важных и интересных выводов. Во-первых, при добавлении одного действия из функции G, а именно при добавлении

```
v[a] += (m[sigma[i][e]] ^ u256[sigma[i][e1]]) + v[b];
```

время решения кратно возрастает. При этом, добавляя следующие этапы, итоговое время оказывается меньшим. На первый взгляд, это контринтуитивно, однако, такое поведение SAT-решателя нередко встречается, например, оно описывается в работе [14]. В процессе анализа было установлено, что при выполнении действия, где происходит смешивание части внутреннего состояния регистра с входом функции сжатия, сложность обращения кратно возрастает по сравнению со

Число раундов	Время нахождения прообраза на одном ядре, с.
1 8/64	0.51
1 9/64	21 596
1 10/64	8 798
1 11/64	20 004
1 12/64	11 654
1 13/64	не решено
1 14/64	не решено
1 15/64	не решено
1 16/64	не решено

Таблица 9: Обращение нулевого хэша для BLAKE-256 (256 нулевых бит).

Число раундов	Время нахождения прообраза на одном ядре, с.
1 8/64	0.29
1 9/64	2 133
1 10/64	27 089
1 11/64	2 281
1 12/64	не решено
1 13/64	не решено
1 14/64	не решено
1 15/64	не решено
1 16/64	не решено

Таблица 10: Обращение единичного хэша для BLAKE-256 (256 единичных бит).

сложностью обращения действий, где часть внутреннего состояния регистра смешивается с другими частями этого же регистра.

По сравнению с работой [13] удалось продвинуться вперёд на $3/16$ раунда при обращении функции сжатия (обратить 1 полный раунд и дополнительно $3/16$ раунда). В дальнейшем необходимо попробовать оптимизировать некоторые операции для упрощения задачи.

GRØSTL

Для криптографической хэш-функции Grøstl за основу была взята реализация из открытого репозитория криптовалюты Monero [15].

Для того, чтобы доказать ненадёжность хэш-функции, необходимо решить задачу поиска прообраза функции сжатия. Поэтому для работы были выделены перестановки P и Q. Так как каждая из перестановок сама по себе состоит из 10 раундов и ещё 8 подраундов, необходимо было их сократить, чтобы найти решение КНФ за приемлемое время.

Было принято решение оставить первую и последнюю операции в RND512P и RND512Q, а также варьировать количество функций COLUMN в каждом из них. За один раунд можно считать 16 выполнений COLUMN, по 8 в P и в Q. В таблицах 11 и 12 показано время обращения нулевого и единичного хэшей соответственно в зависимости от числа операций в раундах. Обозначение «xPyQ» означает, что было выполнено x вызовов функции COLUMN в перестановке P и y вызовов функции COLUMN в перестановке Q.

На выходе получилась 512-битная строка, поэтому подобно способу, использованному в статье

[13], последние 256 бит просто отбрасываются, а результатом хэш-функции считаются первые 256 бит.

Часть от раунда	Число подраундов перестановок	Время решения КНФ на одном ядре, с
10/16	4P6Q	94.14
10/16	6P4Q	2.13
12/16	8P4Q	3
10/16	5P5Q	37 641
13/16	8P5Q	27 077

Таблица 11: Обращение нулевого хэша для Grøstl-256 (256 нулевых бит).

Часть от раунда	Число подраундов перестановок	Время решения КНФ на одном ядре, с
10/16	4P6Q	50.72
10/16	6P4Q	2.20
12/16	8P4Q	2.88
10/16	5P5Q	21 046
13/16	8P5Q	11 780

Таблица 12: Обращение единичного хэша для Grøstl-256 (256 единичных бит).

Как было понятно из описания, перестановки P и Q не одинаковы. При изменении количества выполнений функции COLUMN в P или в Q можно заметить, что перестановка Q даёт больший прирост сложности при увеличении количества COLUMN, чем перестановка P. Интересно также то, что при увеличении количества подраундов перестановки P, время поиска решения для КНФ уменьшается.

Удалось обратить 0.8125 раунда, что эквивалентно 8 вызовам функции COLUMN в перестановке P и 5 вызовам функции COLUMN в перестановке Q; в работе [13] было обращено примерно 0.5 раунда, что было эквивалентно только перестановке P, так как рассматривалось только выходное преобразование, а не непосредственно функция сжатия.

Заключение

В работе был проведён практический криптоанализ криптографических хэш-функций BLAKE-256 и GRØSTL-256. С помощью современного SAT-решателя, а также программного средства для сведения задач к SAT, удалось решить более сложные (в сравнении с исследованием [13]) задачи поиска прообразов для обеих рассмотренных криптографических хэш-функций. Для хэш-функции BLAKE удалось обратить 1 полный раунд и дополнительно $3/16$ раунда, для GRØSTL – $13/16$ раунда.

В качестве дальнейших исследований планируется провести SAT-криптоанализ для остальных хэш-функций – финалистов конкурса NIST, а также попробовать оптимизировать решатель к задачам криптоанализа.

ЛИТЕРАТУРА

[1] Богачкова И. А., Заикин О. С., Кочемазов С. Е., Отпущенников И. В., Семенов А. А. Применение алгоритмов решения проблемы булевой выполнимости к криптоанализу хэш-функций семейства MD // Прикладная дискретная математика. Приложение. 2015. №. 8. С. 139-142.

[2] Impagliazzo R., Levin L. A., Luby M. Pseudo-random generation from one-way functions // Proceedings of the twenty-first annual ACM symposium on Theory of computing. 1989. pp. 12-24.

[3] Stephen A. Cook. The Complexity of Theorem-Proving Procedures // Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. 1971. pp. 151—158.

[4] Biere A., Fleury M. Gimsatul, IsaSAT and Kissat entering the SAT Competition 2022 // Proceedings of SAT Competition. 2022. pp. 10-11.

[5] Marques-Silva J. P., Sakallah K. A. GRASP — a new search algorithm for satisfiability // Proceedings of International Conference on Computer Aided Design. IEEE, 1996. pp. 220–227.

[6] Davis M., Logemann G., Loveland D. A machine program for theorem-proving // Communications of the ACM. – 1962. Vol. 5. No. 7. pp. 394–397.

[7] Clarke E. M. , Kroening D., Lerda F. A Tool for Checking ANSI-C Programs // In Proceedings of TACAS. 2004. pp. 168-176.

[8] Aumasson J. P. et al. The hash function BLAKE. 2014.

[9] Merkle R. C. A certified digital signature // Proceedings of Conference on the Theory and Application of Cryptology. 1989. pp. 218–238.

[10] Damgård I. B. A design principle for hash functions // Proceedings of Conference on the Theory and Application of Cryptology. 1989. С. 416–427.

[11] Gauravaram P. et al. Grøstl-a SHA-3 candidate // Dagstuhl Seminar Proceedings. – Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[12] BLAKE [Электронный ресурс] // Github. URL : `https://github.com/veorq/BLAKE.git` (дата обращения: 14.08.23)

[13] Homsirikamol E. et al. Security margin evaluation of SHA-3 contest finalists through SAT-based attacks // Proceedings of the 11th IFIP TC 8 International Conference. 2012. pp. 56–67.

[14] Zaikin O. Inverting 43-step MD4 via cube-and-conquer // Proceedings of IJCAI-ECAI. 2022. pp. 1894–1900.

[15] GRØSTL [Электронный ресурс] // Github. URL : `https://github.com/monero-project/monero` (дата обращения: 14.08.23)

Куратор исследования– к.т.н., ведущий научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН, Олег Сергеевич Заикин.

ПОСТКВАНТОВЫЕ КРИПТОСИСТЕМЫ

Уменьшение времени работы постквантовых криптосистем, основанных на решётках

Д. А. Котов[1], А. Д. Рудзянский[2], А. В. Куценко[3], А. О. Бахарев[3]
[1] Национальный исследовательский ядерный университет «МИФИ»
[2] Национальный исследовательский университет "ВШЭ"
[3]Новосибирский государственный университет
E-mail: danielkotof@yandex.ru, adrudzyanskiy@edu.hse.ru, alexandrkutsenko@bk.ru, a.bakharev@g.nsu.ru

Аннотация

Ускорение алгоритмов поствантовой криптографии является актуальной задачей. В данной работе мы анализируем возможность ускорения произведения многочленов и улучшаем быстродействие операций, используемых в решётчатых криптосистемах. Практической частью проекта является оптимизация реализации криптосистемы Крыжовник, одного из участников проекта стандартизации постквантовой цифровой подписи РФ.

Ключевые слова: *криптография на решётках, постквантовая криптография, цифровая подпись.*

Криптография на решётках

Криптографические примитивы на решётках – одно из самых перспективных направлений постквантовой криптографии не только ввиду стойкости этих примитивов к атакам на квантовом и классическом компьютере, но и вследствие большого спектра конструкций (гомоморфное шифрование, электронные голосования, различные типы подписей). Криптографические конструкции на решетках не только относительно просты в теории, но и значимы на практике, и поэтому в скором будущем будут стандартизированы.

Основным объектом рассмотрения является цифровая подпись, основанная на алегбраических решётках, “Крыжовник” [1]. Построение данной цифровой подписи и сопровождающие описание определения представлены в следующем разделе.

Цифровая подпись “Крыжовник”

Основные обозначения и определения

Будем означать $\mathbb{Z}/q\mathbb{Z}$ - кольцо целых по четному модулю q, результат $z \mod q$ представляем в интервале $[0, q-1]$. Далее обозначим за R, R_q, R_p кольца многочленов $\mathbb{Z}[x]/(x^n+1)$, $\mathbb{Z}/q\mathbb{Z}[x]/(x^n+1)$, $\mathbb{Z}/p\mathbb{Z}[x]/(x^n+1)$. Векторы будем обозначать строчными жирными буквами (например, **x**), матрицы – прописными (например, **A**), константы – обычными строчными. Обозначаем за **I** единичную матрицу. Элементы кольца $\mathbb{Z}[x]/(x^n+1)$ будем понимать как векторы-коэффициенты многочленов. Векторы по умолчанию являются векторами-столбцами. Евклидова (или ℓ_2) норма вектора **x** определяется как $||\mathbf{x}|| = ||\mathbf{x}||_2 = \sqrt{\sum_i x_i^2}$, а ℓ_∞-норма как $||\mathbf{x}||_\infty = max_i|x_i|$

Многочленам из кольца R мы ставим в соответствие векторы-коэффициенты длины n, поэтому произведение векторов $\mathbf{x} \cdot \mathbf{y}$ надо понимать как произведение соответствующих многочленов.

Элементу $\mathbf{a} \in R_q$ ставим в соответствие матрицу $rot(\mathbf{a}) \in \mathbb{Z}/q\mathbb{Z}^{n\times n}$, i-ая строка которой - коэффициенты многочлена $x^{i-1} \cdot \mathbf{a}$. Такая матрица задает произведение любого элемента из R_q на многочлен $\mathbf{a}$.

Для конечного множества S запись $s \leftarrow S$ обозначает, что s выбрано в соответствии со случайным равномерным распределением на S. За S_β^ℓ обозначим множество векторов длины ℓ, каждый элемент которого взят в соответствии с равномерным распределением из множества $[-\beta, \beta]$.

Для любого $x \in \mathbb{Q}$ запись $\mathsf{Round}(x) \in \mathbb{Z}$ означает взятие ближайшего целого, где $1/2$ округляется до 1. Для целого $x \in \mathbb{Z}/q\mathbb{Z}$, в бинарном представлении которого $\log q$ бит, функция $\mathsf{MSB}(x, d)$ (соответственно, $\mathsf{LSB}(x, d)$) означает взятие d наибольших (соответственно, наименьших) бит. Все операции распространяются на векторы и матрицы по-коэффициентно.

В схеме цифровой подписи "Крыжовник" будут использоваться 2 модуля $q = 2^\nu$ и $p = 2^\mu$, $\nu > \mu$. "Конвертирование" элемента $x \in \mathbb{Z}/q\mathbb{Z}$ в $x' \in \mathbb{Z}/q\mathbb{Z}$ происходит по правилу $x' = \mathsf{Round}(x \cdot \frac{p}{q})$.

Для всякого целого $\omega > 0$ положим $B_\omega = \{\mathbf{x} \in \mathbb{R}|,\ ||\mathbf{x}||_\infty = 1, ||\mathbf{x}|| = \sqrt{\omega}\} \subseteq R$.

Определение 1. Цифровая подпись - примитив, состоящий из трех алгоритмов:

- вероятностный алгоритм генерации ключевой пары $\mathsf{KeyGen(par)}$, возвращающей секретный ключ sk и ключ верефикации vk;
- вероятностный алгоритм генерации подписи $\mathsf{Sign(sk}, m)$, который для сообщения $m \in M$ возвращает подпись σ;
- вероятностный алгоритм $\mathsf{Verify(m}, \sigma, \mathsf{vk})$, который возвращает либо "Accept"(подпись σ корректна для (m, vk)), либо "Reject" (подпись σ не корректна для (m, vk))

Цифровая подпись корректна с долей ошибки ϵ, если для всех пар $(\mathsf{sk}, \mathsf{vk}) \in \mathsf{Keygen(par)}$ и всех сообщений $m \in M$ имеем $Pr[\mathsf{Verify}(m, \mathsf{Sign}(\mathsf{sk}, m), \mathsf{vk}) = \text{"Accept"}] \geq 1 - \epsilon$

Сложные задачи на решётках

Безопасность подписи основывается на двух "сложных в среднем" задачах. Первая - задача Обучения с Округлением (от англ. **Learning with Rounding, LWR**) - детерминированная версия задачи Обучения с Ошибками (**Learning with Errors**). В основе подписи ключей подписи лежит трудность этой модульной версии задачи над факторкольцом R_q. Все вычисления производятся в факторкольце R_q, матрица $\mathbf{A}$ формируется как блочная матрица из $k \cdot \ell$ элементов из R_q.

Определение 2. *Задача Обучения с Округлением (MLWR)*

Пусть $q \geq p \geq 1,\ k, \ell \geq 1$ - целые числа. MLWR-распределение для вектора $\mathbf{s} \leftarrow \mathbb{R}_q^\ell$ есть множество пар вида($\mathbf{A}, Round(\frac{p}{q} \cdot \mathbf{A}) \cdot \mathbf{s}$), где $\tilde{\mathbf{A}} \leftarrow \mathbb{R}_q^{k\cdot\ell}$.

Задача поиска: Для заданного произвольным образом числа выборок из MLWR-распределения для вектора $\mathbf{s} \leftarrow \mathbb{R}_q^\ell$ восстановить $\mathbf{s}$.

Задача различения распределений: Для заданного произвольным образом болъшого числа въборок из $\widetilde{R}_q^{k\times\ell} \times R_p^k$ определить, являются ли они равномерно распределенными или же $MLWR$ распределенными для вектора $\mathbf{s} \leftarrow R_q^\ell$.

Вероятность успеха алгоритма, решающего задачу MLWR с введёнными выше параметрами, будем обозначать $\mathrm{Adv}_{k,\ell,p,q}^{\mathrm{MLWR}}$.

Определение 3. *Задача нахождения Короткого Целочисленного Решения (MSIS)*

Зафиксируем $b \in \mathbb{N}$ и пусть $\mathbf{A} \leftarrow R_q^{k\times\ell}$. Модульная задача нахождения короткого целочисленного решения, параметризованная посредством $b > 0$, заключается в нахождении "короткого"ненулевого прообраза $\mathbf{y} \leftarrow R_q^{k+\ell}$ в решетке, определяемой $\mathbf{A}$, т.е.

$$\mathbf{y} \neq 0, \quad [\mathbb{I} \mid \mathbf{A}] \cdot \mathbf{y} = 0 \quad u \quad \|\mathbf{y}\|_\infty \leq b. \tag{8}$$

Вероятность успеха алгоритма, решающего задачу MSIS с введёнными выше параметрами, будем обозначать $\mathrm{Adv}^{\mathrm{MSIS}}_{k,\ell,q,b}$.

Цифровая подпись зависит от следующих параметров: $q = 2^\nu,\ p = 2^\mu\ \nu > \mu$. Будет использоваться криптографическая хэш-функция $\mathbf{H}\ : \{0,1\}^*\ \rightarrow \mathbf{B}_\omega$. Для стойкости к квантовым атакам должно выполняться $|B_\omega|\ \geq\ 2^{256}$, что достигается при $\omega\ \geq\ 60$. Параметры k, l отвечают за размерности ключей. Параметры s, γ задают интервалы для коэффициентов многочленов в процессе генерации ключей или подписи, параметры d, β отвечают за корректность и безопасность схемы. Подпись формируется для сообщений $M\ \in\ \{0,1\}^*$

Ниже представлены три алгоритма:

Алгоритм 1. Генерация ключей

Вход: $k > \ell > 1, q > p, s$

Выход: $\mathbf{A}, \mathbf{t}$

1: $\mathbf{A} \leftarrow R_q^{k\times\ell}$
2: $\mathbf{s} \leftarrow S_s^\ell$
3: $\mathbf{t} = \mathsf{Round}\left(\frac{p}{q} \cdot \mathbf{As}\right)$
4: `вернуть` sk = **s**, vk = (**A, t**)

Алгоритм 2. Генерация подписи

Вход: $q = 2^\nu, p = 2^\mu, \ell > 1, M, \mathbf{A}, \mathbf{t}, \mathbf{s}, d, \mathcal{H}, \beta, \gamma, \omega$

Выход: (**z**, **c**)

1: $\mathbf{y} \leftarrow S_{\gamma-1}^\ell$
2: $\mathbf{c} = \mathcal{H}(\mathsf{MSB}(\mathbf{A} \cdot \mathbf{y}, d), M)$
3: $\mathbf{z} = \mathbf{y} + \mathbf{sc}$
4: $\mathbf{w} = \mathbf{Az} - \mathbf{t} \cdot 2^{\nu-\mu} \cdot \mathbf{c}$
5: `если` $(\|\mathsf{LSB}(\pm\mathbf{w}, \nu - d)\|_\infty) \geq 2^{\nu-d} - \omega \cdot 2^{\nu-\mu+1})$ или $(\|\mathbf{z}\|_\infty \geq \gamma - \beta)$, `то`
6: `перезапуск`
7: `вернуть` (**z**, **c**)

Алгоритм 3. Проверка подписи

Вход: $M, \mathbf{z}, \mathbf{c}, \mathbf{A}, \mathbf{t}, d, \mathcal{H}, \beta, \gamma$

Выход: Принять или отклонить

1: $w = \mathbf{Az} - \mathbf{t} \cdot 2^{\nu-\mu} \cdot \mathbf{c}$
2: $\mathbf{c}' = \mathcal{H}(\mathsf{MSB}(\mathbf{w}, d)), M)$
3: `если` $\mathbf{c}' == \mathbf{c}$ `и` $||z||_{\inf} \leq \gamma - \beta$ `тогда`
4: `вернуть` “Принять"
5: `иначе`
6: `вернуть` “Отклонить"

Методы оптимизации

В “Крыжовнике” самыми трудоёмкими операциями являются умножения $\mathbf{As}$ в алгоритме генерации ключей, $\mathbf{Az}$ в алгоритмах генерации и проверки подписи. В нашем проекте мы работали над оптимизацией умножения многочленов матрицы $\mathbf{A}$ на многочлены вектора $\mathbf{s}$. Для ускорения в [2] был предложен вариант использовать NTT (от англ. Number Theoretic Transform) - частный случай FFT (от англ. Fast Fourier Transform) для конечных полей $\mathbb{Z}_q$. Для многочленов степени n NTT можно применять, когда n - степень двойки, а $q \equiv 1 \pmod{2n}$. В большей части LWR и LWE криптосистем n обычно выбирает степенью двойки, так что q должно быть выбрано так, чтобы удовлетворять критерию выше. Однако $\mathbb{R}_q$ не является кольцом, в котором данное преобразование можно применять, так как q само является степенью двойки.

Условие $q \equiv 1 \pmod{2n}$ для NTT необходимо для существования первообразного корня в кольце:

Определение 4. Пусть $\mathbb{R}$ - кольцо, $\alpha \in \mathbb{R}$ - элемент кольца. α - *первообразный корень*, если: $\alpha^n = 1$ и $\sum_{j=0}^{n-1} \alpha^{jk} = 0$ для $1 \leq k < n$.

Основная идея оптимизации умножения многочленов заключается в том, чтобы набрать несколько простых чисел p_i, для которых произведение $P = \prod_{i=0}^{k} p_i > q$ гораздо больше q (достаточно $> n \cdot q^2/2$). p_i имеют вид $2^k p' + 1$, где p' — другое число. Это так называемые “NTT-friendly prime”. Тогда для в каждом из p_i мы делаем следующие действия:
1) $NTT^{-1}(NTT(\mathbf{A}) \cdot NTT(\mathbf{s}))$ для каждого p_i
2) собрать результаты вместе с помощью Китайской теоремы об остатках
После выполнения Китайской теоремы об остатках корректно выполненные вычисления по модулю P вернет корректно произведение многочленов по модулю q.

Проверка сложности Следует упомянуть, что сравнение сложности операций, над которыми проводилась оптимизация, уже было проведено в [3].

Заключение

В рамках данной работы была изучена цифровая подпись “Крыжовник”, являющаяся кандидатом на стандарт постквантовой криптосистемы в Российской Федерации. Была проанализирована операция умножения многочленов, и был найден более быстрый алгоритм реализации данной операции. Была запрограммирована ускоренная версия умножения полиномов.

ЛИТЕРАТУРА

[1] Е. А. Киршанова, Н. С. Колесников, Е. С. Малыгина, С. А. Новоселов, “Проект стандартизации постквантовой цифровой подписи”, ПДМ. Приложение, 2020, № 13, 44–51. SibeCrypt’20

[2] Chung, Chi-Ming Marvin, et al. "NTT multiplication for NTT-unfriendly rings: New speed records for Saber and NTRU on Cortex-M4 and AVX2."IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 159-188.

[3] Mera, Jose Maria Bermudo, Angshuman Karmakar, and Ingrid Verbauwhede. "Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography."Cryptology ePrint Archive (2020).

Кураторы исследования –

к.ф.-м.н., старший преподаватель кафедры теоретической кибернетики ММФ НГУ, Куценко Александр Владимирович;

инженер Математического центра в Академгородке, ассистент кафедры теоретической кибернетики ММФ НГУ, студент ММФ НГУ, Бахарев Александр Олегович.

Разработка эвристических методов криптоанализа постквантовых криптосистем

И. Д. Иогансон[1], А. Г. Леевик[1], К. И. Зименко[2], А. В. Куценко[3], А. О. Бахарев[3]
[1]Национальный исследовательский университет ИТМО
[2]Институт компьютерных технологий и информационной безопасности ИТА ЮФУ
[3]Новосибирский государственный университет
E-mail: ivan.ioganson@yandex.ru, anton.leevik@gmail.com, kirillzimenko@mail.ru, alexandrkutsenko@bk.ru, a.bakharev@g.nsu.ru

Аннотация

Задача нахождения кратчайшего вектора является одной из основных вычислительных задач в теории решёток, так как стойкость большинства постквантовых криптосистем на решетках зависит от эффективности решения данной задачи. Исходя из этого разработка и анализ эвристических методов криптоанализа постквантовых криптосистем являются важными направлениями исследований в современной криптографии. В рамках проекта были изучены известные работы, посвящённые улучшению методов перечисления решеток с дискретным отсечением. Авторами был проанализирован алгоритм, представленный в работе (Luan L. et al., 2023), и изучена возможность его ускорения.

Ключевые слова: *криптография на решётках, алгоритм перечисления, дискретное отсечение, SVP.*

Введение

В последние десятилетия развитие квантовых вычислений стало активным направлением исследований в области информационных технологий. Однако вместе с потенциальными преимуществами квантовые компьютеры также представляют значительную угрозу для современных криптографических систем, основанных на классических вычислениях. Используемые в настоящее время стандарты асимметричного шифрования, основанные на задачах факторизации и дискретного логарифмирования, могут быть взломаны за полиномиальное время с помощью квантовых алгоритмов, таких как алгоритм Шора [1].

Для обеспечения безопасности информации в эпоху квантовых вычислений, были предложены постквантовые криптосистемы, основанные на математических задачах, для которых пока не существует полиномиальных квантовых и классических алгоритмов решения. Одним из известных и активно исследуемых направлений постквантовой криптографии является теория решеток. В 2022 году Национальный институт стандартов и технологий США выбрал четыре постквантовые криптосистемы (три схемы электронной подписи и одну схему выработки общего ключа) для стандартизации [2], трое из которых построены на теории решеток.

В теории решеток основной математической задачей, на которой строятся большинство криптосистем, является задача поиска кратчейшего вектора в решетке или SVP (Shortest Vector Problem). Данная задача является NP-трудной [3]. Однако обычно рассматривают приближенный вариант данной задачи, а именно задачу SVP_γ, для решения которой требуется найти вектор, длина которого не превышает на γ длину кратчайшего вектора. Для экспоненциального значения $\gamma = 2^{O(n)}$, где n — ранг решетки, существует полиномиальный алгоритм нахождения кратчайшего вектора, алгоритм LLL [4]. Для взлома существующих криптосистем на решетках требуется решить задачу SVP_γ для $\gamma = poly(n)$, которую LLL с большой вероятностью решить не сможет, поэтому требуется использовать другие алгоритмы поиска кратчайшего вектора.

На данный момент существуют два основных типа алгоритмов, которые могут решить задачу SVP, за экспоненциальное время. Первый — алгоритм просеивания, впервые упомянутый в [8], строится на поиске крайтчайшего вектора, путем вычисления разницы векторов решетки, сгенерированных специальным образом. Временная и пространственная сложности алгоритма просеивания составляют $2^{O(n)}$. Второй тип — алгоритм перечисления, впервые представленный в работах [5, 6, 7]. Суть алгоритма перечисления состоит в нахождении кратчайшего вектора решетки путем перебора коэффициентов в многомерном параллелепипеде. В отличие от алгоритма просеивания пространственная сложность алгоритма перечисления равна $O(n)$, однако его временная сложность равна $2^{n \log n}$. Алгоритм перечисления используется в алгоритме редукции базиса BKZ (block Korkine-Zolotareff) [9], представляющем из себя обобщение алгоритма LLL. Суть алгоритма заключается в том, что в процессе работы BKZ ищется кратчайший вектор в подрешетке, порожденной β ($\beta \leq n$) векторами оригинальной базисной матрицы, с помощью алгоритма перечисления и далее он заносится в базисную матрицу, поданную на вход алгоритма BKZ. Таким образом, на выходе получается редуцированный базис решетки. При параметре $\beta = 2$ итоговый результат будет эквивалентен результату алгоритма LLL. Временная сложность алгоритма BKZ равна $2^{O(\beta \log \beta)}$, а коэффициент аппроксимации γ соответственно равен $\beta^{O(\frac{n}{\beta})}$.

В нашей работе сделан акцент на алгоритмах перечисления, а также предложено улучшение одного из алгоритмов на основе эвристического анализа.

Общие сведения

Определение 1. Пусть $\mathbb{R}^m$ — m-мерное пространство действительных чисел. *Решеткой* называется множество всех комбинаций n линейно-независимых векторов $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ с целыми коэффициентами:

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}.$$

Параметры n и m представляют собой ранг и размерность решетки, где $n \leq m$. Здесь и далее будем представлять вектора, как вектора-столбцы. Линейно независимая система векторов, порождающая решётку, называется *базисом решётки*. Базисные вектора могут быть представлены в виде базисной матрицы $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, тогда решетка может быть представлена следующим образом:

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}.$$

Одним из важных параметров решетки является фундаментальный параллелепипед, который задается по следующей формуле:

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} = (x_0, x_1, \ldots, x_n) \in \mathbb{R}^n, x_i \in [0, 1)\}.$$

Объем фундаментального параллелпипеда равен *определителю* решетки $\det \mathcal{L}(\mathbf{B})$. Определитель решетки является ее инвариантом, отсюда следует, что одна и та же решетка может быть задана разными базисами. Для того, чтобы построить эквивалентный базис требуется базис решетки умножить на целочисленную унимодулярную матрицу (квадратную матрицу, с целыми коэффициентами, определитель которой равен ± 1). То есть выполняется следующее свойство:

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}') \Leftrightarrow \mathbf{B}' = \mathbf{BU},$$

где $\mathbf{U}$ — унимодулярная матрица.

Определение 2. *Минимальным расстоянием* λ_1 называется норма кратчайшего ненулевого вектора в решетке $\mathcal{L}$, то есть

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{x}\|.$$

В случайных решетках существует метод определения примерной длины для λ_1, основанный на Гауссовой эвристике [10].

Эвристика 1. Пусть $\mathcal{L}$ — решетка в $\mathbb{R}^n$, а S — это измеримое множество в $\mathbb{R}^n$, тогда

$$|\mathcal{L} \cap S| \approx \mathrm{vol}(S)/\det(\mathcal{L}).$$

Если эвристика соблюдается, то будем предполагать, что значение $\lambda_1(\mathcal{L})$ будет близко к

$$\mathrm{GH}(\mathcal{L}) = \frac{\det(\mathcal{L})^{\frac{1}{n}}}{\mathrm{vol}(\mathrm{Ball}_n(1))^{\frac{1}{n}}},$$

где $\mathrm{vol}(\mathrm{Ball}_n(1))$ — гиперобъем n-мерного шара единичного радиуса.

Данная эвристике может не соблюдаться на ряде решеток, однако для случайных решеток значение λ_1 близко к значению $\mathrm{GH}(\mathcal{L})$ [11].

Определение 3. *Задача поиска кратчайшего вектора (Shortest Vector Problem, SVP).* Пусть дан базис $\mathbf{B}$, который задает решетку $\mathcal{L}$, требуется найти ненулевой вектор $\mathbf{v} \in \mathcal{L}$ такой, что $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

Алгоритм 1. Ортогонализации Грама-Шмидта.

Вход: базис $\mathbf{B} \in \mathbb{Z}^{m \times n}$.

Выход: ортогонализованный базис $\mathbf{B}^* \in \mathbb{Q}^{m \times n}$.

1. $\mathbf{b}_1^* = \mathbf{b}_1$.
2. $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$, где $\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$ для всех $i = 2, \ldots, n$.

Определение 4. *Ортогональная проекция.* Пусть $\pi_i : \mathbb{R}^m \to span(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})^{\perp}$ - i-ая проекция на ортогональную плоскость к заданной $i-1$ базисными векторами. Для вектора $\mathbf{v} \in \mathbb{R}^m$ зададим его i-ую проекцию как $\pi_i(\mathbf{v}) = \mathbf{v} - \sum_{j=1}^{i-1} \frac{\langle \mathbf{v}, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \mathbf{b}_j^*$. Так как каждый вектор решетки можно представить через ортогональный базис $\mathbf{B}^*$, то есть $\mathbf{v} = \sum_{i=1}^{n} u_i \mathbf{b}_i^*$, где $u_i \in \mathbb{Q}$, то проекцию можно также представить как $\pi_i(\mathbf{v}) = \sum_{j=i}^{n} u_j \mathbf{b}_j^*$.

Алгоритмы перечисления

Здесь и далее будем рассматривать только полноранговые решетки, то есть где $m = n$. Алгоритм перечисления, впервые упомянутый в работах [5, 6, 7], получает на вход базис $\mathbf{B}$ и радиус R и формирует множество точек решетки $\mathcal{L}$, лежащих внутри n-мерного шара радиуса R, перебрав которое можно найти кратчайший вектор решетки.

Ключевая идея состоит в рекурсивном поиске, используя проекции, которые позволяют уменьшить размерность решетки, так как если $\|\mathbf{v}\| \leq R$, то тогда и $\|\pi_k(\mathbf{v})\| \leq R$ для всех $1 \leq k \leq n$. Таким образом, перебор точек решетки начинается со спроецированной решетки $\pi_n(\mathcal{L})$ размерности 1 и последовательно переходит к решеткам большей размерности $\pi_{n-1}(\mathcal{L})$, ..., $\pi_2(\mathcal{L})$ и соответственно к самой решетке $\mathcal{L}$. Например, пусть мы перебрали все вектора во множестве $\pi_{k+1}(\mathcal{L}) \cap S$, где

$S = \mathrm{Ball}_n(R)$, тогда переходя к решетке $\pi_k(\mathcal{L})$ мы фиксируем ранее найденные точки и перебираем далее относительно зафиксированной точки. Иными словами мы строим дерево перечисления, где вершинами являются координаты точек в проективных решетках, а поиск кратчайшего вектора представляет из себя поиск в глубину по этому дереву.

Новым шагом развития алгоритмов перечисления стали работы Шнорра [12, 13] об использовании отсечения части области перечисления. Формально можно сказать, что алгоритм перечисления с отсечением принимает на вход дополнительно множество $P \subseteq \mathbb{R}^n$ и формирует новое множество точек $\mathcal{L} \cap S \cap P$, перебирая которое, алгоритм найдет кратчайший вектор. Пусть $f : \{1, \ldots, n\} \to [0, 1]$, тогда множество P может задать в общем случае по следующей формуле [14]:

$$P = \{\mathbf{x} \in \mathbb{R}^n : \|\pi_{n+i-1}(\mathbf{x})\| \leq f(i) \cdot R, \forall i \in \{1, \ldots, n\}\}.$$

Преимущество данной модификации алгоритма перечисления заключается в том, что при удачно выбранном множестве P перебор значительно сокращается. Стоит отметить, что при неправильно выбранном P либо ни один вектор решетки не попадет в пересечение множеств $\mathcal{L} \cap S \cap P$, либо в данном пересечении множеств не найдётся вектор нужной длины. Таким образом, данная версия алгоритма является вероятностной, когда предыдущая версия была детерминированной.

Применение отсечения области S значительно ускорило алгоритм перечисления, однако вопрос нахождения других способов задания областей отсечения, которые дадут лучшие результаты по сравнению с предыдущими до сих пор остается открытым.

В 2017 году на конференции Eurocrypt2017 была представлена новая версия алгоритма перечисления, представляющая новое дискретное отсечение [15]. Концепция дискретного отсечения основывается на разделении пространства решетки на непересекающиеся сектора $\mathcal{C}(\mathbf{t})$, задаваемые тегами $\mathbf{t} \in T$, где T - счетное множество. Внутри каждого сектора $\mathcal{C}(\mathbf{t})$ лежит только одна точка решетки, которая может быть получена по тегу за полиномиальное время. Таким образом, вместо перечисления координат точек решетки, производится перечисление секторов пространства.

В качестве тривиального примера можно привести следующий вариант разделения пространства решетки на следующие сектора $\mathcal{C}(\mathbf{t})$, где $T = \mathbb{Z}^n$:

$$\mathcal{C}(\mathbf{t}) = \mathbf{B}\mathbf{t} + \mathcal{D},$$

$$\mathcal{D} = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^* : x_i \in [-1/2, 1/2) \right\}.$$

Однако данный вариант разделения пространства не несет особой пользы, так как вектор решетки мы знаем из самого тега $\mathbf{t}$, так как $\mathcal{L} \cap \mathcal{C}(\mathbf{t}) = \{\mathbf{B}\mathbf{t}\}$, поэтому перебор секторов сводится к перебору векторов.

В качестве нетривиального примера разделения пространства в [15] приводится разделение Бабая.

Определение 5. *Ортогональное разделение (или разделение Бабая).* Пусть дан базис $\mathbf{B}$ полноранговой решетки $\mathcal{L}$ ранга n и $T = \mathbb{Z}^n$. Тогда разделением Бабая пространства решетки будет называться следующее разделение:

$$\mathcal{C}_{\mathbb{Z}}(\mathbf{t}) = \mathbf{B}^*\mathbf{t} + \mathcal{D},\ \mathcal{D} = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^* : x_i \in [-1/2, 1/2) \right\}.$$

Такое название разделение получило из-за того, что для того чтобы вычислить точку решетки внутри каждого сектора, используется алгоритм Бабая. В работе [16] представлен иной способ разделения пространства решетки, где каждый тег решетки представлялся в виде вектора натуральных чисел, такое разделение получило название как натуральное.

Определение 6. *Натуральное разделение решетки*. Пусть дан базис $\mathbf{B}$ полноранговой решетки $\mathcal{L}$ ранга n, пусть $T = \mathbb{N}^n$, тогда пусть натуральным разделением пространства решетки будет называться следующее разделение:

$$\mathcal{C}_{\mathbb{N}}(\mathbf{t}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^* : x_i \in \left(-\frac{t_i + 1}{2}, -\frac{t_i}{2} \right] \cup \left(\frac{t_i}{2}, \frac{t_i + 1}{2} \right] \right\}.$$

В общем случае для того чтобы построить дискретное разделение решетки, удовлетворяющее свойству, что внутри каждого сектора должна быть представлена только одна точка решетки, требуется построить биективное отображение $f : \mathcal{L} \to T$, то есть каждый вектор решетки можно задать через соответствующий тег разделения. В работе [16] представлено доказательство того, что такое отображение для натурального разделения решетки биективно. Таким образом, вопрос более эффективного разделения решетки сводится к нахождению такого отображения, которое даст лучшие результаты для алгоритма перечисления.

Для сравнения двух видов разделения в [15] анализируются значения математического ожидания секторов разделения, из чего был сделан вывод, что использование натурального разделения в алгоритме перечисления более эффективно, чем использование ортогонального разделения.

Исходя из определения разделения по тегу $\mathbf{t}$ можно определить приблизительное значение вектора внутри сектора $\mathcal{C}(\mathbf{t})$, таким образом, главная идея перечисления с дискретным отсечением состоит в поиске множества секторов $\bigcup_{\mathbf{t} \in T} \mathcal{C}(\mathbf{t})$, в которых "скорее всего"находятся короткие вектора решетки. Для оценки примерной длины вектора, находящегося внутри соответствующего сектора, вычисляется математическое ожидание сектора. После определения множества $\mathcal{U}$, состоящего из всех подходящих тегов $\mathbf{t}$, итеративно для каждого $\mathbf{t} \in \mathcal{U}$ вычисляется $\mathbf{v} \in \mathcal{L} \cap \mathcal{C}(\mathbf{t})$ и, если $\|\mathbf{v}\| \leq R$, то тогда решение найдено. Алгоритм перечисления с дискретным отсечением представлен ниже.

Алгоритм 2. Перечисление с дискретным отсечением.

Вход: редуцированный базис $\mathbf{B}$, количество точек для перебора M, длинна целевого вектора R.

Выход: $\mathbf{v} \in \mathcal{L}$ такой, что $\|\mathbf{v}\| \leq R$.

1. Редуцировать базис B.
2. Пока Истина:
   (a) $S \leftarrow \emptyset$.
   (b) Найти оптимальный радиус r такой, что существуют M тегов, для которых $f(\mathbf{t}) < r$.
   (c) Перебрать все M тегов и сохранить во множество S.
   (d) Для всех $\mathbf{t} \in S$:
      i. Декодировать вектор $\mathbf{v}$ из $\mathbf{t}$.
      ii. Если $\|\mathbf{v}\|^2 \leq R^2$, то вернуть $\mathbf{v}$.
   (e) Изменить базис $\mathbf{B}$, умножив его на случайную унимодулярную матрицу и перемешав вектора в базисе.
   (f) Редуцировать базис $\mathbf{B}$.

В работе [17] предложены улучшения данного алгоритма, в частности на шаге 2.b в работе предложен алгоритм бинарного поиска, дающий более точную оценку на радиус r. Программная

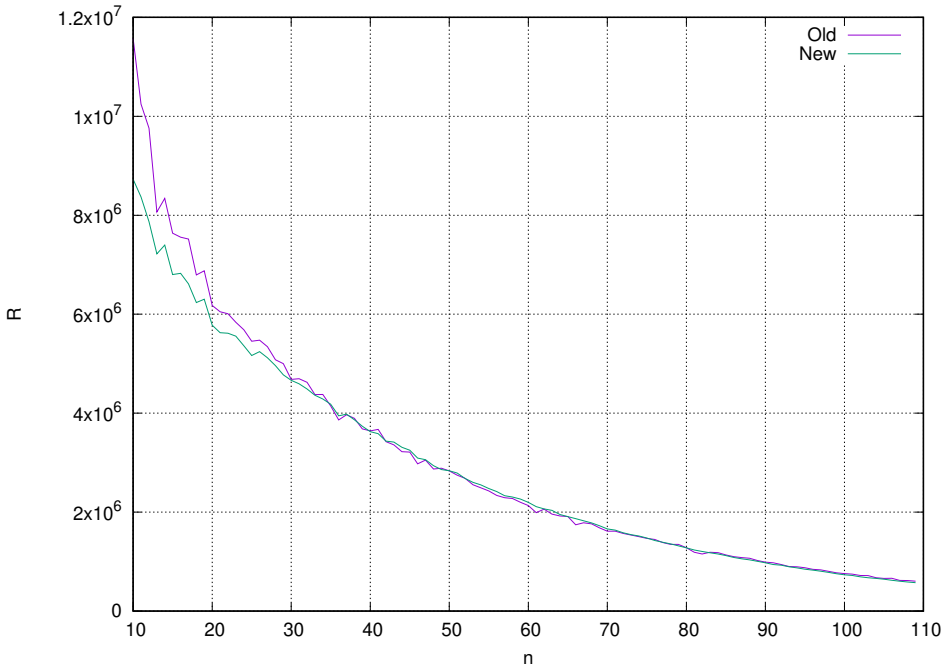


Рис. 1: Старый и новый радиусы при $M = 1000$

реализация данной модификации представлена в открытом репозитории на Github [18]. В процессе тестирования программной реализации алгоритма выяснилось, что $\approx 15\%$ времени каждого раунда тратится на определение оптимального радиуса (шаг 2.b алгоритма 2), поэтому в целях ускорения работы алгоритма был оптимизирован соответствующий шаг.

Предложенное ускорение

Для улучшения быстродействия алгоритма был предложен новый метод для определения оптимального радиуса перебора. Вместо использования бинарного поиска с перебором количества точек было принято решение вывести эвристическую формулу зависимости радиуса от параметров решетки и искомого количества точек. В итоге получилась следующая формула:

$$r = 2.2\left(\frac{\det(\mathbf{B}^*)\cdot M}{\mathrm{vol}(\mathrm{Ball}_n(1))}\right)^{\frac{1}{n}} \lg(M)\cdot 10^{\frac{-n}{75-\left(\frac{4.5}{\lg(M)}\right)^3}},$$

где M — искомое количество точек, n — размерность решетки, $\mathbf{B}^*$ — ортоганализированный базис решетки, $\mathrm{vol}(\mathrm{Ball}_n(1))$ — гиперобъем n-мерного шара единичного радиуса. Формула была получена на основе аппроксимации данных, полученных с помощью оригинального алгоритма.

Используя описанную выше формулу, был проведен ряд экспериментов, чтобы выяснить насколько достоверные результаты выдает данная формула. Были построены графики зависимости радиуса перебора от размерности решетки при нескольких фиксированных значениях искомого количества точек. Результаты экспериментов приведены на рисунках 1–3.

Также был проведен эксперимент, позволяющий выяснить как новое решение сказывается на быстродействии всего алгоритма. Эксперимент заключался в следующем: фиксировались определенные параметры, генерировалась случайная решетка, для этой решетки запускались алгоритмы перечисления со старым и новым способами определения радиуса и замерялось время выполнения алгоритмов. Эксперимент проводился 10 раз и вычислялось среднее арифметическое времени выполнения. Результаты приведены в таблицах 1 и 2.

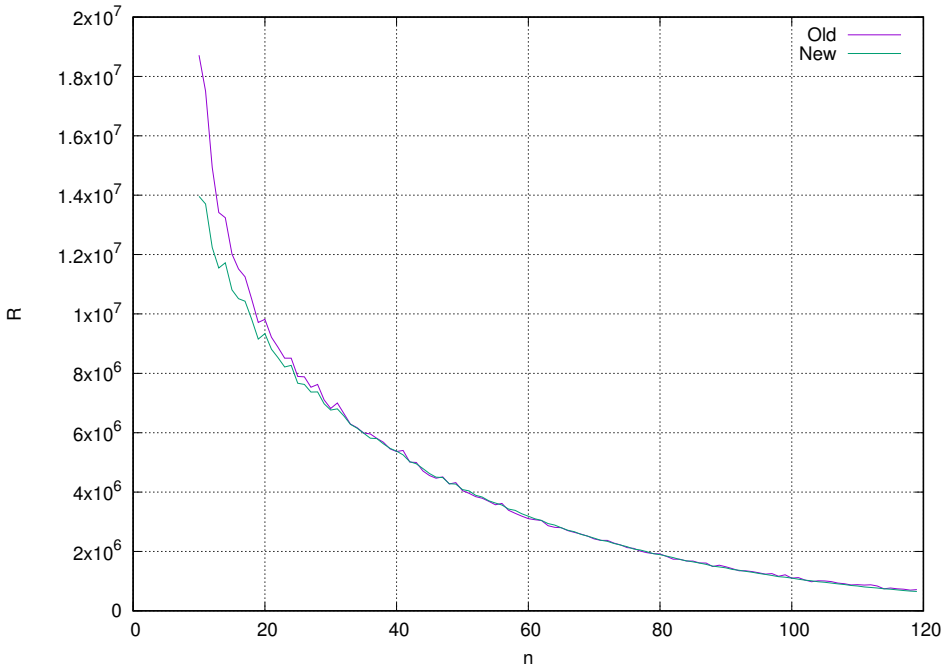


Рис. 2: Старый и новый радиусы при $M = 10000$

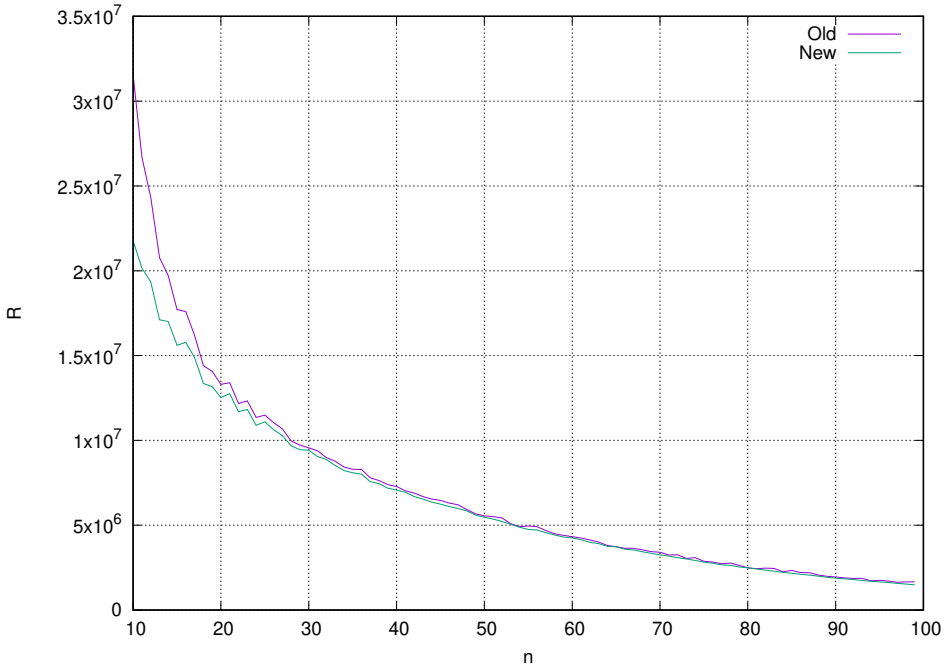


Рис. 3: Старый и новый радиусы при $M = 100000$

$M \backslash n$	60	70	80
1000	1.553	9.084	–
5000	0.758	–	–
10000	1.05	3.666	14.385
30000	–	1.269	–
50000	–	–	13.638
100000	–	–	17.325

Таблица 13: Время работы старого решения, сек.

$M\backslash n$	60	70	80
1000	0.72	18.233	–
5000	0.412	–	–
10000	1.108	1.496	28.68
30000	–	1.887	–
50000	–	–	10.085
100000	–	–	8.61

Таблица 14: Время работы нового решения, сек.

Заключение

В итоге, в ходе это работы была изучена статья [17], из которой был взят и оптимизирован с помощью эвристических методов алгоритм перечисления с дискретным отсечением. Также была получена формула зависимости радиуса от параметров решетки и искомого количества точек.

Полученные в результате экспериментов данные, представленные на рисунках $1 - 3$, показывают, что предложенная эвристическая формула близка к значениям, полученным с помощью эмпирического алгоритма, представленного в оригинальной статье, и может быть использована для получения приблизительного значения радиуса поиска.

Также были проведены эксперименты для сравнения быстродействия оригинального алгоритма и модифицированного. Результаты приведены в таблицах 1 и 2. Можно заметить, что в 5 из 9 поставленных экспериментов модифицированный алгоритм работает быстрее. Таким образом, можно сделать вывод, что разработанное решение действительно позволяет ускорить алгоритм в некоторых случаях.

ЛИТЕРАТУРА

[1] Shor P. W. Algorithms for quantum computation: discrete logarithms and factoring //Proceedings 35th annual symposium on foundations of computer science. – Ieee, 1994. – С. 124–134.

[2] Alagic G., Apon D., Cooper D. et al. Status report on the third round of the NIST post-quantum cryptography standardization process // US Department of Commerce, NIST. – 2022.

[3] Ajtai M. The shortest vector problem in L2 is NP-hard for randomized reductions //Proceedings of the thirtieth annual ACM symposium on Theory of computing. – 1998. – С. 10–19.

[4] Lenstra A. K., Lenstra H. W., Lovász L. Factoring polynomials with rational coefficients //Mathematische annalen. – 1982. – Т. 261. – №. ARTICLE. – С. 515–534.

[5] Pohst M. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications //ACM Sigsam Bulletin. – 1981. – Т. 15. – №. 1. – С. 37-44.

[6] Kannan R. Improved algorithms for integer programming and related lattice problems //Proceedings of the fifteenth annual ACM symposium on Theory of computing. – 1983. – С. 193–206.

[7] Fincke U., Pohst M. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis //Mathematics of computation. – 1985. – Т. 44. – №. 170. – С. 463–471.

[8] Ajtai M., Kumar R., Sivakumar D. A sieve algorithm for the shortest lattice vector problem //Proceedings of the thirty-third annual ACM symposium on Theory of computing. – 2001. – С. 601–610.

[9] Korkine A., Zolotareff G. Sur les formes quadratiques positives //Mathematische Annalen. – 1877. – Т. 11. – №. 2. – С. 242–292.

[10] Gama N., Nguyen P. Q., Regev O. Lattice enumeration using extreme pruning //Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. – Springer Berlin Heidelberg, 2010. – С. 257–278.

[11] Rogers C. A. The number of lattice points in a set //Proceedings of the London Mathematical Society. – 1956. – Т. 3. – №. 2. – С. 305–320.

[12] Schnorr C. P., Euchner M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems //Mathematical programming. – 1994. – Т. 66. – С. 181-199.

[13] Schnorr C. P., Hörner H. H. Attacking the Chor-Rivest cryptosystem by improved lattice reduction //International Conference on the Theory and Applications of Cryptographic Techniques. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1995. – С. 1-12.

[14] Gama N., Nguyen P. Q., Regev O. Lattice enumeration using extreme pruning //Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. – Springer Berlin Heidelberg, 2010. – С. 257-278.

[15] Aono Y., Nguyen P. Q. Random sampling revisited: lattice enumeration with discrete pruning //Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II 36. – Springer International Publishing, 2017. – С. 65-102.

[16] Fukase M., Kashiwabara K. An accelerated algorithm for solving SVP based on statistical analysis //Journal of Information Processing. – 2015. – Т. 23. – №. 1. – С. 67-80.

[17] Luan L. et al. Lattice Enumeration with Discrete Pruning: Improvements, Cost Estimation and Optimal Parameters //Mathematics. – 2023. – Т. 11. – №. 3. – С. 766.

[18] DP-ENUM. Лицензия: GNU GPL 2.0. URL: GitHub, https://github.com/LunaLuan9555/DP-ENUM (дата обращения: 14.08.2023). Режим доступа: открытый.

Кураторы исследования –
к.ф.-м.н., старший преподаватель кафедры теоретической кибернетики ММФ НГУ, Куценко Александр Владимирович;
инженер Математического центра в Академгородке, ассистент кафедры теоретической кибернетики ММФ НГУ, студент ММФ НГУ, Бахарев Александр Олегович.

Приложение квазициклических кодов в криптографии

А. А. Кунинец[1], Е. А. Калинина[2], К. П. Якушенокс[3], А. А. Нецветайлов[1], Е. С. Малыгина[1],
[1]Балтийский федеральный университет им. И.Канта
[2]Национальный исследовательский университет ИТМО
[3]НИУ ВШЭ
E-mail: artkuninets@yandex.ru, e.kalinina@itmo.ru, kpyakushenoks@edu.hse.ru,
andrey.netsvetaylov@bk.ru, EMalygina@kantiana.ru

Аннотация

В работе представлено исследование квазициклических альтернантных кодов, их структуры и криптографической стойкости. Данные коды могли бы уменьшить длину ключа для постквантовых систем на кодах, однако, ввиду особенностей построения квазициклических кодов Гоппы возникает возможность редуцирования ключевого пространства оригинального кода к ключевому пространству кода с меньшими параметрами, который не является стойким к структурной атаке.

Ключевые слова: *квазициклические коды, альтернантные коды, инвариантные коды, группа автоморфизмов кода.*

Введение

Активное развитие квантовых технологий и стремительный рост вычислительной мощности квантового компьютера ставит под угрозу безопасность современных криптографических стандартов. Это связано с тем, что криптостойкость большинства асимметричных алгоритмов основывается на сложности задач факторизации целых чисел (например, криптосистема RSA) и дискретного логарифмирования (например, протокол Диффи-Хеллмана), что делает их неустойчивыми к атакам с использованием квантового компьютера. Однако, в последние несколько лет успешно развивается направление постквантовой криптографии, к которому относятся алгоритмы, основывающиеся на сложности задач, для которых не существует полиномиального алгоритма решения, даже на квантовом компьютере.

В конце 2016 года Национальный институт стандартов и технологий США (The National Institute of Standards and Technology – NIST) объявил о начале конкурса по стандартизации постквантовых алгоритмов. Одним из перспективных направлений в этой области стала криптография на кодах, исправляющих ошибки.

В данной работе будет рассмотрен принцип построения квазициклических кодов Гоппы, которые использовались в криптосистеме BIG QUAKE – участнике первого раунда конкурса стандартизации NIST. Далее будет рассмотрено редуцирование ключевой безопасности квазициклических кодов Гоппы к ключевой безопасности инвариантного кода с меньшими параметрами, описанное в [**?**], а также описана структурная атака на рассматриваемый квазициклический код.

Введение в алгебраические коды

Группа автоморфизмов кода

Пусть $\mathfrak{S}_n$ – группа перестановок множества $\{1, \ldots, n\}$. Определим группу перестановок кода.

Определение 1. (Группа перестановок кода). Пусть $\mathscr{C}$ – линейный код длинны n над полем $\mathbb{F}_q$, а $\sigma \in \mathfrak{S}_n$ – перестановка, действующая на кодовое слово следующим образом: $\sigma(\mathbf{c}) = \left(c_{\sigma(1)}, \ldots, c_{\sigma(n)}\right)$. Код $\mathscr{C}$ называется *инвариантным относительно* σ, если $\sigma(\mathscr{C}) = \mathscr{C}$. Группа перестановок кода $\mathscr{C}$ определяется следующим образом:

$$\mathrm{Perm}(\mathscr{C}) := \{\sigma \in \mathfrak{S}_n \mid \sigma(\mathscr{C}) = \mathscr{C}\}.$$

Определение 2. (Группа автоморфизмов кода). Пусть $\mathscr{C}$ – линейный код длинны n над $\mathbb{F}_q$, π – любой автоморфизм конечного поля $\mathbb{F}_q$, $a \in \left(\mathbb{F}_q^*\right)^n$ и $\sigma \in \mathfrak{S}_n$ – перестановка. Определим действие на $\mathscr{C}$ полупрямого произведения $\left(\mathbb{F}_q^*\right)^n \rtimes (\mathrm{Aut}\,(\mathbb{F}_q) \times \mathfrak{S}_n)$ следующим образом:

$$(\mathbf{a}, \pi, \sigma)(\mathbf{c}) := \left(\pi\left(a_1 c_{\sigma(1)}\right), \ldots, \pi\left(a_n c_{\sigma(n)}\right)\right),$$

где $\mathbf{c} \in \mathscr{C}$. Более того, существует групповой закон в $\left(\mathbb{F}_q^*\right)^n \rtimes (\mathrm{Aut}\,(\mathbb{F}_q) \times \mathfrak{S}_n)$, заданный следующим соотношением:

$$(\mathbf{a}, \pi, \sigma) \cdot (\mathbf{b}, \gamma, \tau) := (\mathbf{a} \star \pi(\sigma(\mathbf{b})), \pi \circ \gamma, \sigma \circ \tau).$$

Наборы элементов $(\mathbf{a}, \pi, \sigma) \in \left(\mathbb{F}_q^*\right)^n \rtimes (\mathrm{Aut}\,(\mathbb{F}_q) \times \mathfrak{S}_n)$, которые допускают инвариантность $\mathscr{C}$, составляют группу автоморфизмов кода $\mathscr{C}$, обозначать которую мы будем как $\mathrm{Aut}(\mathscr{C})$.

АГ-коды на проективной прямой

В данной части мы будем рассматривать алгебро-геометрические коды (АГ-коды) и их подполевые подкоды, определенные на проективной прямой. Такие коды также известны как обобщенные коды Рида-Соломона (GRS-коды) и альтернантные коды, соответственно.

Пусть $\mathbf{P}^1$ – это проективная прямая над полем $\mathbb{F}_{q^m}$, тогда $\mathbb{F}_{q^m}(\mathbf{P}^1) = \mathbb{F}_{q^m}(x)$ – функциональное поле проективной прямой. $\mathbb{F}_{q^m}(x)$ также называется рациональным функциональным полем над $\mathbb{F}_{q^m}$, а полюсом функции x называется бесконечно удаленная точка прямой $\mathbf{P}^1$, которую будем обозначать P_∞.

Обозначим за $\mathcal{P} \subseteq \mathbf{P}^1$ – множество попарно различных рациональных точек, а за G – дивизор $\mathbb{F}_{q^m}(\mathbf{P}^1)$, носитель которого не пересекается с носителем $\mathcal{P}$. Определим рациональный АГ-код $\mathscr{C}$ над $\mathbf{P}^1$:

$$\mathscr{C} := C_L(\mathbf{P}^1, \mathcal{P}, G).$$

Далее обозначим точки множества $\mathcal{P}$ как $P_i = (x_i : 1)$ для всех $i \in \{1, \ldots, n\}$. Чтобы провести аналогию с классическим определением GRS-кодов, построим порождающую матрицу кода $\mathscr{C}$. Будем считать, что $G\ \deg(G)P_\infty$, это означает, $G + (h) = \deg(G)P_\infty$, где (h) – главный дивизор функции $h \in \mathbb{F}_{q^m}(x)$. Более того, для любого $s \in N$ пространство Римана–Роха, ассоциированное с дивизором sP_∞ имеет вид:

$$L(sP_\infty) = \{x^i | i \in \{0, ..., s\}\}.$$

Теперь запишем пространство Римана–Роха, ассоциированного с дивизором G:

$$L(G) = \{hx^i | i \in \{0, ..., s\}\}.$$

Напомним, что размерность кода равна $k = deg(G) + 1$. Пусть $x := (x(P_1), \ldots, x(P_n)) = (x_1, \ldots, x_n)$ и $y := (h(P_1), \ldots, h(P_n))$, тогда матрица

$$\mathbf{V_r(x,y)} := \begin{pmatrix} y_1 & \ldots & y_n \\ y_1 x_1 & \ldots & y_n x_n \\ y_1 x_1^2 & \ldots & y_n (x_n)^2 \\ \vdots & & \vdots \\ y_1 x_1^{k-1} & \ldots & y_n x_n^{k-1} \end{pmatrix}$$

является порождающей матрицей кода $\mathscr{C}$.

Определение 3. (Обобщённые коды Рида-Соломона) [**?**]. Зададим вектор $\mathbf{x} := (x_1, x_2, \ldots, x_n) \in \mathbb{F}_{q^m}^n$, состоящий из n различных элементов набор, и вектор $\mathbf{y} := (y_1, y_2, \ldots, y_n) \in (\mathbb{F}_{q^m}^*)^n$, $k \in \mathbb{Z}_+$, состоящий из n ненулевых элементов. Обобщённый код Рида-Соломона (GRS-код) размерности k задаётся следующим образом:

$$\mathbf{GRS}_k(\mathbf{x},\mathbf{y}) := \{(y_1 f(x_1), \ldots, y_n f(x_n))| f \in \mathbb{F}_{q^m}[z] \text{ и } \deg(f) < k\}.$$

Вектор $\mathbf{x}$ называется носителем, а вектор $\mathbf{y}$ – множителем кода $\mathbf{GRS}_k(\mathbf{x},\mathbf{y})$. Также отметим, что любой рациональный АГ-код является обобщенным кодом Рида-Соломона.

Определение 4. (Альтернантный код). Пусть $\mathbf{x} \in \mathbb{F}_{q^m}^n$ – это носитель, а $\mathbf{y} \in (\mathbb{F}_{q^m}^*)^n$ – множитель кода $\mathbf{GRS}_k(\mathbf{x},\mathbf{y})$, тогда альтернантный код над $\mathbb{F}_q$ задается следующим образом:

$$\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y}) := \mathbf{GRS}_r(\mathbf{x},\mathbf{y})^{\perp} \cap \mathbb{F}_q^n.$$

где r – это степень кода $\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})$.

Так как альтернантные коды являются подполевыми подкодами GRS-кодов и, соответственно, являются подполевыми подкодами АГ-кодов, далее мы будем использовать алгебро-геометрическую нотацию:

$$\mathscr{A}_{r,q}(\mathcal{P}, G) := C_{\mathrm{L}}\left(\mathbf{P}^1, \mathcal{P}, G\right)^{\perp} \cap \mathbb{F}_q^n, \tag{9}$$

Определение 5. (Классический код Гоппы). Пусть $\mathbf{x} \in \mathbb{F}_{q^m}^n$ – вектор с попарно различными координатами и $\Gamma \in \mathbb{F}_{q^m}[z]$ – многочлен, такой, что $\Gamma(x_i) \neq 0$ для любых $i \in \{0, \ldots, n-1\}$. Классический код Гоппы $\mathscr{G}_q(\mathrm{x}, \Gamma)$, ассоциированный с Γ и порождённый $\mathbf{x}$, определяется следующим образом:

$$\mathscr{G}_q(\mathbf{x}, \Gamma) := \mathscr{A}_{r,q}\left(\mathbf{x}, \Gamma(\mathbf{x})^{-1}\right),$$

где $r := \deg \Gamma$. Полином Γ называют многочленом Гоппы, а m – степенью расширения кода Гоппы.

Теорема 3. *Из определения 5 следует, что классический код Гоппы можно построить используя алгебро-геометрический подход. В терминах АГ-кода классический код Гоппы примет вид:*

$$\mathscr{G}_q(\boldsymbol{x}, \Gamma) := C_{\mathrm{L}}\left(\mathbf{P}^1, \mathcal{P}, G_0 - P_\infty\right)^{\perp} \cap \mathbb{F}_q^n = C_{\mathrm{L}}\left(\mathbf{P}^1, \mathcal{P}, A - G_0\right) \cap \mathbb{F}_q^n,$$

где $A = (h'(z)) + (n-1)P_\infty$, $h(z) = \prod_{x_i \in x}(z - x_i)$, $G_0 = (\Gamma)_0$ – *дивизор нулей функции* Γ.

Квазициклические альтернантные коды

Квазициклические коды

Пусть $\mathbb{F}$ – конечное поле, ℓ – положительное целое.

Определение 6. Пусть $\sigma : \mathbb{F}^n \to \mathbb{F}^n$ – отображение циклического сдвига

$$\sigma_\ell : \begin{pmatrix} \mathbb{F}^\ell & \to & \mathbb{F}^\ell \\ (x_0, x_1, ..., x_{\ell-1}) & \mapsto & (x_\ell, x_0, ..., x_{\ell-1}) \end{pmatrix}.$$

Пусть n – целое и $\ell | n$. Тогда σ_ℓ – ℓ-квазициклический сдвиг, полученный поблочным применением σ:

$$\sigma : \begin{pmatrix} \mathbb{F}^n & \to & \mathbb{F}^n \\ \left(\mathbf{b}_1 | ... | \mathbf{b}_{\frac{n}{\ell}}\right) & \mapsto & \left(\sigma_\ell(\mathbf{b}_1) | ... | \sigma_\ell(\mathbf{b}_{\frac{n}{\ell}})\right) \end{pmatrix},$$

где $\mathbf{b}_i$ – блоки длины ℓ, $i = \overline{1, \frac{n}{\ell}}$. Это определение проиллюстрировано на рисунке 1

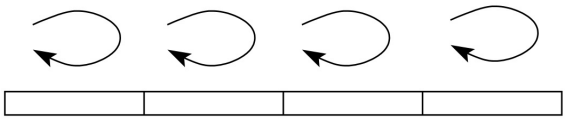


Рис. 1: Иллюстрация квазициклического сдвига

Определение 7. Код $\mathscr{C} \subseteq \mathbb{F}^n$ называется *ℓ–квазициклическим* (ℓ-QC), если код устойчив относительно отображения циклического сдвига σ. При этом ℓ называется *порядком* квазицикличности кода.

Определение 8 (Блочно-циркулянтная матрица)**.** Пусть $\mathbf{M}$ – матрица. Матрица называется *ℓ-блочно-циркулянтной*, если она состоит из $\mathbf{M}_i$ циркулянтных матриц порядка ℓ:

$$\mathbf{M} = \left(\begin{array}{c|c|c} & & \\ \hline \ldots & \mathbf{M}_i & \ldots \\ \hline & & \end{array}\right), \quad \text{где} \quad \mathbf{M} := \begin{pmatrix} a_0 & a_1 & \ldots & a_{\ell-1} \\ a_{\ell-1} & a_0 & \ldots & a_{\ell-2} \\ \vdots & \ddots & \ddots & \vdots \\ a_1 & \ldots & a_{\ell-1} & a_0 \end{pmatrix}.$$

Матрица такого типа может быть представлена первыми строками каждого блока размера ℓ. Таким образом, мы можем восстановить ℓ-блочно-циркулянтную матрицу $\mathbf{M}$ из k строк, зная ее строки с индексом $1, \ell+1, \ldots, k-\ell+1$.

Замечание. Код $\mathscr{C}$, имеющий ℓ-блочно-циркулянтную порождающую матрицу, является ℓ-QC кодом. Отметим, что QC-код $\mathscr{C}$ размерности k не обязан иметь ℓ-блочно-циркулянтную порождающую матрицу. Однако будет существовать ℓ-блочно-циркулянтная матрица $\mathbf{G}$, состоящая из $k' \geq k$ строк, которая порождает данный код $\mathscr{C}$.

Чтобы оптимально уменьшить размер открытого ключа криптосистемы Мак-Элиса, далее будем рассматривать коды, удовлетворяющие следующему условию: ℓ-QC код $\mathscr{C}$ с параметрами $[n, k]$ допускает порождающую матрицу вида

$$\mathbf{M} = \left(\mathbf{I}_k | \mathbf{M}\right),$$

где $\mathbf{I}_k$ – единичная матрица порядка k, $\mathbf{M}$ – ℓ-блочно-циркулянтная матрица. В этом случае будем называть код $\mathscr{C}$ – *систематическим квазициклическим*.

Замечание. Для выполнения предыдущего условия размерность ℓ-QC кода должна быть кратна ℓ.

Определение 9. Получив матрицу $\mathbf{G}$ в систематической форме, определим $\rho(\mathbf{G})$ как матрицу, полученную путём удаления первой строки каждого блока матрицы $\mathbf{M}$. То есть $\rho(\mathbf{G})$ получается путём записывания строк из $\mathbf{M}$ с индексами $1, \ell+1, 2\ell+1, \ldots, (n-k)-\ell+1$. Отсюда имеем следующее соотношение:

$$\mathrm{NumberOfRows}(\mathbf{G}) = \ell \times \mathrm{NumberOfRows}(\rho(\mathbf{G})).$$

Определение 10 (Инвариантный код)**.** Пусть $\mathscr{C} \subseteq \mathbb{F}^n$ – ℓ-QC код. *Инвариантный код* определяется следующим образом:

$$\mathscr{C}^\sigma := \{c \in \mathscr{C} | \sigma(c) = c\}.$$

Так как данный код имеет повторяющиеся элементы, далее мы будем использовать *проколотый инвариантный код*, который определяется следующим образом:

$$\bar{\mathscr{C}^\sigma} := \mathrm{Punct}_{\mathcal{I}_\ell}(\mathscr{C}^\sigma),$$

где $\mathcal{I}_\ell := \{1, \ldots, n\} \backslash \{1, \ell+1, \ldots, n-\ell+1\}$. Пусть $\sigma_{\mathscr{C}}$ – ℓ-QC сдвиг σ, суженный на код $\mathscr{C}$, тогда можно записать $\bar{\mathscr{C}^\sigma} = \mathrm{Punct}_{\mathcal{I}_\ell}(\ker(\sigma_{\mathscr{C}} - \mathrm{id}))$.

Далее под инвариантным кодом будем иметь ввиду и инвариантный, и проколотый инвариантный код, однако обозначения останутся разными.

Замечание. Отметим, что инвариантность коммутирует с операцией вычисления подполевого подкода. Действительно, если $\mathscr{C}$ – линейный код над $\mathbb{F}_{q^m}$, устойчивый относительно действия перестановки σ, то

$$(\mathscr{C} \cap \mathbb{F}_q^n)^\sigma = \{c \in \mathscr{C} | c \in \mathbb{F}_q^n \text{ и } \sigma(c) = c\} = \mathscr{C}^\sigma \cap \mathbb{F}_q^n.$$

Индуцированные перестановки альтернантных кодов

В этой главе мы рассмотрим метод построения альтернантных кодов, определённых над $\mathbb{F}_{q^m}$, устойчивых к заданной перестановке множества $\{1, \ldots, n\}$, где n – длина кода. Так как альтернантные коды являются подполевыми подкодами GRS-кодов, для начала необходимо исследовать GRS-коды и их перестановки.

Дадим определение проективной линейной группы:

$$PGL_2(\mathbb{F}_q^m) = \left\{ \begin{matrix} \mathbf{P}^1 \to \mathbf{P}^1 \\ (x:y) \mapsto (ax+by : cx+dy) \end{matrix} \;\middle|\; \left\{ \begin{matrix} a,b,c,d \in \mathbb{F}_{q^m}, \\ ad - bc \neq 0 \end{matrix} \right\} \right\}.$$

В [**?**] рассмотрено алгебро-геометрическое построение GRS-кодов, а также доказано, что вся группа перестановок кода индуцирована действием проективной линейной группы $PGL_2(\mathbb{F}_{q^m})$, являющейся группой автоморфизмов проективной прямой $\mathbf{P}^1$.

Элементы $PGL_2(\mathbb{F}_{q^m})$ также имеют матричное представление, а именно,

$$\forall \sigma \in PGL_2(\mathbb{F}_{q^m}) \text{ можно представить как } \sigma = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \text{ где } ad - bc \neq 0.$$

Определение 11. Пусть $\mathcal{P} \subseteq \mathbf{P}^1$ – множество n попарно различных точек проективной прямой, G, G' – дивизоры на $\mathbf{P}^1$, такие, что $\mathrm{supp}(G), \mathrm{supp}(G') \cap \mathrm{supp}(\mathcal{P}) = \varnothing$. Определим отношение эквивалентности дивизоров относительно $\mathcal{P}$ следующим образом:

$$G \sim_{\mathcal{P}} G', \text{если } \exists f \in \mathbb{F}_{q^m}(\mathbf{P}^1) \;\mid\; f \neq 0;\; G - G' = (f) \text{ и } f(P) = 1 \text{ для } \forall P \in \mathcal{P}.$$

Лемма 1. *Пусть $\mathcal{P} \subseteq \mathbf{P}^1$ и G, G' – два дивизора $\mathbf{P}^1$, такие что $\mathrm{supp}(G) \cap \mathrm{supp}(\mathcal{P}) = \varnothing$ и $\mathrm{supp}(G') \cap \mathrm{supp}(\mathcal{P}) = \varnothing$. Если $G \sim_{\mathcal{P}} G'$, тогда $C_L(\mathbf{P}^1, \mathcal{P}, G) = C_L(\mathbf{P}^1, \mathcal{P}, G')$.*

Следующая теорема определяет всевозможные перестановки для АГ-кода, ассоциированного с дивизором G.

Теорема 4. *Пусть $\mathscr{C} = C_L(\mathbf{P}^1, \mathcal{P}, G) \subseteq \mathbb{F}_{q^m}^n$ – АГ-код, $1 \leq \deg(G) \leq n-3$. Тогда*

$$\mathrm{Perm}(\mathscr{C}) = \left\{ \sigma \in \mathrm{Aut}\left(\mathbf{P}^1\right) \mid \sigma(\mathcal{P}) = \mathcal{P} \text{ и } \sigma(G) \sim_{\mathcal{P}} G \right\}.$$

Теперь перейдём непосредственно к построению GRS-кодов с перестановками. Пусть $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ – автоморфизм, действующий на носитель $\mathcal{P} \subseteq \mathbf{P}^1$ и дивизор $G \in \mathrm{Div}(\mathbf{P}^1)$. Тогда σ индуцирует перестановку кода $\mathscr{C} = C_L(\mathbf{P}^1, \mathcal{P}, G)$. Индуцированная перестановка $\tilde{\sigma} \in \mathfrak{S}_n$ определяется следующим образом:

$$\tilde{\sigma} : \left\{ \begin{array}{c} \mathscr{C} \longrightarrow \mathscr{C} \\ (f(P_1), \ldots, f(P_n)) \longmapsto (f(\sigma(P_1)), \ldots, f(\sigma(P_n))) \end{array} \right\}$$

Так как альтернантные коды являются подполевыми подкодами GRS-кодов, то их группа перестановок в точности является группой перестановок исходного GRS-кода.

Лемма 2. *Пусть $\mathscr{C} := \mathscr{A}_{r,q}(\mathcal{P}, G)$ – альтернантный код над $\mathbb{F}_q$ и $\sigma \in \mathrm{Perm}(C_L(\mathbf{P}^1, \mathcal{P}, G))$, тогда $\sigma \in \mathrm{Perm}(\mathscr{C})$.*

Построение квазициклических альтернантных кодов

Теперь мы имеем все необходимые свойства для построения альтернатных кодов, инвариантных относительно заданной перестановки.

Пусть $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ и $\mathrm{ord}(\sigma) = \ell$. Для точки $P \in \mathbf{P}^1$ определим множество

$$\mathrm{Orb}_\sigma(P) := \left\{ \sigma^i(P) \mid i \in \{0, \ldots, \ell - 1\} \right\},$$

являющееся её орбитой относительно действия σ. Пусть n – целое положительное число и $\ell | n$. Тогда определим носитель кода:

$$\mathcal{P} := \coprod_{i=1}^{n/\ell} \mathrm{Orb}_\sigma(P_i),$$

где точки $P_i \in \mathbf{P}^1(\mathbb{F}_{q^m})$ попарно различны. Затем определим дивизор

$$G := \sum_{i=1}^{s} t_i \sum_{Q \in \mathrm{Orb}(Q_i)} Q,$$

где Q_i – точки $\mathbf{P}^1$, $s \in \mathbb{N}, t_i \in \mathbb{Z}$ для $i \in \{1, \ldots, s\}$. При этом

$$\deg(G) = \sum_{i=1}^{s} t_i \ell \deg(Q_i).$$

Как было показано в разделе 2.2, автоморфизм σ индуцирует перестановку $C_{\mathrm{L}}(\mathbf{P}^1, \mathcal{P}, G)$. Для краткости, и автоморфизм $\mathbf{P}^1$, и индуцированную перестановку мы будем обозначать через σ. Тогда, согласно Лемме 2, σ также является перестановкой кода $\mathscr{A}_{r,q}(\mathcal{P}, G) := C_{\mathrm{L}}(\mathbf{P}^1, \mathcal{P}, G)^{\perp} \cap \mathbb{F}_q^n$.

Структурный анализ инвариантных кодов

В данном разделе мы покажем, что инвариантный код для QC альтернантного кода также является инвариантным кодом. Так как инвариантная операция коммутирует с операцией подполевого подкода, то для анализа структуры инвариантных кодов достаточно рассмотреть инвариантный код GRS-кода.

Лемма 3. *Пусть $\mathscr{C} := C_L(\mathbf{P}^1, \mathcal{P}, G)$ и $\sigma \in \mathrm{Perm}(\mathscr{C})$. Если $c = \mathrm{Ev}_{\mathcal{P}}(f) \in \mathscr{C}$ такое, что $\sigma(c) = c$, тогда f является σ-инвариантным т.е. $f \circ \sigma = f$.*

Для упрощения рассуждений мы предполагаем, что G строится на основе одной рациональной точки Q, т.е. $G = t\sum_{R\in \mathrm{Orb}_\sigma(Q)} R$. В общем случае полученный результат останется корректным.

Зададим

$$\sigma^j(P_i) := (\alpha_{i\ell+j} : \beta_{i\ell+j}) \text{ для } i \in \left\{0,\dots,\frac{n}{\ell}-1\right\}, j \in \{0,\dots,\ell-1\},$$
$$\sigma^j(Q) := (\gamma_j : \delta_j) \text{ для } j \in \{0,\dots,\ell-1\}.$$

Лемма 4. *Пусть $G = t\sum_{R\in Orb_\sigma(Q)} R$, тогда имеем*

$$L(G) = \left\{ \frac{F(X,Y)}{\prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y)^t} \mid F \in \mathbb{F}_{q^m}[X,Y] \text{ – однородный многочлен степени } t\ell. \right\}$$

Далее мы изучим действие автоморфизма $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ на пространство Римана-Роха $L(G)$.

Теорема 5. *Пусть $C_L(\mathbf{P}^1,\mathcal{P},G) \subseteq \mathbb{F}_{q^m}^n$ – АГ-код длины n, размерности k, с действующим на него автоморфизмом $\sigma \in PGL_2(\mathbb{F}_{q^m})$ порядка ℓ, где $\ell \mid n$. Пусть, как и выше, определены $\mathcal{P}$ и G. Тогда инвариантный код $\overline{C_L(\mathbf{P}^1,\mathcal{P},G)}^{\sigma}$ является АГ-кодом длины n/ℓ и размерности $\lfloor k/\ell \rfloor$.*

Следствие 1. *Пусть $\mathscr{A}_{r,q}(\mathcal{P},G) := C_L(\mathbf{P}^1,\mathcal{P},G) \cap \mathbb{F}_q^n$ – альтернантный код длины n, порядка r, с действующим на него автоморфизмом $\sigma \in PGL_2(\mathbb{F}_{q^m})$ порядка ℓ, где $\ell \mid n$. Пусть $\mathcal{P}$ и G определены как в (3.4) и (3.5). Тогда инвариантный код $\overline{\mathscr{A}_{\lfloor r/\ell \rfloor,q}(\mathcal{P},G)}^{\sigma}$ является альтернантным кодом длины n/l и порядка $\lfloor r/\ell \rfloor$*

Рассмотрим $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ с $\ell = \mathrm{ord}(\sigma)$, а также носитель $\mathcal{P}$ и дивизор G, определенные ранее. Автоморфизм $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ можно представить в виде матрицы $M \in \mathrm{GL}_2(\mathbb{F}_{q^m})$. Нотация $M \sim N$, где $M, N \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$, означает, что существует матрица $P \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$, такая, что $M = PNP^{-1}$. В зависимости от собственных векторов матрицы M мы можем выделить три случая:

1. $M \sim \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$, где $a \in \mathbb{F}_{q^m}$ (σ диагонализирована в $\mathbb{F}_{q^m}$),

2. $M \sim \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$, где $b \in \mathbb{F}_{q^m}$ (σ тригонализирована в $\mathbb{F}_{q^m}$),

3. $M \sim \begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{q^m} \end{pmatrix}$, где $\alpha \in \mathbb{F}_{q^{2m}}$ (σ диагонализирована в $\mathbb{F}_{q^{2m}}$).

Следующее предположение позволяет нам получить структуру однородных полиномов, зафиксированных диагональным автоморфизмом σ.

Предложение 3. *Пусть $F \in \mathbb{F}_{q^m}[X,Y]$ – однородный полином степени $t\ell$, и $a \in \mathbb{F}_{q^m}$ порядка ℓ. Если $F(aX,Y) = F(X,Y)$, тогда $F(X,Y) = R\left(X^\ell, Y^\ell\right)$, где $R \in \mathbb{F}_{q^m}[X,Y]$ является однородным полиномом степени t.*

Описание атаки

В этом разделе покажем, что ключевую безопасность QC-альтернантного кода можно редуцировать к ключевой безопасности его инвариантного кода. Рассмотрим автоморфизм $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ и альтернантный код

$$\mathscr{A}_{r,q}(\mathcal{P},G) := C_{\mathrm{L}}\left(\mathbf{P}^1,\mathcal{P},G\right)^{\perp} \cap \mathbb{F}_q^n,$$

являющийся устойчивым относительно действия σ. Носитель $\mathcal{P}$ и дивизор G определены в (2) and (3). Зная попрождающую матрицу кода $\mathscr{A}_{r,q}(\mathcal{P}, G)$ и индуцированную перестановку σ, можно вычислить инвариантный код $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)}^{\sigma}$. Обозначим за инвариантный код – альтернантный код $\mathscr{A}_{r,q}(\tilde{\mathcal{P}}, \tilde{G})$ для некоторых носителя $\tilde{\mathcal{P}}$ и дивизора $\tilde{G}$ с малыми параметрами. Зная $\tilde{\mathcal{P}}$ и $\tilde{G}$, можно восстановить $\mathcal{P}$ и G. Существует взаимосвязь между $\tilde{\mathcal{P}}$ и $\tilde{G}$ инвариантного кода с $\mathcal{P}$ и G исходного альтернантного кода.

Предложение 4. *Пусть $\mathscr{C} = C_L(\mathbf{P}^1, \mathcal{P}, G)$ – АГ-код и*

$$\sigma : \begin{array}{ccc} \mathbf{P}^1 & \to & \mathbf{P}^1 \\ (x:y) & \mapsto & (ax:y) \end{array} \tag{10}$$

Определим $\tilde{P}_i = (\alpha_i^\ell : \beta_i^\ell)$ и $\tilde{G} = t \cdot \left((-1)^{\ell-1} a^{\frac{\ell(\ell-1)}{2}} \left(\frac{\gamma_0}{\delta_0}\right)^\ell : 1 \right)$ или $\tilde{G} = P_\infty$. Тогда справедливо $\mathscr{C}^{\bar{\sigma}} = C_L(\mathbf{P}^1, \tilde{\mathcal{P}}, \tilde{G})$.

Будем предполагать, что $\tilde{\mathcal{P}}$ и $\tilde{G}$ для инвариантного кода $\mathscr{A}_{r,q}(\tilde{\mathcal{P}}, \tilde{G})$ известны. Обозначим $\tilde{\mathcal{P}} := \left\{ \left(\tilde{\alpha}_i : \tilde{\beta}_i \right) \mid i \in \left\{1, \ldots, \frac{n}{\ell}\right\} \right\}$. Также предположим, что G строится с помощью орбиты одной рациональной точки Q. Для публичного кода мы будем использовать следующие обозначения:

$$\begin{aligned} \mathcal{P} &:= \left\{ (\alpha_{i,j} : 1) \mid i \in \left\{1, \ldots, \frac{n}{\ell}\right\}, j \in \{0, \ldots, \ell-1\} \right\}, \\ G &:= t \sum_{j=0}^{\ell-1} \sigma^j(Q), \end{aligned}$$

где $\sigma^j(Q) := (\gamma_j : \delta_j)$ для всех $j \in \{0, \ldots, \ell-1\}$.

Восстановление дивизора и носителя

Отметим, что в (10) элемент $a \in \mathbb{F}_{q^m}^\times$ является примитивным корнем степени ℓ из единицы. Существует $\varphi(\ell) < n$ таких a, где φ – функция Эйлера. Далее будем предполагать, что нам известен элемент a.

Зная параметры a и $\tilde{Q}$, мы можем восстановить дивизор G.

Замечание. В случае, если $\tilde{Q} \neq P_\infty$, для всех $i \in \{0, \ldots, \ell-1\}$ имеем

$$\tilde{Q} = \left((-1)^{\ell-1} \left(a^i\right)^{\frac{\ell(\ell-1)}{2}} \left(\frac{\gamma_i}{\delta_i}\right)^\ell : 1 \right). \tag{11}$$

Обозначим $\mu_\ell := \{\sigma^i(a) \mid i \in \{0, \ldots, \ell-1\}\}$ и затем для каждого $a \in \mu_\ell$ восстанавливаем соответствующую точку носителя дивизора G. Отметим, что множество μ_ℓ состоит из примитивных корней степени ℓ из единицы.

Более детально вычисление дивизора G на основании знания $\tilde{G}$, описано в Алгоритме 1. основной и наиболее затратный шаг – вычисление корней многочлена $p(X) := a^{\frac{\ell(\ell-1)}{2}} X^\ell - \tilde{\gamma} \in \mathbb{F}_{q^m}[X]$, что можно сделать, используя, например, алгоритм Берлекэмпа.

Далее восстановим носитель $\mathcal{P}'$ при условии, что $\mathscr{A}_{r,q}(\mathcal{P}', G) = \mathscr{A}_{r,q}(\mathcal{P}, G)$. Координаты точки $P := (x : y)$ из $\mathcal{P}$ удовлетворяют системе

$$\begin{cases} x^\ell - \tilde{\alpha}_i = 0 \\ y^\ell - \tilde{\beta}_i = 0 \end{cases} \tag{12}$$

Алгоритм 1 Восстановление дивизора G

Вход: $\tilde{G}$ — дивизор инвариантного кода $\overline{\mathcal{A}_r, q(P,G)}^{\sigma}$.

Выход: Дивизор G.

1: $a \leftarrow a^{\ell} \equiv 1 \pmod{q^m}$
2: **Если** $\widetilde{Q} \neq P_\infty$ **тогда**
3: $\quad \Gamma \leftarrow$ корни$(a^{\ell(\ell-1)/2}X^{\ell} - \tilde{\gamma})$
4: $\quad G \leftarrow t\sum\limits_{\gamma\in\Gamma}(\gamma : 1)$
5: **иначе**
6: $\quad G = t \cdot \ell \cdot P_\infty$
7: **Вернуть** G

для $i \in \left\{1, \ldots, \frac{n}{\ell}\right\}$, где $\left(\tilde{\alpha}_i : \tilde{\beta}_i\right) = \tilde{P}_i$. Зная $\tilde{\mathcal{P}}$, мы можем восстановить все элементы $\mathcal{P}$, однако они будут представлять собой неупорядоченное множество. Найдем решение (α_i, β_i) в (12) для каждого $i \in \left\{1, \ldots, \frac{n}{\ell}\right\}$ и выберем $a \in \mu_\ell$. Далее будем полагать, что множество

$$\mathcal{P}' := \left\{ \left(a^j \frac{\alpha_i}{\beta_i} : 1 \right) \mid j \in \{0, \ldots, \ell - 1\}, i \in \left\{1, \ldots, \frac{n}{\ell}\right\} \right\}$$

является носителем кода $\mathscr{A}_{r,q}(\mathcal{P}', G)$, являющегося перестановочным относительно кода $\mathscr{A}_{r,q}(\mathcal{P}, G)$. Для каждого множества решений $S := \left\{(\alpha_i, \beta_i) \mid i \in \left\{1, \ldots, \frac{n}{\ell}\right\}\right\}$ и всякого $a \in \mu_\ell$ мы имеем различные соответствующие им носители $\mathcal{P}'$.

Восстановление перестановки

Восстановим перестановку между $\mathscr{A}_{r,q}(\mathcal{P}', G)$ и $\mathscr{A}_{r,q}(\mathcal{P}, G)$. Пусть $\mathbf{G}$ – порождающая матрица кода $\mathscr{A}_{r,q}(\mathcal{P}, G)$, $\mathbf{H}'$ – проверочная матрица кода $\mathscr{A}_{r,q}(\mathcal{P}', G)$. перестановку между $\mathscr{A}_{r,q}(\mathcal{P}, G)$ и $\mathscr{A}_{r,q}(\mathcal{P}', G)$ зададим с помощью матрицы $\mathbf{\Pi}$:

$$\mathbf{G} \cdot \mathbf{\Pi} \cdot \mathbf{H}'^{\top} = 0. \tag{13}$$

Предположим, мы выбрали $a \in \mu_\ell$, тогда перестановочная матрица $\mathbf{\Pi}$ имеет следующий вид:

$$\begin{pmatrix} \sum_{i=1}^{\ell} x_{\ell,i}\mathbf{J}^i & \ldots & (0) \\ \vdots & \ddots & \vdots \\ (0) & \ldots & \sum_{i=1}^{\ell} x_{\frac{n}{\ell},i}\mathbf{J}^i \end{pmatrix}, \text{ где } \mathbf{J} = \begin{pmatrix} 0 & \ldots & \ldots & 0 & 1 \\ 1 & \ddots & & & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 \end{pmatrix}$$

Отметим, что $\mathbf{J}$ – матрица размера $\ell \times \ell$, а $x_{j,i} \in \{0, 1\}$ – неизвестные для $j \in \left\{1, \ldots, \frac{n}{\ell}\right\}$ и $i \in \{1, \ldots, \ell\}$. В таком случае линейная система (13) имеет n неизвестных. Если мы предположим, что $k \leq n - \frac{1}{n}$, то $n \leq (n-k)k$, и мы можем найти единственное решение для $\mathbf{\Pi}$.

В Алгоритме 2 представлено восстановление перестановочной матрицы $\mathbf{\Pi}$ и подходящего выбора элемента $a \in \mu_\ell$.

Алгоритм 2 Восстановление носителя $\mathcal{P}$

Вход: $\mathbf{G}$ – порождающая матрица квазициклического альтернантного кода, дивизор G, носитель $\widetilde{\mathcal{P}}$ инвариантного кода.

Выход: $\varnothing$, если решение не найдено. В противном случае, $\mathcal{P}'$ такое, что $\mathcal{A}_{r,q}(\mathcal{P}', G) = \mathcal{A}_{r,q}(\mathcal{P}, G)$.

Для $i \in \{1, \ldots, n\}$ **сделать**
  $\alpha_i \leftarrow$ корни $(x^\ell - \tilde{\alpha}_i)$
  $\beta_i \leftarrow$ корни $(y^\ell - \tilde{\beta}_i)$
Для $a \in \mu_\ell$ **сделать**
  /* Угадываем $\mathcal{P}'$ */
  $\mathcal{P}' \leftarrow \{(a^j \frac{\alpha_i}{\beta_i} : 1) \mid j \in \{0 \ldots \ell - 1\}, i \in \{1 \ldots \frac{n}{l}\}\}$
  $\mathscr{C} \leftarrow \mathcal{A}_{r,q}(\mathcal{P}, G)$
  Если $\mathscr{C} = \mathcal{A}_{r,q}(\mathcal{P}, G)$ **тогда**
    Вернуть $\mathcal{P}'$
  иначе
    $\mathbf{H} \leftarrow$ Матрица проверки четности$(\mathscr{C})$
    $S \leftarrow$ решить $\mathbf{G} \cdot \mathbf{\Pi} \cdot \mathbf{H}'^{\top} = 0,$ где $\mathbf{\Pi}$ — перестановочная матрица
    Если $\dim(S) = 1$ **тогда**
      Вернуть $(\pi(\mathcal{P}'))$ $\quad //\pi \in \mathfrak{S}_n$ ассоциирована с $\mathbf{\Pi}$
Вернуть $\varnothing$

Пример

В данном разделе приведём пример построения квазициклического альтернантного кода $\mathscr{A}_{r,q}(\mathcal{P}, G) := C_{\mathrm{L}}\left(\mathbf{P}^1, \mathcal{P}, G\right)^{\perp} \cap \mathbb{F}_q^n$, а также пример восстановления параметров оригинального кода через порождающую матрицу и параметры кода $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)}^{\sigma}$. Все вычисления, произведённые нами, были выполнены в системах компьютерной алгебры Sage и Magma.

При построении кода будем использовать параметр $n = 21$, отображение σ, такое, что $\mathrm{ord}(\sigma) := \ell = 3$ и

$$\sigma = \begin{pmatrix} \alpha^{21} & 0 \\ 0 & 1 \end{pmatrix}.$$

Замечание. $\sigma \sim \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$, где $a \in \mathbb{F}_{q^m}$. Следовательно, рассматривается случай диагонализации, описанный в разделе 3.

Рассмотрим проективную прямую $\mathbf{P}^1$ над полем $\mathbb{F}_{64}$, а в качестве носителя $\mathcal{P}$ выберем следующие рациональные точки проективной прямой:

$$
\begin{aligned}
P_{1,2,3} &:= [(a:1), (a^4+a^3+a^2:1), (a^4+a^3+a^2+a:1)],\\
P_{4,5,6} &:= [(a^2:1), (a^5+a^4+a^3:1), (a^5+a^4+a^3+a^2:1)],\\
P_{7,8,9} &:= [(a^3:1), (a^5+a^3+a+1:1), (a^5+a+1:1)],\\
P_{10,11,12} &:= [(a^4:1), (a^3+a^2+1:1), (a^4+a^3+a^2+1:1)],\\
P_{13,14,15} &:= [(a^5:1), (a^4+a^3+a:1), (a^5+a^4+a^3+a:1)],\\
P_{16,17,18} &:= [(a^4+a^3+a+1:1), (a^5+a^4+a^2:1), (a^5+a^3+a^2+a+1:1)],\\
P_{19,20,21} &:= [(a^5+a^4+a^2+a:1), (a^5+a^4+a+1:1), (a^2+1:1)],\\
\mathcal{P} &:= \coprod_{i=1}^{n/\ell} \mathrm{Orb}_\sigma\left(P_i\right).
\end{aligned}
$$

Замечание. Для наполнения носителя $\mathcal{P}$ необходимо выбирать точки с непересекающимися орбитами.

Для построения дивизора G используем единственную точку $Q := (a^3+1:1)$ и параметр $t := 1$. В результате дивизор примет вид $G := \sum_{R\in\mathrm{Orb}(Q)} R = (a^3+1:1)+(a^5+a^2+1:1)+(a^5+a^3+a^2:1)$

В соответствии с Леммой 4, базис пространства Римана-Роха, ассоциированного с дивизором G, выглядит следующим образом:

$$
L(G) = \left\{ \frac{x^3+y^3}{x^3+(a^5+a^2+a+1)}, \frac{x\cdot y^2}{x^3+(a^5+a^2+a+1)}, \frac{x^2\cdot y}{x^3+(a^5+a^2+a+1)}, \frac{x^3}{x^3+(a^5+a^2+a+1)} \right\}.
$$

В результате получаем $[21,3]$–квазициклический код с порождающей матрицей

$$
G := \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1\\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1\\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}.
$$

Далее покажем, что знание $\overline{\mathscr{A}_{r,q}(\mathcal{P},G)}^{\sigma} = \mathscr{A}_{r,q}(\tilde{\mathcal{P}},\tilde{G})$, носителя $\tilde{\mathcal{P}}$ и дивизора $\tilde{G}$ с малыми параметрами позволит нам восстановить оригинальные параметры кода $\mathscr{A}_{r,q}(\mathcal{P},G)$.

Сперва восстановим дивизор G. Как уже было сказано ранее, в случае, если $\sigma \sim \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$, то дивизор G можно восстановить, применив Алгоритм 1:

$$
\tilde{G} = 1\cdot(a^5+a^2+a+1:1),
$$

$$
a^{\ell(\ell-1)/2}X^\ell - \tilde{\gamma} = X^3 + (a^5+a^2+a+1). \tag{14}
$$

Корням многочлена (14) соответствуют следующие точки проективной прямой:

$$
(a^3+1:1), (a^5+a^2+1:1), (a^5+a^3+a^2:1),
$$

которые входят в носитель оригинального дивизора G. Таким образом, дивизор G полностью восстановлен.

Теперь перейдём к восстановлению носителя $\mathcal{P}$. В первую очередь необходимо найти решения системы (12):

$$
\begin{cases}
x^\ell - \tilde{\alpha}_i = 0,\\
y^\ell - \tilde{\beta}_i = 0,
\end{cases}
$$

принимая во внимание, что

$$\tilde{P} = \{(a^3 : 1), (a^4 + a^3 + a + 1 : 1), (a^5 + a^4 + a^2 + 1 : 1), (a^5 + a^3 + 1 : 1), (a^5 + a^2, 1), \\ (a^4 + a^2 + a + 1 : 1), (a^3 + a^2 + a : 1)\}.$$

Корням системы соответствуют следующие точки проективной прямой:

$$(a^4 + a^3 + a^2 + a : 1), (a^5 + a^4 + a^3 + a^2 : 1), (a^5 + a^3 + a + 1 : 1), \\ (a^3 + a^2 + 1 : 1), (a^4 + a^3 + a : 1), (a^4 + a^3 + a + 1 : 1), (a^5 + a^4 + a^2 + a : 1).$$

При этом

$$\mathcal{P}' := \{(a^j \frac{\alpha_i}{\beta_i} : 1) \mid j \in \{0 \dots \ell - 1\}, i \in \{1 \dots \frac{n}{l}\}\} = \{(a^4 + a^3 + a^2 + a : 1), (a : 1), \\ (a^4 + a^3 + a^2 : 1), (a^5 + a^4 + a^3 + a^2 : 1), (a^2 : 1), (a^5 + a^4 + a^3 : 1), (a^5 + a^3 + a + 1 : 1), \\ (a^5 + a + 1 : 1), (a^3 : 1), (a^3 + a^2 + 1 : 1), (a^4 + a^3 + a^2 + 1 : 1), (a^4 : 1), (a^4 + a^3 + a : 1), \\ (a^5 + a^4 + a^3 + a : 1), (a^5 : 1), (a^4 + a^3 + a + 1 : 1), (a^5 + a^4 + a^2 : 1), \\ (a^5 + a^3 + a^2 + a + 1 : 1), (a^5 + a^4 + a^2 + a : 1), (a^5 + a^4 + a + 1 : 1), (a^2 + 1 : 1)\}.$$

Нетрудно заметить, что носители оригинального дивизора $\mathcal{P}$ и дивизора $\mathcal{P}'$ отличаются на перестановку. Последний шаг – восстановление перестановки между $\mathscr{A}_{r,q}(\mathcal{P}', G)$ и $\mathscr{A}_{r,q}(\mathcal{P}, G)$ путём решения матричного уравнения (13).

$$G := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix},$$

$$H'^T := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Перестановочная матрица имеет вид:

$$\mathbf{\Pi} := \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}$$

Отметим, что в носителе $\mathcal{P}'$ в блоках $1, 2$ на первом месте стоят третьи элементы из орбит соответствующих точек носителя оригинального дивизора, что соответствует единицам на главной диагонали матрицы $\mathbf{\Pi}$. В блоках $6, 7$ элементы не переставлены, а в блоках $3, 4, 5$ на первое место встали вторые элементы из орбит. Таким образом, найдя перестановку $\mathbf{\Pi}$, мы восстановили оригинальный носитель $\mathcal{P}$.

Кураторы исследования –

к.ф.-м.н., доцент Балтийского федерального университета им. Иммануила Канта, Малыгина Екатерина Сергеевна.

КОДЫ, ИСПРАВЛЯЮЩИЕ ОШИБКИ

Задача синдромного декодирования кодов общего положения

М. А. Арбейтер[1], К. П. Якушенокс[2]
[1]РГРТУ им. В.Ф. Уткина
[2]НИУ ВШЭ
E-mail: I.Maksar@yandex.ru, kpyackushenoks@edu.hse.ru

Аннотация

Алгоритм Штерна - один из самых эффективных подходов для решения NP-трудной поисковой задачи синдромного декодирования. В ходе данной работы была реализована вариация алгоритма Штерна SD-ISD и протестирована на кодах Гоппы.

Ключевые слова: *Линейный код, синдромное декодирование, алгоритм Штерна.*

Работа посвящена изучению известных алгоритмов для решения задачи синдромного декодирования линейных кодов.

Определение 1. V_r — векторное пространство размерности r над полем $GF(2)$.

Определение 2. Двоичным линейным кодом C называется подпространство размерности k векторного пространства V_r.

Определение 3. Проверочной матрицей двоичного линейного кода C называется такая матрица H, что равенство $Hc^\intercal = 0$ выполняется тогда и только тогда, когда $c \in C$.

Определение 4. Вектор

$$s = Hy^\intercal,$$

где H — проверочная матрица, называется *синдромом* вектора y.

Определение 5. Общая задача синдромного декодирования кода с проверочной матрицей H — задача поиска по вектору s такого вектора e, что $He^\intercal = s$ и $wt(e) \to min$.

Определение 6. Поисковая задача синдромного декодирования $sSD(H, s, t)$ — задача поиска по синдрому $s \in V_r$ такого вектора $e \in V_n$, что $wt(e) = t$ и выполнено равенство $He^\intercal = s^\intercal$.

Данная задача принадлежит к классу NP-трудных. За счёт этого она имеет применение в постквантовой криптографии, например в криптосистеме Мак-Элиса [2]. Наиболее популярные алгоритмы для решения данной задачи основаны на *парадоксе дней рождения*: алгоритм на информационном множестве и алгоритм Штерна. Эти алгоритмы имеют асимптотику, близкую к $O(2^{0.05n})$.

Также эту задачу можно рассматривать как обобщение задачи о рюкзаке на случай векторного пространства:

$$a_1 \cdot b_1 + a_2 \cdot b_2 + ... + a_n \cdot b_n = s,$$

$$wt(a) = t,$$

$$a \in GF(2), b \in V_r$$

Применение к данной задачи методов основанных на генетических алгоритмах и методов отжига не даёт весомых результатов. Одной из проблем, возникающих при их использовании — трудность в выборе метрики для оценки близости к ответу.

Приведём алгоритм Штерна [1]. Введём обозначения: $H_{j\times m}$ — матрица над полем $GF(2)$. t - искомый вес Хемминга. Найти x такой, что $Hx^{\intercal} = 0$.

Шаг 1. Приводим матрицу H к каноническому виду при помощи метода Гаусса. Нумерация при этом сохраняется.

Шаг 2. Разделяем равновероятно оставшиеся столбцы в правой части на множества X и Y.

Шаг 3. Выбираем множество L индексов строк.

Шаг 4. Для каждого p-элементного подмножества A множества X составляем вектор π_A, оставляя только те строки, которые входят в множество L. Аналогичные действия проводим со множеством Y, получая B и π_B.

Шаг 5. Для всех пар A и B таких, что $\pi_A = \pi_B$, вычислить сумму всех векторов из объединения соответствующих множеств A и B. Если вектор V, полученный таким образом, имеет вес $t - 2p$, то тогда возвращаем решение x построенное следующим образом: $x_m = 1$ если $m \in A \cup B$ или для некоторого $i \leq n - k$ верно $u_i = 1$. В противном случае $x_m = 0$.

В ходе работы был реализован алгоритм SD-ISD — вариация алгоритма Штерна [3]. Отличие от оригинального заключается в том, что к матрице H метод Гаусса применяется не до конца, приводя к виду единичной матрицы лишь угловую подматрицу.

Программа написана на языке Python для упрощения изучения алгоритма и влияния на его производительность параметров l и p. Тестовые данные взяты с сайта [4]. Результаты работы приведены в таблице 1.

Размерность матрицы H	Вес t	Время выполнения программы	l	p
(39, 158)	5	5.22	5	2
(39, 158)	5	0.33	10	2
(48, 192)	6	10.37	10	2
(57, 229)	7	0.04	15	1
(57, 229)	7	0.52	15	2
(57, 229)	7	12.51	15	3

Таблица 15: Результаты реализации алгоритма.

Выводы:

В данной работе были рассмотрены вероятностные алгоритмы декодирования линейных кодов, которые в своей асимптотике работают быстрее, чем декодирование по синдрому, декодирование по максимому правдоподобию. В рассмотренных алгоритмах прослеживается общая идея, характерная для самых быстрых алгоритмов, производных от алгоритма Штерна.

ЛИТЕРАТУРА

[1] J. Stern. A method for finding codewords of small weight // Lecture Notes in Computer Science, Springer, 1988, V. 388, P. 106–113

[2] D. J. Bernstein. Attacking and defending the McEliece cryptosystem // Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008

[3] Y. Hamdaoui. A Non Asymptotic Analysis of Information Set Decoding // 2013, IACR Cryptol

[4] https://decodingchallenge.org/goppa

Кураторы исследования –
Чижов И.В., канд. физ.-мат. наук, доцент кафедры ИБ ВМК МГУ;
Коломеец Н.А., канд. физ.-мат. наук, ст. преп. кафедры теоретической кибернетики НГУ.

Задача эквивалентности линейных кодов

А. А. Бучнев[1]
[1]Университет Иннополис, г. Иннополис
E-mail: a.buchnev@innopolis.university

Аннотация

В рамках данной работы представлена реализация алгоритма решения задачи эквивалентности двоичных линейных кодов. Используется алгоритм, основа которого заключается в нахождении такого инварианта кода, что при перестановке π координат кодовых слов, полученный инвариант будет отличаться на ту же перестановку π.

Ключевые слова: *теория кодирования, линейные коды, эквивалентность линейных кодов.*

Введение

Задачу эквивалентности линейных кодов можно сформулировать следующим образом: "Являются ли два данных линейных кода эквивалентными?". Решением задачи является ответ, эквивалентны ли линейные коды, а также соответствующие преобразования, позволяющие из одного кода получить другой. Введём необходимые определения для решения данной задачи.

Определение 1. *Двоичным линейным кодом* длины n размерности k называется подпространство C размерности k пространства $\mathbb{F}_2^n$. Элементы C называются *кодовыми словами*. Весом двоичного вектора t длины n называется следующая величина: $wt(t) = \#\{i : t_i = 1, i \in \{1 \dots n\}\}$, где t_i означает i-тую координату вектора t.

Любое кодовое слово, полученное из сообщения m предствимо в виде: $c = mG, c \in C, m \in \mathbb{F}_2^k$, где *порождающая матрица* G является матрицей полного ранга, строки которой называются базисом кода C.

Заметим, что матрица G однозначно определяет код C, потому что C является множеством всех линейных комбинаций векторов-строк матрицы G. Два линейных кода называются эквивалентными, если кодовые слова одного кода получаются перестановкой координат из кодовых слов другого кода.

Для удобства работы, переформулируем задачу эквивалентности линейных кодов в матричной форме: «Являются ли два кода, порождённых матрицами G и G', эквивалентными?». Соответственно и решение можно переформулировать в матричной форме: найти два таких линейных преобразования P и S, где P — перестановочная матрица размерности $n \times n$, а S — невырожденная матрица размерности $k \times k$, удовлетворяющие следующему равенству:

$$G = SG'P \tag{15}$$

Авторы [1] доказали, что если $P \neq NP$, то задача эквивалентности линейных кодов не является NP-полной, но в то же время они показали, что эта задача является такой же сложной, как и задача изоморфизма графов.

В рамках данного проекта мы будем решать задачу эквивалентности двоичных линейных кодов с помощью «The support splitting algorithm» алгоритма, описанного в [2]. Суть данного алгоритма

заключается в нахождении такого инварианта линейного кода, что при сравнении двух инвариантов эквивалентных линейных кодов, эти инварианты будут отличаться лишь перестановкой π соответсвующих координат, из которой можно легко восстановить перестановку, осуществляемой матрицей P.

Описание работы алгоритма

В [2] предлагается один из способов нахождения вышеупомянутого инварианта через поочерёдное выкалывание координат порождающей матрицы и последующего нахождения весового спектра пересечения кода и дуального ему. Н. Сендрие утверждает, что перестановка между двумя спектрами задаёт перестановку между эквивалентными кодами, однако эта перестановка может быть неоднозначна, единственность перестановки будет только в случае, когда все подгруппы полученной группы перестановок тривиальны. Хоть и полученная перестановка может неоднозначно задавать перестановку между кодами, результат работы алгоритма может существенно сузить перебор вариантов перестановок.

Результаты работы

Было принято решение использовать предложенный в статье инвариант, который находится через подсчёт весового спектра кода. Результатом работы является программная реализация алгоритма поиска подгрупп перестановок между двумя случайными эквивалентными кодами на языке программирования C. Программный код публично доступен по следующей ссылке:
`https://github.com/y4cer/linear_code_equivalency`.

Время работы реализации

Тестирование производилось на персональном компьтере с ЦП Intel Core i7-10750H, исходный код был скомпилирован с флагом оптимизации `-Ofast`. Среднее время работы алгоритма для матриц размерности 50×100 составил 0.08 секунд. Текущая реализация алгоритма не допускает увеличение размерности, ограничения текущей реализации подробнее описаны в секции Работа на будущее.

Работа на будущее

В будущем, предлагается оптимизировать работу с памятью и временем, а также процедуру финального восстановления перестановки. На данный момент для хранения одного бита данных используется 8 бит, что является потенциальным местом для оптимизации работы с памятью, также эта оптимизация ускорит время работы алгоритма Гаусса, от корректности работы которого зависит правильность полученных перестановок, более того, существуют эффективные алгоритмы работы с двоичными матрицами, предложенные в [3].

ЛИТЕРАТУРА

[1] Petrank E., Roth R.M. Is code equivalence easy to decide? // IEEE Transactions on Information Theory, vol. 43, no. 5, pp. 1602–1604, 1997.

[2] Sendrier N. The Support Splitting Algorithm // INRIA. Report no. 3637, 1999. https://inria.hal.science/inria-00073037/file/RR-3637.pdf

[3] Çetin K. Koç, Sarath N. Arachchige, A fast algorithm for gaussian elimination over GF(2) and its implementation on the GAPP // Journal of Parallel and Distributed Computing. 1991, vol. 13, no 1, pp. 118-122

Кураторы исследования –

Чижов И.В., канд. физ.-мат. наук, доцент кафедры ИБ ВМК МГУ;

Коломеец Н.А., канд. физ.-мат. наук, ст. преп. кафедры теоретической кибернетики НГУ.

БЛОКЧЕЙН-ТЕХНОЛОГИИ

Анализ и реализация протокола конфиденциальных активов

А. Ф. Хуцаева[1], А. В. Исайчева[2]
[1]Университет ИТМО
[2]БФУ им. Канта
E-mail: afkhutsaeva@itmo.ru, alencha5555555@gmail.com

Аннотация

Технология конфиденциальных активов позволяет скрывать как суммы транзакций, так и типы активов, повышая их конфиденциальность и взаимозаменяемость. Сокрытие переводимых сумм, а точнее доказательство корректности денежного перевода, осуществляется с помощью схем обязательств.

В рамках данного проекта проведено сравнение свойств схем обязательств Педерсена и Эль-Гамаля, построенных на математическом аппарате эллиптических кривых. Программная реализация схем показала разницу в размерах обязательств и незначительную разницу при их вычислении, в двух случаях у схемы Педерсена лучший результат. Как итог, предложен постквантовый вариант схемы Педерсена на изогениях эллиптических кривых.

Ключевые слова: *схема обязательства, конфиденциальные активы, блокчейн.*

Конфиденциальные активы в контексте технологии блокчейн и криптовалют, основанных на блокчейне, могут иметь разное значение для разных аудиторий, а также могут быть чем-то совершенно другим или уникальным в зависимости от варианта использования. Они представляют собой особый тип цифрового актива и наследуют все его свойства, но в отличие от цифровых активов, данный тип сохраняет свою конфиденциальность при передаче или использовании. Таким образом, они имеют ценность, ими можно владеть, но они не имеют физического присутствия и могут существовать только в форме криптографического токена или его производной, которая также является криптографически защищенной, по крайней мере, в предположении дискретной логарифмической задачи [1].

Основой данных активов являются конфиденциальные транзакции, скрывающие сумму перевода, сохраняя при этом способность публичной блокчейн-сети проверять, что записи по учету переводимых средств и неизрасходованные выходные данные транзакций (UTXO) по-прежнему совпадают.

Для обеспечения конфиденциальности и подтверждения целостности данных используется схема обязательств. Криптографическая схема обязательств является способом предоставления доказательства размера переводимой суммы без раскрытия ее величины. Например, пользователь Алиса создает обязательство к сумме транзакции, отправляя в блокчейн хэш данной суммы. Благодаря этому, наблюдатели не могут узнать точную сумму, но могут проверить, что она соответствует обязательству. Позднее, после проведения транзакции, Алиса раскрывает исходную сумму, обеспечивая доказательство правильности операции.

Теоретические сведения

Определение 1. Схема обязательств - криптографический примитив, позволяющий стороне P сделать предположение m о некотором событии без разглашения какой-либо информации о m, с возможностью в дальнейшем раскрыть m.

Иначе схему обязательств можно задать как набор алгоритмов $Setup, Com, Ver$:

1. $Setup$: на вход подается требуемый уровень безопасности системы, а на выходе выдается публичный параметр ck – ключ обязательства.

2. Com: на основе ключа обязательства, предположения m (сообщения) и некоторого случайно выбранного значения r сторона P генерирует обязательство $C(m, r) = Com(m, r)$.

3. $Open$: когда P рассекречивает свое предположение и значение r, проверяющая сторона V запускает заключительный алгоритм Ver. Алгоритм позволяет проверить верность рассекреченных $m^{'}, r^{'} : Com(m, r) == Com(m^{'}, r^{'})$.

С точки зрения безопасности схема обязательств должна обладать следующими свойствами:

- сокрытие (hiding): обязательство не раскрывает никакой информации о предположении (сообщении).

- связывание (binding): сгенерировать одно и то же обязательство для двух разных предположений (сообщений) - трудная задача.

Таким образом, благодаря вышеописанным свойствам обеспечивается безопасность для двух сторон. Так, доказывающая сторона P уверена, что проверяющая сторона V преждевременно не получит какую-либо информацию о m, анализируя обязательство $C(m, r)$. В свою очередь сторона V уверена, что P не сможет подделать свое предположение после публикации $C(m, r)$.

Для этих двух свойств могут быть определены различные уровни безопасности. Двумя наиболее важными комбинациями из них являются:

- идеальное связывание (perfect binding) и вычислительное сокрытие (computationally hiding);

- вычислительное связывание (computationally binding) и идеальное сокрытие (pefrect hiding).

Вычислительное сокрытие или связывание - свойство обеспечивается тем фактом, что лежащая в основе схемы математическая задача слишком сложна для решения с использованием существующих вычислительных мощностей за разумное время (т.е. сегодня ее невозможно взломать, поскольку вычислительные ресурсы ограничены).

Идеальное сокрытие или связывание - свойство обеспечивается тем фактом, что при решении задачи, лежащей в основе схемы, свойство все равно не нарушается.(т.е. при достаточных вычислительных ресурсах свойство будет сохранено).

Схема Педерсена

Построим схему Педерсена [2] с использованием математического аппарата эллиптических кривых.

- Setup: $\mathbb{G} :< G >, H = h(\alpha)$, α - дополнительный параметр;

- Com(m, r): $C(m, r) = mH + rG$;

- Open($m^{'}, r^{'}, C(m, r)$): $true$, если $C(m, r) = m^{'}H + r^{'}G$, иначе $false$.

Пусть имеется группа точек эллиптической кривой над конечным полем E/F_p, G – генераторная точка кривой. Выберем $\alpha \in_R \mathbb{Z}_q$, где q - порядок генераторной точки, и построим хэш-функцию H как: $H = h(\alpha) = \mathrm{SHA256}(\alpha) * G$.

Такое обязательство в дальнейшем может быть раскрыто, если будут опубликованы параметры m и r. Схема реализует классическую схему Педерсена с единственным отличием в том, что в качестве группы используется группа точек эллиптической кривой.

Обязательство Педерсена обладает свойством *аддитивности*:

$$C(a+b) = C(a) + C(b),$$

где $C(a)$ и $C(b)$ - обязательства по сумме a и b.
Таким образом любой может доказать, например, что входы (сумма, которую мы отправляем) равны выходам (сумма, которую получает принимающая сторона), не раскрывая имеющейся у него суммы. Для простоты сумма комиссии не учитывается.

Схема Эль-Гамаля

Построим схему Эль-Гамаля [3] с сиспользованием математического аппарата эллиптических кривых.

- Setup: $\mathbb{G} :< G >$, $H = h(\alpha)$, α - дополнительный параметр;
- Com(m, r): $C(m, r) = (rG, mrH)$
- Open($m^{'}, r^{'}, C(m, r)$): $true$, если $C(m, r) = (r^{'}G, m^{'}r^{'}H)$, иначе $false$

Пусть имеется группа точек эллиптической кривой над конечным полем E/F_p, G – генераторная точка кривой. Выберем $\alpha \in_R \mathbb{Z}_q$, где q - порядок генераторной точки, и построим хэш-функцию H как: $H = h(\alpha) = \mathrm{SHA256}(\alpha) * G$.

Такое обязательство в дальнейшем может быть раскрыто, если будут опубликованы параметры m и r.

Доказательство диапазона

Одной из проблем аддитивных обязательств является то, что если у нас есть обязательства $C(a_1), C(a_2), C(b_1)$ и $C(b_2)$, и мы намерены использовать их с целью доказательства, что $(a_1 + a_2) - (b_1 + b_2) = 0$, то эти обязательства будут применяться, только если одно из значений в уравнении будет отрицательным.

Например, у нас может быть $a_1 = 6, a_2 = 5, b_1 = 21$ и $b_2 = -10$, получим:

$$(6 + 5) - (21 + (-10)) = 0$$

Мы могли бы сохранить 21 выход и отбросить -10 выход, эффективно создав таким образом на 10 монет больше, чем вложили.

Решением данной проблемы является доказательство каждой суммы выхода в определённом диапазоне при помощи схемы колцевых подписей Борромео [5].

Сравнение схем обязательств

Проведем сравнение рассмотренных схем обязательств.

Безопасность

Можно заметить, что схема обязательств Педерсена обладает *идеальным сокрытием* и *вычислительным связыванием*. Поскольку обязательство строится как $C(m, r) = mH + rG$, то в случае, когда задача дискретного логарифмирования на эллиптической кривой будет решаться за полиномиальное время, определить, какое именно значение фиксировалось в обязательстве, невозможно.

Другими словами, может существовать набор значений $\{m_i^{'}, r_i^{'}\}$, удовлетворяющий обязательству $C(m, r)$, что подтверждает свойство идеального сокрытия. Согласно с данным утверждением можно заметить, что схема Педерсена действительно является вычислительно связанной, то есть можно сгенерировать одно и то же обязательство для двух разных значений $\{m_i^{'}, r_i^{'}\}$.

Рассматривая схему Эль-Гамаля, можно увидеть обратную ситуацию. Данная схема обладает *вычислительным сокрытием* и *идеальным связыванием*. Так, при построении обязательства высчитывается два значения rG, mrH, в случае решения задачи дискретного логарифмирования можно однозначно определить r, а значит и вычислить предположение m. Следовательно, выполняется свойство вычислительного сокрытия. Тогда выполняется идеальное связывание, то есть невозможно сгенерировать одно и то же обязательство для двух разных предположений.

Говоря о применимости схем, стоит отметить, что в зависимости от поставленных задач наиболее оптимальным вариантом является одна из рассмотренных схем. В рамках конфиденциальных активов наиболее приоритетным свойством является идеальное скорытие, позволяющее владельцу актива сохранить в тайне информацию о передаваемых данных (средствах).

Данные схемы обязательств строятся на сложности решения задачи дискретного логарифмирования. Очевидно, что с появлением квантового компьютера с помощью алгоритма Шора можно будет решить данную задачу. Получается, что предложенные протоколы со временем перестанут быть актуальными. Следовательно, необходимо построение схем обязательств на задачах устойчивых к атакам на квантовом компьютере.

Программная реалиазция

При программной реализации схем обязательств Педерсена и Эль-Гамаля с помощью SageMath использовалась кривая *secp256k1*. Так, размеры обязательств Педерсена и Эль-Гамаля составили 242 и 327 байт соответственно (при сообщении m длиной 77 символов). Разница во времени работы двух обязательств составляет доли секунд, однако при подсчете ряда обязательств интервал увеличивается.

Предлагаемое решение

Как отмечалось ранее, в настоящее время актуально построение схем обязательств на задачах устойчивых к атакам на квантовом и классическом компьютере. Одной из таких задач на данный момент является задача поиска изогений эллиптических кривых, а именно сложность обращения задачи группового действия (Group action inverse problem).

Определение 2. Имеются две эллиптические кривые E_1, E_2 с определенными параметрами, необходимо найти такой идеал $\mathfrak{a}$, что $E_2 = [\mathfrak{a}]E_1$.

Более подробное описание данной задачи и детальное пояснение формул можно найти в [5]. Считается, что данная задача является трудно вычислимой. Так, аналогично структуре схемы Педерсена на эллиптических кривых можно построить схему Педерсена на математическом аппарате изогений эллиптических кривых. Тогда получаем, что

- Setup: $\mathfrak{g}, \mathfrak{h} = h(\alpha), \alpha$ - дополнительный параметр
- Com(m,r): $C(m, r) = [\mathfrak{g}^m][\mathfrak{h}^r]E_0$
- Open(m,r,C):$true$, если $C = [\mathfrak{g}^m][\mathfrak{h}^r]E_0$

В данном случае $\mathfrak{h} = h(\alpha) = \mathfrak{g}^{\mathrm{SHA256}(\alpha)}$, где $\alpha \in_R \mathbb{Z}_q, q-$ порядок группы классов идеалов.

Такое обязательство в дальнейшем может быть раскрыто, если будут опубликованы параметры m и r. Схема реализует классическую схему Педерсена с единственным отличием в том, что построена на ином математическом аппарате. Можно заметить, что предложенная схема также

обладает идеальным сокрытием и вычислительным связыванием. Аналогично можно построить схему Эль-Гамаля на изогениях эллиптических кривых.

Заключение

Схемы обязательств Педерсена и Эль-Гамаля обладают разными свойствами, что обеспечивает вариативность их выбора в зависимости от поставленной задачи. Так, в случае, когда для пользователя приоритетом является сохранение в секрете своего предположения, наиболее подходящим решением является схема Педерсена. Схема Эль-Гамаля находит свое применение, когда для пользователя важно сохранить однозначность предположения.

Программная реализация показала, что при малых значениях скорость работы схем почти не отличается. Однако размер передаваемых значений разнится. В данном случае схема Педерсена обладает малыми размерами данных. Данный факт заметен и по структуре схемы, так как обязательство Эль-Гамаля содержит в себе два вычисляемых значения, что компенсируется одним из его свойств - идеальное связывание.

Также предложена постквантовая схема обязательств на изогениях эллиптических кривых. Достоинством предложенной схемы является устойчивость к атакам с помощью алгоритма Шора, но требуются большие вычислительные мощности для подсчета изогений.

ЛИТЕРАТУРА

[1] Tari Labs University. Confidential Assets, 2023. https://tlu.tarilabs.com/digital-assets/confidential-assets

[2] Pedersen T. P. Non-interactive and information-theoretic secure verifiable secret sharing //Advances in Cryptology—CRYPTO'91: Proceedings. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2001. – С. 129-140.

[3] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms //IEEE transactions on information theory. – 1985. – Т. 31. – №. 4. – С. 469-472.

[4] Beullens W., Kleinjung T., Vercauteren F. CSI-FiSh: efficient isogeny based signatures through class group computations //International Conference on the Theory and Application of Cryptology and Information Security. – Cham : Springer International Publishing, 2019. – С. 227-247.

[5] Poelstra A. et al. Confidential assets //International Conference on Financial Cryptography and Data Security. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2018. – С. 43-63.

Кураторы исследования –
Мельничук Е.М., ассистент Института физико-математических наук и информационных технологий, БФУ им. Канта;

Обеспечение приватности данных в смарт-контрактах

Р. С. Кругляков[1], С. И. Ларионова[1], Е. Д. Фролов[2],В. А. Башкиров[2],Р. Т. Еремеев[3],В. А. Далевич[4],
[1]Санкт-Петербургский государственный университет аэрокосмического приборостроения
[2]Балтийский федеральный университет имени Иммануила Канта
[3]Томский Государственный Университет
[4]Новосибирский государственный университет
E-mail: 2121roma21@gmail.com, sofia.larion@yandex.ru, casually_unbolted753@8Shield.net, justmarvinnn@gmail.com, ruslan.eremeev.2016@gmail.com, vlad.dalevich@bk.ru

Аннотация

В работе рассмотрены вопросы конфиденциальности данных в контексте технологии блокчейн и смарт-контрактов. Защита информации в смарт-контрактах является необходимым условием для успешного использования технологии в различных сферах деятельности. Для этого мы предлагаем использовать протокол zk-SNARK и инструмент ZoKrates для реализации приватности данных в смарт-контрактах на блокчейн-платформе Ethereum. В статье описываются задачи, которые нужно выполнить для реализации данного решения, а также приводятся основные принципы работы протокола zk-SNARK и инструмента ZoKrates.

Ключевые слова: *блокчейн, приватность, Ethereum, Solidity, смарт-контракты, zk-SNARK, ZoKrates*

С развитием технологий блокчейн и смарт-контрактов возникает всё больше вопросов о конфиденциальности данных. Приватность является одним из основных прав каждого человека. В контексте блокчейн приватность данных становится ещё более важной, так как все транзакции и данные хранятся в открытом доступе. Это может привести к утечке финасновых, персональных и других критически важных данных. Поэтому защита инфомации в смарт-контактах является необходимым условием для успешного использования технолгии в различных сферах деятельности.
Цель: Реализовать приватность данных в смарт-контрактах
Задачи:

1. Получить опыт взаимодействия с блокчейн платформой Ethereum и с тестовой сетью Sepolia.
2. Изучить базовые возможности языка программирования Solidity, написать простой смарт-контракт.
3. Изучить протокол zk-SNARK.
4. Познакомиться с инструментом ZoKrates.
5. С помощью ZoKrates реализовать конкретную схему на базе протокола zk-SNARK.
6. Реализованный протокол запустить в тестовой сети Sepolia.

Ethereum

Блокчейн - это распределенная база данных, которая хранит информацию о транзакциях в виде блоков, связанных между собой в цепочку. Ethereum – это блокчейн-платформа со встроенной виртуальной машиной и поддержкой смарт-контрактов [1]. Ethereum предоставляет реализацию операций для проверки доказательства zk-SNARK в виде предварительно скомпилированных контрактов. С их помощью есть возможность реализовать схемы на основе доказательства с нулевым разглашением в коде смарт-контрактов.

В Ethereum одна основная и несколько тестовых сетей, которые предназначены для тестирования своих децентрализованных приложений и смарт-контрактов перед их развертыванием в основной сети Ethereum mainnet.

Мы выбрали Sepolia TestNet, так как она наиболее поддерживаемая в экосистеме Ethereum.

Zk-SNARK

Zk-SNARK - протокол доказательства с нулевым разглашением. В протоколе есть доказывающая и проверяющая стороны, а также программа C, которая принимает два вида данных: публичные, приватные. Программа возвращает true, если данные удовлетворяют определенным ограничениям, реализованным в C, и false - иначе.

Протокол zk-SNARK состоит из трёх алгоритмов:

1. G (Generator) принимает на вход программу C и параметр λ (secret parameter), работает на доверенной стороне. Возвращает vk (ключ верификации) и pk (ключ для доказательства).

2. P (Proof) принимает на вход публичные и приватные данные и возвращает доказательство, которое может быть проверено с помощью алгоритма V.

3. V (Verify) приниамет на вход публичные данные и доказательство и возвращает true, если доказательство корректно (владелец обладает приватными данными) и false в противном случае.

У протокола существуют следующие проблемы:

1. Как выбрать доверенную сторону, которая будет запускать алгоритм G и генерировать ключи pk и vk.

2. Как сохранить приватность параметра безопасности λ.

ZoKrates

Создание протокола zk-SNARK напрямую на Solidity может быть сложным и неэффективным процессом, так как Solidity не является языком программирования, специализированным для работы с доказательством с нулевым разглашением. Инструмент Zokrates [3][4], напротив, был специально разработан для работы с доказательствами знаний и обеспечивает более эффективный и удобный способ создания и проверки доказательств знаний в блокчейн-приложениях.

Описание решения

В рамках работы была рассмотрена задача доказательства знания элемента множества без раскрытия его значения. Допустим, у нас есть компания, которая разрабатывает программное обеспечение, и пользователь, который купил лицензию. Задача клиента - доказать, что он владеет лицензией без разглашения ключа активации.

У компании есть множество ключей, для которых вычислено значение хэш-функций. Эти значения записаны в блокчейн с помощью смарт-контракта. Для доказательства владения ключом пользователь отправляет в программу свой ключ.

Данная программа представляет собой комплексную систему, включающую в себя следующие компоненты:

1. Пользовательский интерфейс (UI), разработанный на языке программирования C# с использованием библиотеки Nethereum.web3. Этот интерфейс обеспечивает взаимодействие пользователя с другими частями системы, предоставляя удобный способ ввода данных и отображения результатов.

2. Смарт-контракт, реализованный на языке программирования Solidity [2]. Этот контракт выполняет роль децентрализованного хранилища хэшей приватных ключей, используемых в системе. Он обеспечивает сохранность данных благодаря технологии блокчейн, позволяя пользователям сохранять хэши своих ключей в безопасной и надежной среде. Кроме того, смарт-контракт также обеспечивает сохранение доказательств с нулевым разглашением (ZKP), сгенерированных пользователями при активации лицензий. После успешной генерации доказательства пользователям необходимо загрузить эти доказательства в смарт-контракт. Это действие создает транзакцию в блокчейне, которая подтверждает успешную активацию лицензии на основе предоставленных данных (в случае знания ключа активации).

3. Генератор доказательств zk-SNARK разработан с использованием инструмента ZoKrates. Этот генератор позволяет пользователям создавать доказательства с нулевым разглашением. В данном случае, пользователь подает свой ключ продукта, и с помощью массива хэшей, полученных из блокчейна, генерируется доказательство, подтверждающее право пользователя на активацию лицензии. После этого доказательство может быть выгружено в блокчейн для последующей проверки.

Цель данной программы заключается в обеспечении безопасного и надежного способа активации лицензий для продуктов. Пользовательский ключ продукта хранится в безопасной среде, хэши ключей сохраняются на блокчейне для обеспечения прозрачности и надежности, а использование технологии доказательства с нулевым разглашением позволяет подтвердить право пользователя на активацию лицензии без раскрытия конфиденциальных данных.

Данная система представляет инновационный подход к управлению лицензиями и обеспечению безопасности в сфере программного обеспечения. Преимущества этой системы как для компании, так и для пользователей остаются очевидными и обоснованными.

Для компании, внедрение данной системы позволяет достичь следующих преимуществ:

1. **Противодействие пиратству.** Пиратство в сфере программного обеспечения давно является серьезной проблемой для компаний. За счет сохранения доказательств с нулевым разглашением в блокчейне система предотвращает возможность повторного использования одного и того же доказательства для активации лицензии. Это делает пиратские активации невозможными, так как без уникального доказательства доступ к продукту будет ограничен.

2. **Приватность данных.** Благодаря технологии доказательства с нулевым разглашением, компания не получает доступ к конкретным данным пользователей, таким как ключи продукта. Это обеспечивает высокий уровень приватности и доверия со стороны пользователей. Компания не имеет возможности идентифицировать конкретных пользователей и отзывать лицензии на индивидуальном уровне.

3. **Децентрализация и надежность.** Хранение хэшей ключей в блокчейне снижает риски потери данных и обеспечивает уровень надежности, который может быть сложно достичь в централизованных системах.

С позиции пользователя также имеются значительные преимущества:

1. **Приватность и безопасность.** Пользователи могут быть уверены, что их конфиденциальные данные, такие как приватные ключи, остаются в безопасности. Компания не может получить доступ к их личным данным, что обеспечивает дополнительный уровень приватности и защиты.

2. **Гарантированная лицензия.** Однократное использование доказательства для активации лицензии и его сохранение в блокчейне обеспечивают уверенность пользователя в стабильной и гарантированной работе продукта. Это также защищает от возможных попыток незаконного воспроизводства лицензий.

3. **Прозрачность.** Благодаря использованию блокчейна, процессы активации и проверки лицензий становятся более прозрачными и учетными. Это способствует снижению вероятности споров между пользователями и компанией.

Данная система представляет инновационное решение, объединяющее преимущества децентрализации блокчейна, безопасности хранения данных и приватности доказательств с нулевым разглашением. Такой сбалансированный подход позволяет компаниям эффективно защищать интересы и сохранять лояльность пользователей, обеспечивая им безопасное использование программного обеспечения. Схемы работы представлены на рисунках 1 и 2.

Программа, использующая Zokrates, представляет собой реализацию алгоритма для генерации и проверки доказательств с нулевым разглашением. Данная программа принимает на вход ключ и массив хэшей, представляющих хэши всех ключей, хранящихся в блокчейне.

Процесс работы программы подразумевает следующие шаги:

1. **Инициализация переменных.** Программа создает изменяемую логическую (bool) переменную result, которая будет возвращаться в конце выполнения программы.

2. **Генерация массива нулей.** Создается массив u32[8] zero, который необходим для генерации хэша переданного ключа.

3. **Вычисление хэша ключа.** Программа вычисляет хэш переданного пользователем ключа.

4. **Цикл сравнения хэшей.** Программа в цикле проходит по всем элементам массива хэшей, сравнивая каждый элемент с сгенерированным хэшем ключа. Если хэш ключа совпадает с элементом массива, то в массив bool[4] mutvalues записывается значение true, иначе - false.

5. **Функция для проверки наличия true.** Программа определяет функцию is_true(bool[4] mutdata), которая принимает массив логических значений в качестве аргумента. Эта функция просматривает переданный массив и перемещает все встреченные значения true в конец массива. В итоге, если в массиве было хотя бы одно значение true, функция вернет true, иначе - false.

6. **Возврат результата.** В конце программы, результат функции is_true(values) возвращается в переменную result.

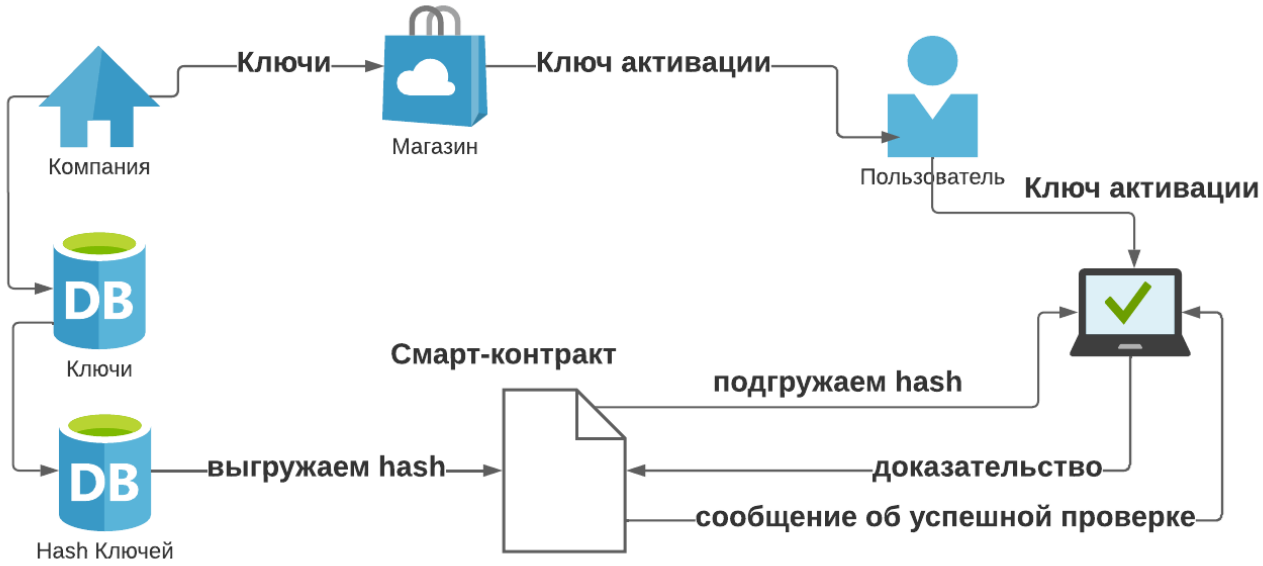


Рис. 1: Структурная схема

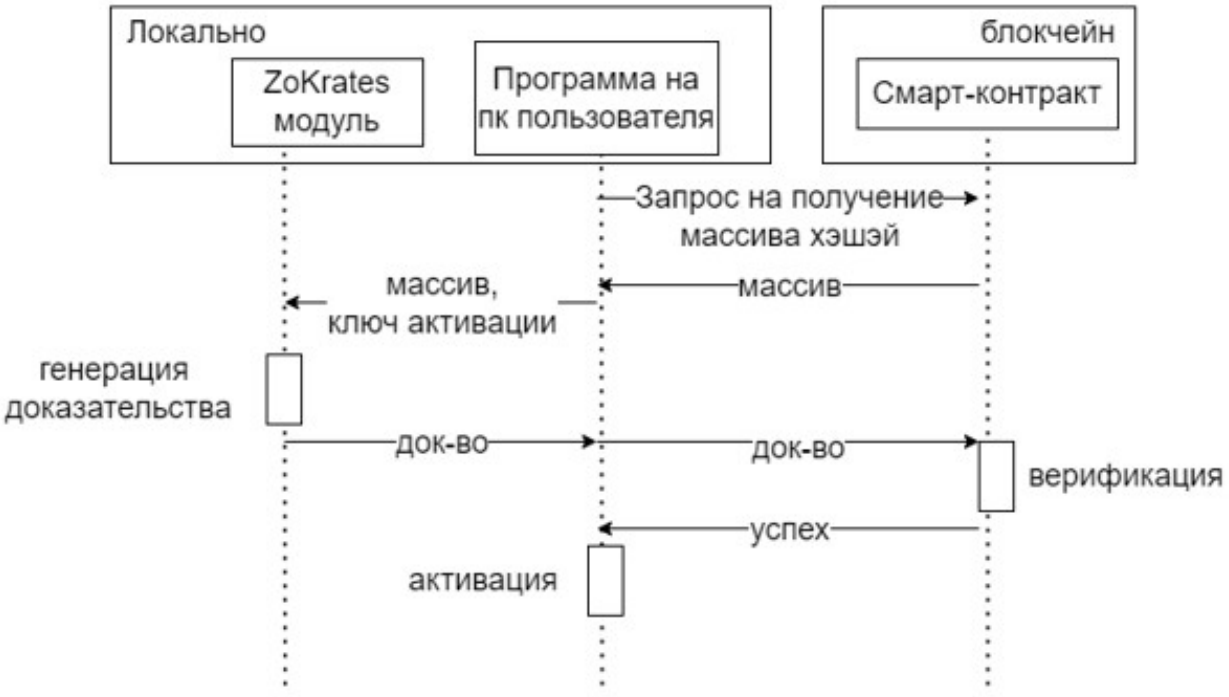


Рис. 2: Диаграмма последовательности генерации и проверки доказательства

ЛИТЕРАТУРА

[1] Ethereum White Paper [Электронный ресурс]. – URL: https://ethereum.org/en/whitepaper/.

[2] Руководство по Solidity [Электронный ресурс]. – URL: https://github.com/ethereum/wiki/wiki/

[3] Zokrates [Электронный ресурс]. – URL:https://github.com/Zokrates/ZoKrates

[4] Документация Zokrates [Электронный ресурс]. – URL:https://zokrates.github.io/introduction.html

Куратор исследования –
Кондырев Д.О. Преподаватель Новосибирского государственного университета

Оптимизация вычисления целочисленного квадратного корня в смарт-контракте пула ликвидности

А. А. Чубань, Н. А. Попов, А. И. Сацута
Балтийский Федеральный Университет имени И. Канта
E-mail: ArtemChuban@yandex.ru, popovnikita01@gmail.com, tolya.satzuta@gmail.com

Аннотация

В данной работе было проведено исследование принципов работы основных алгоритмов маркетмейкеров и создания пулов ликвидности, рассмотрен один из вариантов оптимизации использования газа в смарт-контракте, изучены оптимальные алгоритмы вычисления целочисленного квадратного корня и реализован самый эффективный по трате газа из них. В заключение оптимизированный смарт-контракт был размещен во второй слой блокчейн сети Ethereum протокола zkSync для масштабирования и конфиденциальности.

Ключевые слова: *Blockchain, Ethereum, Decentralized Finance, Liquidity Pool, Smart-Contract, Optimisation, Square root, Zero-knowledge, zkSync.*

Благодаря возникновению децентрализованных финансов возник бурный рост ончейн-аналитики. Объемы финансов в децентрализованных биржах порой могут серьезно превышать объемы централизованных бирж. Вместе с этим рынок продолжает расширяться в результате появления новых видов продуктов.

CEX, DEX и пулы ликвидности

В первые годы появления и развития криптовалют и рынка вокруг них, основными площадками служили централизованные биржи (CEX). Из крупнейших представителей подобного вида бирж - Binance, Huobi, Kraken. Однако с дальнейшим развитием рынка у CEX появились конкуренты в лице децентрализованных бирж - DEX. Главное отличие CEX от DEX заключается в том, что CEX контролируется централизованной стороной, а DEX работает без посредников. На централизованной бирже пользователи должны регистрироваться, проходить верификацию и доверять свои средства третьей стороне, но взамен получать доступ к простому интерфейсу и высоколиквидному рынку. На DEX пользователи могут торговать напрямую друг с другом, используя свои кошельки и без требования делиться своими приватными ключами, но и сами же пользователи выступают ответственными за наполнение резервов биржи, насыщая те самые пулы ликвидности.

Определение 1. Ликвидность - это возможность быстро продать криптовалюту или токен по рыночной или близкой к ней цене.

Определение 2. Пул ликвидности (liquidity pool) - это специальный счет (смарт-контракт), на который любой пользователь может отправить свои средства под блокировку для поддержания уровня резервов биржи взамен на часть комиссионных отчислений платформы.

Без существования таких пулов в принципе невозможно существования децентрализованных финансов - они обеспечивают торговлю, кредитование и много других функций. В работе пулов задействовано несколько ролей:

- **AMM (автоматические маркетмейкеры).** Они представляют собой смарт-контракты, которым отводится автоматизация процесса взаимодействия трейдеров с пулом и корректировки цен обоих токенов с помощью математических формул в зависимости от колебания спроса и предложения.

- **Поставщики ликвидности.** Это участники рынка, которые блокируют свои токены в пулах, обеспечивая тем самым возможность трейдинга для других участников, и получают за это комиссию. Для этого они предоставляют два токена, торговую пару эквивалентной стоимости. Посколькуо поставщиком ликвидности может стать кто угодно, AMM сделали рынок более доступным.

- **Трейдеры.** Благодаря созданному рынку для конкретных пар токенов трейдеры имеют возможность покупать и продавать их, пользуясь ликвидностью. По сути, происходит взаимодействие peer-to-peer.

Для AMM существует 4 основных алгоритма регулирования цены токенов внутри пула, принадлежащие к классу маркетмейкеров с постоянной функцией (CFMM, Constant Function Market Makers).

CSMM (Constant Sum Market Makers)

Маркетмейкер с постоянной суммой представляет собой самую простую реализацию маркетмейкера с постоянной функцией и удовлетворяется следующим уравнением:

$$\sum_{i=1}^{n} R_i = k$$

где R_i - резерв каждого i-го токена, а k - некая константа. И хотя эта функция защищает участников рынка - трейдеров - от возможного проскальзывания, но она не обеспечивает бесконечной ликвидности и поэтому не подходит для чистого её применения при создании пулов ликвидности. График у данного вида функций представляет собой прямую линию в случае использования пары токенов и задается функцией $x + y = k$.

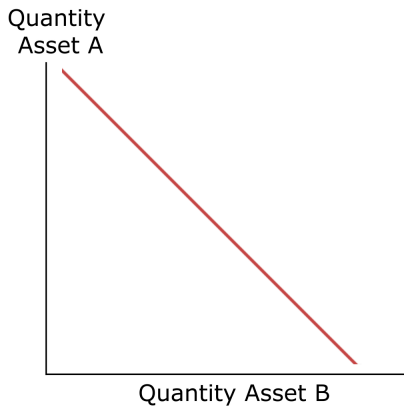


Рис. 1: Прямая для функции с постоянной суммой

CPMM (Constant Product Market Makers)

Маркетмейкер с постоянным произведением для определения рыночной цены пары токенов был впервые релизован в Uniswap (впоследствии изменен начиная с версии v3) и представляет собой следующее уравнение:

$$(R_\alpha - \Delta_\alpha)(R_\beta - \gamma\Delta_\beta) = k$$

где R_α и R_β - резервы каждого токена в паре, а γ - комиссия за транзакцию. Обмен любого количества токенов на другой должен изменить резервы таким образом, чтобы произведение $R_\alpha \cdot R_\beta$ оставалось равным константе k, при учете нулевой комиссии γ. Часто это уравнение упрощается до вида $x \cdot y = k$, где x и y - запасы каждого токена в паре. На практике Uniswap использует комиссию $\gamma = 0.3\%$, что в свою очередь при каждом обмене увеличивает k. Если посмотреть на график для этого вида маркетмейкеров, то мы увидим обычную гиперболу.

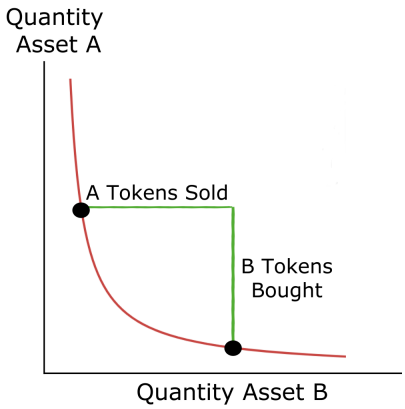


Рис. 2: Кривая для функции с постоянным произведением

CMMM (Constant Mean Market Makers)

Маркетмейкеры с постоянным средним значением является обобщением варианта маркетмейкеров с постоянным произведением, но для использования с более чем двумя активами и их весами за пределами $50/50$. Первым представителем такого варианта AMM является Balancer, где рыночные цены каждого токена удовлетворяют следующему уравнению:

$$\prod_{i=1}^{n} R_i^{W_i} = k$$

где R_i - запасы каждого токена, W_i - вес каждого токена и k - некая константа. Иначе говоря, при отсутствии комиссий такие маркетмейкеры обеспечивают постоянство среднего геометрического взвешенного всех токенов. Например, функция для трех токенов будет иметь вид $(x \cdot y \cdot z)^{\frac{1}{3}} = k$.

Hybrid CFMM (Hybrid Constant Function Market Makers)

Логичным вариантом развития AMM является использование гибридных функций для достижения желаемых свойств в зависимости от торгуемых токенов. Среди представителей такого варианта маркетмейкера наиболее выделяются Curve (Stableswap), Shell Protocol. В случае со Stableswap сложная функция действует как функция с постоянной суммой, когда запасы токенов сбалансированы, и смещается в сторону функции с постоянным произведением, когда запасы становятся несбалансированые.

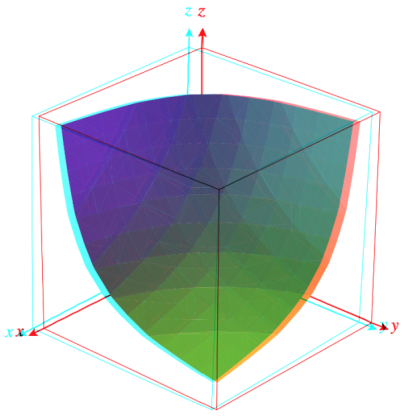


Рис. 3: Вид функции для трех токенов [4]

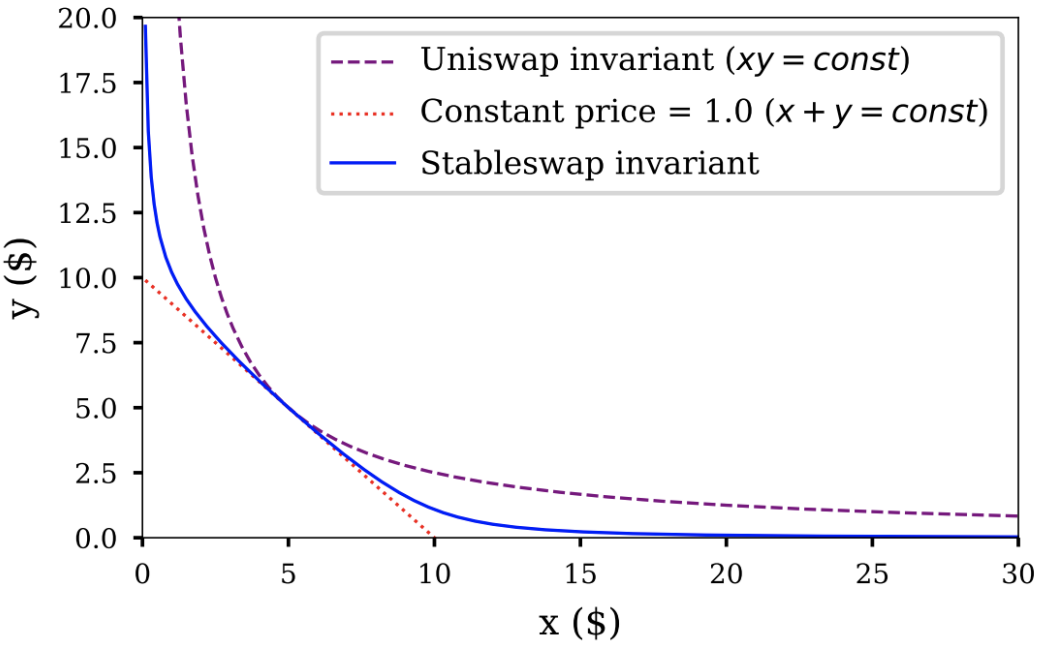


Рис. 4: Сравнение графиков Uniswap (CPMM), CSMM и Stableswap (Hybrid CFMM) [5]

Оптимизация алгоритма нахождения целочисленного корня

Функции чеканки [10] $\_mintFee$ и $mint$ смарт-контракта пула ликвидности, реализованного Uniswap, используют в своей реализации функцию нахождения целочисленного квадратного корня 256-битного числа. Функция $mint$ вызывает вспомогательную функцию $sqrt$, а также функцию $\_mintFee$, которая вызывает функцию $sqrt$ дважды, при ненулевом значении переменной $mintTo$. Следовательно, оптимизация алгоритма нахождения целочисленного квадратного корня уменьшит количество используемого газа при транзакциях при каждом вызове функции $sqrt$.

Вавилонский метод нахождения квадратного корня

Исходная функция $sqrt$ работает по Вавилонскому методу [11] нахождения квадратного корня, также известному как метод Герона. Основная методика заключается в том, что если x больше квадратного корня неотрицательного целого числа S, то $\frac{S}{x}$ будет меньше корня, и наоборот. Так что среднее этих двух чисел будет близким к корню, что основывается на неравенстве о сред-

нем арифметическом, геометрическом и гармоническом, согласно которому такое среднее всегда больше квадратного корня, что обеспечивает сходимость.

Если x является начальным приближением для $\sqrt{S}$, а ε - ошибка в оценке, такая что $S = (x+\varepsilon)^2$, то раскрыв скобки мы получим $\varepsilon = \frac{S-x^2}{2x+\varepsilon} \approx \frac{S-x^2}{2x}$, так как $\varepsilon \ll x$. Следовательно, мы можем компенсировать ошибку и обновить оценку $x + \varepsilon \approx x + \frac{S-x^2}{2x} = \frac{S+x^2}{2x} = \frac{\frac{S}{x}+x}{2}$.

Поскольку вычисленная ошибка не точна, она станет следующим приближением. Процесс продолжается до тех пор, пока не будет достигнута необходимая точность.

Алгоритм можно представить следующим образом:

$$x_0 \approx \sqrt{S}$$
$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$$
$$\sqrt{S} = \lim_{x\to\infty} x_n$$

Метод цифра за цифрой

Данный метод [11] последовательно ищет каждую цифру квадратного корня. Для понимания основного принципа рассмотрим случай нахождения квадратного корня из числа Z, которое является квадратом двухначного числа XY, где X - цифра десятков, а Y - цифра единиц. Тогда получается $Z = (10X + Y)^2 = 100X^2 + 20XY + Y^2$. Для начала определим значение цифры десятков. X - наибольшая цифра, такая что X^2 не превосходит Z, от которого отброшены две последние цифры. На следующей итерации соединяем пару цифр, умножая X на 2 и помещая результат в позицию десятков, а затем пытаемся найти значение Y.

Такая же идея может быть распространена на вычисление произвольного квадратного корня. Представим, что мы можем разложить квадратный корень из N в сумму n положительных чисел, таких что $N = (a_1 + a_2 + \cdots + a_n)^2$. Используя тождество $(x + y)^2 = x^2 + 2xy + y^2$, преобразуем правую часть $(a_1+a_2+\cdots+a_n)^2 = a_1^2+2a_1a_2+a_2^2+2(a_1+a_2)a_3+a_3^2+\cdots+a_{n-1}^2+2(\sum_{i=1}^{n-1} a_i)a_n+a_n^2 = a_1^2 + (2a_1 + a_2)a_2 + (2(a_1 + a_2) + a_3)a_3 + \cdots + (2(\sum_{i=1}^{n-1} a_i) + a_n)a_n$

Это выражение позволяет найти квадратный корень последовательным подбором значений a_i. Пусть числа $a_1, \ldots, a_{m-1}$ известны, тогда m-й член задается выражением $Y_m = (2\sum_{i=1}^{m-1} a_i + a_m)a_m$. Теперь каждый новый подбор a_m должен удовлетворять рекурсивному равенству $X_m = X_{m-1} - Y_m$, так что $X_m \geq 0$ для всех $1 \leq m \leq n$, при $X_0 = N$. Если $X_n = 0$, то найден точный квадратный корень, иначе сумма a_i дает аппроксимацию квадратного корня и X_n будет являться ошибкой аппроксимации.

Для двойчной системы счисления можно разложить квадрат искомого числа $N^2 = (a_n + \cdots + a_0)^2$, где каждое $a_m = 2^m$ или 0. После мы проходимся по всем 2^m от 2^n до 2^0 и строим приближенное решение $P_m = a_n + a_{n-1} + \cdots + a_m$ в виде суммы всех a_i, для которых будет найдено значение. Чтобы определить, равно ли a_m значению 2^m или 0, используем $P_m = P_{m+1} + 2^m$. Если $P_m^2 \leq N^2$, то полагаем $a_m = 2^m$, иначе $a_m = 0$ и $P_m = P_{m+1}$.

Для избежания возведения P_m на каждом шаге в квадрат, разность $X_m = N^2 - P_m^2$ сохраняется и обновляется на каждой итерации, полагая $X_m = X_{m+1} - Y_m$ с $Y_m = P_m - P_{m-1} = 2P_{m+1}a_m + a_m^2$. Изначально $a_n = P_n = 2^n$ для наибольшего n с $2^{2n} = 4^n \leq N^2$.

В качестве дополнительной оптимизации сохраняется $P_{m+1}2^{m+1}$ и 2^{2m}, два члена Y_m, когда $a_m \neq 0$:

$$c_m = P_{m+1}2^{m+1}$$
$$d_m = 2^{2m}$$
$$Y_m = \begin{cases} c_m + d_m, & \text{если } a_m = 2^m \\ 0, & \text{если } a_m = 0 \end{cases}$$

c_m и d_m можно эффективно обновлять на каждом шаге:

$$c_{m-1} = P_m 2^m = (P_{m+1} + a_m)2^m = P_{m+1}2^m + a_m 2^m = \begin{cases} \frac{c_m}{2} + d_m, & \text{если } a_m = 2^m \\ \frac{c_m}{2}, & \text{если } a_m = 0 \end{cases}$$

$$d_{m-1} = \frac{d_m}{4}$$

$c_{-1} = P_0 2^0 = P_0 = N$ является конечным результатом.

Преобразование метода цифра за цифрой

Изначальный вариант метода цифра за цифрой создан для вычисления квадратного корня числа произвольной длины, но так как мы работаем с языком программирования Solidity, то знаем ограничение по размеру числа - 256 бит. Благодаря этому есть возможность уйти от итерационного подхода и выполнить ограниченное количество операций независимо от входного числа. Такой подход изменит зависимость количества затраченного газа для вычисления квадратного корня с вида $gas(n) = \mathcal{O}(\log \log n)$ на $gas(n) = \mathcal{O}(C)$, где n - число, корень которого необходимо найти, а C - константа. Ниже приведен график зависимости: по горизонтали отложен размер в битах входного числа функции квадратного корня, а по вертикали - количество затраченного газа для нахождения квадратного корня

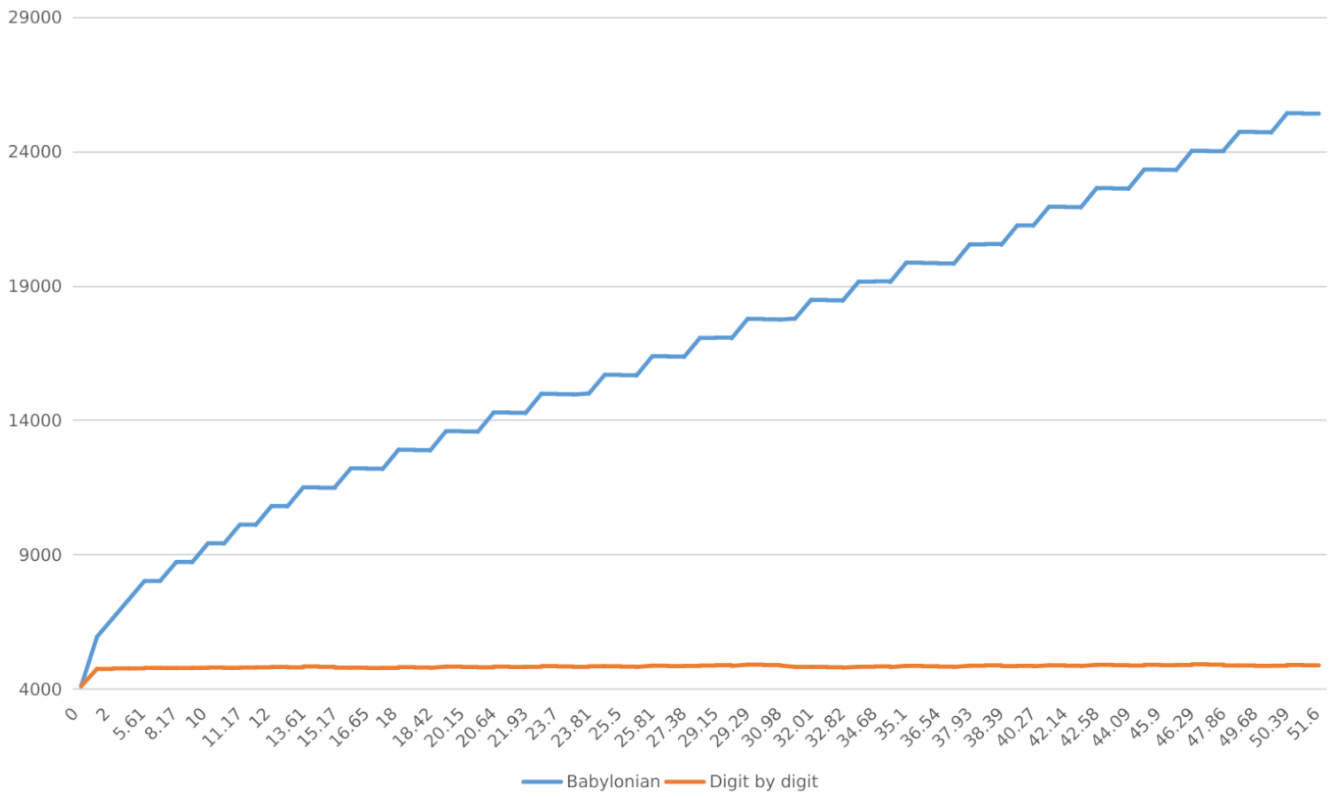


Рис. 5: Зависимость размера входного числа в битах и колиечство используемого газа для вычиления его квадратного корня

Нами было обработано свыше 150.000 транзакций в сети Ethereum смарт-контракта SushiSwap, которые использовали функция $sqrt$. Были вычислены средние значения входного параметра для функции квадратного корня, средняя цена за единицу газа в сети за весь промежуток с момента деплоя смарт-контракта. Итоговое количество ETH, которое можно было бы сэкономить вычисляется

по формуле

$$\sum_{i=1}^{n} gasPrice_i * gasDifference_i * sqrtCallCount$$

где n - количество транзакций, $gasPrice_i$ - цена газа в момент i-той транзакции, $gasDifference_i$ - различие в количестве газа при входном аргументе на i-той транзакции, затраченного старой функцией нахождения квадратного корня и новой, $sqrtCallCount = 2$ - количество вызова функции $sqrt$ в одной транзакции.

Итоговое значение сэкономленного количества ETH: 447.57299412.

zkSync

Сеть zkSync представляет собой сеть второго уровня для Ethereum. Цель данного решения заключается в масштабировании блокчейна, уменьшение комиссий и скорости выполнения транзакций.

Rollups

Сеть zkSync основывается на технологии zero-knowladge rollups, т.е. агрегирует транзакции второго слоя и отправлет их на уровень Ethereum для фиксации. На первый уровень отправляются лишь их доказательства, в отличии от оптимистических свёрток, которые публикуют все данные в сети Ethereum. Так как достоверность транзаций уже указана, нет необходимости ждать пока верификатор убедится в достоверности транзакций. Таким образом, приминениен zkRollups уменьшает цену транзакций, а также скорость их выполнения.

zkSync Era предусматривает наличие гиперцепей, которые являются параррельно работующими zkEVM. Данные гиперцепи связаны общим мостом с первым слоем и гипермостами между собой.

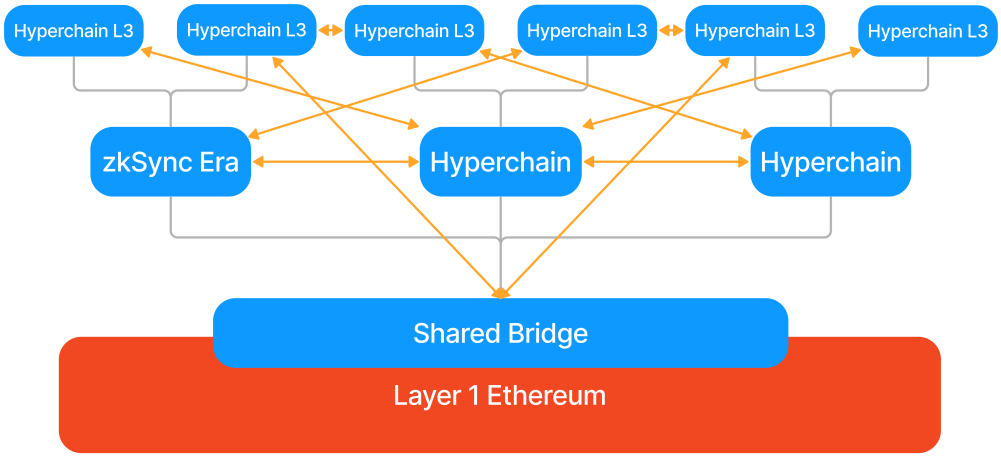


Рис. 6: Гиперцепи, серые линии означают доказательства, оранжевые - гипермосты

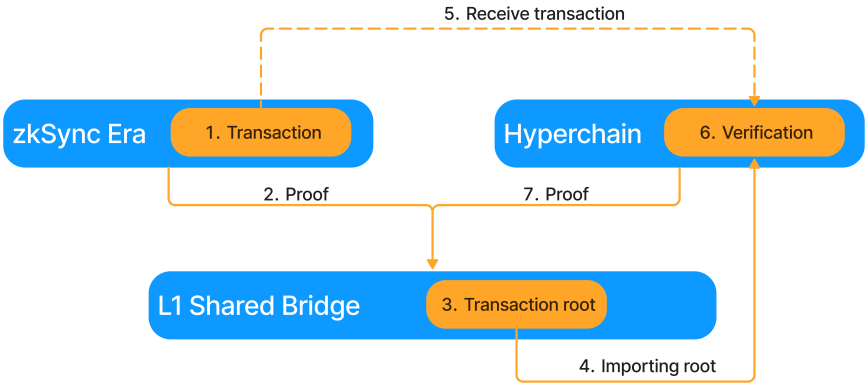


Рис. 7: Порядок выполнения транзакции между гиперцепями

zkEVM

Zero-knowledge Ethereum Virtual Machine(zkEVM) — это виртуальная машина, поддерживающая вычисления с доказательством с нулевым разглашением.

Так как второй уровень являетсz ZK-Rollups, появляется необходимость использовать zkEVM. Всего существует 4 типа виртуальных машин, которые могут использоваться на втором уровне.

1. zkEVM, полностью эквивалентные Ethereum
2. zkEVM, полностью эквивалентные EVM, EVM даннго типа стремятся быть в точности эквивалентными EVM, но не полностью эквивалентными Ethereum.
3. zkEVM, почти эквивалентные EVM, некоторые смарт-контракты не будут поддерживаться
4. zkEVM, эквивалентные языкам высокого уровня, они сопостовимы, например, с Solidity, имеют большую скорость доказательства, но требуют дополнительных действий для получения совместимости с EVM[12].

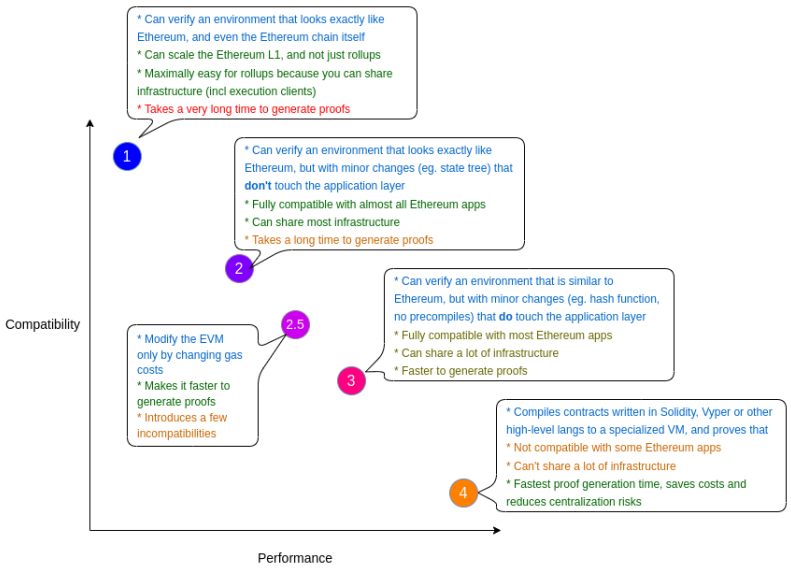


Рис. 8: Классификация zkEVM.

Сеть zkSync использует собственную zkEVM 4-го типа. Uniswap и Sushiswap используют zkSync v2.0 (Era), основное преимущество второй версии является совместимость с EVM, таким образом смарт-контракты во втором слое можно писать на Solidity и Vyper, а миграция уже существующих контрактов не вызывает трудностей. Особенностью аккаунтов в zkSync является их абстракция, т.е. аккаунты zkSync v2.0 могут инициализировать транзакции, как например EOA (внешние учётные записи), и имеют логику, как смарт-контракты в сети Ethereum[13].

Проект был размещён в сеть zkSync Era.

ЛИТЕРАТУРА

[1] Binance Academy, Что такое пулы ликвидности в DeFi и как они работают? (December 2020)

[2] Дилорам Султанходжаева, Как работают пулы ликвидности (Summer 2022)

[3] Dmitriy Berenzon, Constant Function Market Makers: DeFi's "Zero to One" Innovation (April 2020)

[4] Fernando Martinelli Nikolai Mushegian, Balancer Whitepaper - A non-custodial portfolio manager, liquidity provider, and price sensor (September 2019)

[5] Michael Egorov, StableSwap - efficient mechanism for Stablecoin liquidity (November 2019)

[6] Nick Johnson, Euler: The simplest exchange and currency (September 2016)

[7] Vitalik Buterin, Let's run on-chain decentralized exchanges the way we run prediction markets (October 2016)

[8] Vitalik Buterin, Improving front running resistance of x*y=k market makers (March 2018)

[9] Yi Zhang, Xiaohong Chen, Daejun Park, Runtime Verification, Inc., Formal Specification of Constant Product (x*y=k) Market Maker Model and Implementation (October 2018)

[10] Hayden Adams, Noah Zinsmeister, Dan Robinson, Uniswap v2 Whitepaper (March 2020)

[11] Methods of computing square roots (January 2012)

[12] Vitalik Buterin, The different types of ZK-EVMs (August 2022)

[13] Antonio, Dustin Brickwood, niramisa, Technical reference zkSync Era (July 2023)

Кураторы исследования –
Мельничук Евгений Михайлович - ассистент БФУ им. И. Канта.

Реализация системы анонимных платежей на базе сети Ethereum с использованием технологии доказательств с нулевым разглашением

А. А. Блохин[1], В. Д. Боязитов[1], А. С. Евсеев[1], Д. В. Енин[2], А. А. Ерохина[3], Ф. Е. Пивоваров[4], Р. Д. Тогумбаев[1]

[1]Томский государственный университет

[2]Балтийский федеральный институт имени Иммануила Канта

[3]Московский государственный университет имени М.В. Ломоносова

[4]Новосибирский государственный университет

E-mail: blokhinart@mail.ru, colpakov.vadik@yandex.ru, evseevalek5ander@yandex.ru, dani.enin@yandex.ru, erokhina.aa19@physics.msu.ru, f.e.pivovarov@gmail.com, togumbaevr@mail.ru

Аннотация

В данной работе предлагается схема системы анонимных платежей и её реализация на базе сети Ethereum с использованием протокола zkSNARKS. Технология доказательств с нулевым разглашением позволяет пользователям проводить транзакции, не раскрывая детали операций и личной информации, что обеспечивает конфиденциальность и безопасность.

Ключевые слова: *доказательство с нулевым разглашением, Zero-Knowledge Proof, zkSNARK, Ethereum, анонимные платежи*

Введение

Современные технологии блокчейн и криптовалюты внесли революционные изменения в сферу финансов и платежных систем. Однако, существует необходимость в создании системы анонимных платежей, которая обеспечивает конфиденциальность и безопасность транзакций, исключая возможность отслеживания данных о пользователях.

В данной работе представляется исследование и разработка системы анонимных платежей на базе сети Ethereum с использованием технологии доказательств с нулевым разглашением (zero-knowledge proof). Она позволяет подтверждать выполнение определенных условий без раскрытия самой информации, что является ключевым фактором для обеспечения конфиденциальности. На стороне пользователя генерируется доказательство обладания некоторой информацией или подтверждение некоторого факта. Это доказательство затем передается на верификатор через смарт-контракт. Ключи и доказательства используются по одному разу, что также обеспечивает защиту.

В рамках исследования разработана архитектура системы анонимных платежей, которая позволяет пользователям закрыто проводить переводы между собой. Применение технологии доказательств с нулевым разглашением позволяет проверять корректность и достоверность транзакций без раскрытия деталей самой операции.

Схема протокола

Рассмотрим схему протокола. Введём некоторые обозначения:

- (1) – отправитель перевода;
- (2) – получатель;

- $val0$ – сумма изначального пополнения баланса отправителем;
- $val1$ – сумма, оставшаяся у отправителя после перевода;
- $val2$ – сумма перевода получателю;
- 1^* – секретное слово отправителя для баланса перед переводом;
- 1^{**}_{val1} – секретное слово отправителя для вывода $val1$;
- 2^{**}_{val2} – секретное слово получателя для вывода $val2$.

Пополнение

Отправитель (1) придумывает секретное слово 1^* и отправляет на смарт-контракт его хэшированное значение $H(1^*)$ вместе с транзакцией суммой $val0$. На контракте сохраняется хэш конкатенированных хэшей $val0$ и 1^* (см. Рис. 1).

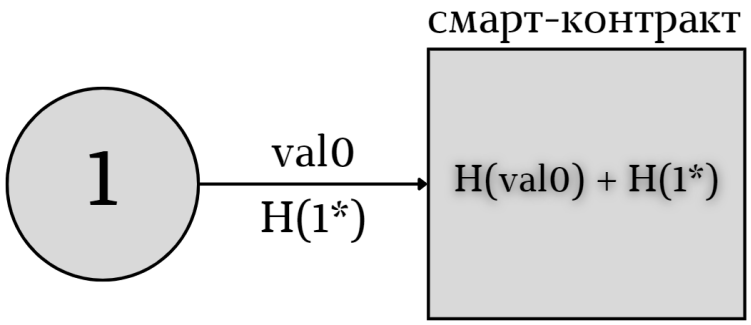


Рис. 1: Пополнение счёта отправителем

Перевод

Отправитель (1) и получатель (2) заранее закрыто договариваются о сумме перевода $val2$. (2) придумывает секретное слово 2^{**}_{val2}, соответствующее этой сумме, а затем отправляет его хэшированное значение (1). (1) также придумывает свое секретное слово 1^{**}_{val1}, соответствующее сумме остатка $val1$ (см. Рис. 2).

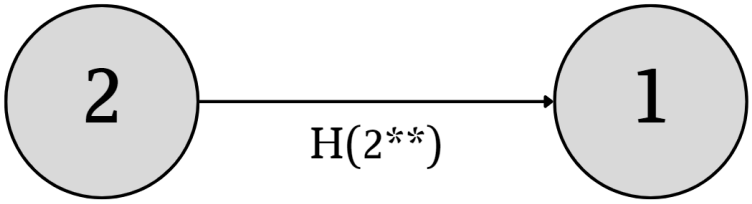


Рис. 2: Передача $H(2^{**}_{val2})$ отправителю получателем

Отправитель (1) посылает на смарт-контракт хэшированные значения следующих величин: своего баланса $val0$, секретного слова 1^*, суммы остатка $val1$, соответствующего ему секретного слова

1^{**}_{val1}, суммы перевода $val2$, соответствующего ему секретного слова 2^{**}_{val2}. Помимо этого (1) через смарт-контракт отправляет на верификатор доказательство с нулевым разглашением знания прообразов сложенных хэшей $val0$ и 1^*, а также доказательства того, что $val1$ и $val2$ положительные величины и их сумма равна $val0$. После подтверждения на смарт-контракте предыдущая запись сложенных хэшей $val0$ и 1^* заменяется на хэши двух других – $H(val1)+H(1^{**}_{val1})$ и $H(val2)+H(2^{**}_{val2})$, то есть хэш конкатенированных хэшей суммы и соответствующего ей ключа (см. Рис. 3).

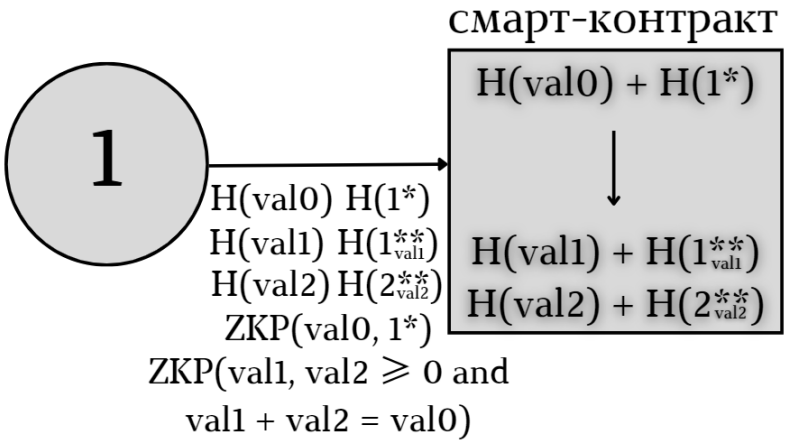


Рис. 3: Разделение счёта на остаток отправителя и перевод получателю

Вывод

Получатель (2) отправляет на смарт-контракт значение $val2$, а также хэшированное значение ключа 2^{**}_{val2} и ZKP знания прообраза $H(2^{**}_{val2})$. После проверки соответствующая запись $H(val2)+H(2^{**}_{val2})$ стирается из памяти контракта, а сумма 2^{**}_{val2} переводится на адрес (2). Для (1) вывод аналогичен, но используются значения $val1$ и 1^{**}_{val1} (см. Рис. 4).

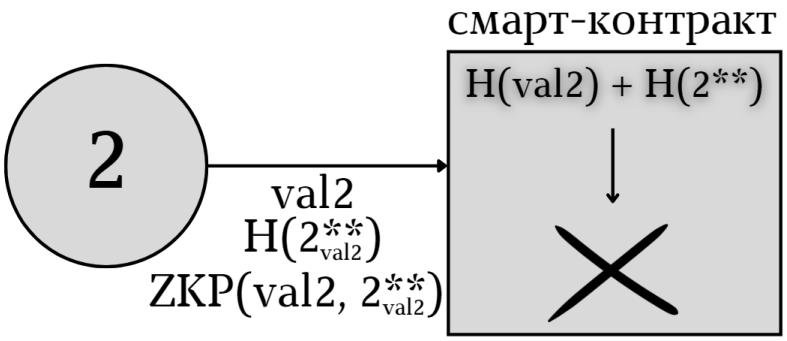


Рис. 4: Вывод денег

Реализация системы

Схема реализована в виде смарт-контракта для сети Ethereum. Программа написана на языке Solidity на платформе Remix - Ethereum IDE. Для создания и верификации доказательств с нулевым разглашением используется утилита ZoKrates, позволяющая автоматизировать написание вычислительного алгоритма и сократить код.

Анонимность обеспечивается за счёт сокрытия переводов внутри контракта. Единственная видимая информация – переводы на контракт и вывод денег с него. Передача ключей между пользователями также закрыта, так как отправляется только хэш, знание прообраза которого подтверждается с помощью доказательства с нулевым разглашением. Уязвимостью является договорённость о сумме транзакции, которую мы считаем секретной и защищенной. Скомпрометированная передача этой информации договоренности между отправителем и получателем о сумме транзакции автоматически раскрывает факт перевода. Для договорённости необходимо использовать другие защищенные каналы.

Заключение

Алгоритм и архитектура системы успешно реализованы с достаточной степенью анонимности, при этом остаётся широкое поле для дальнейших исследований. В первую очередь планируется добавление пользовательского интерфейса и автоматической генерации ключей, встроенной в систему для повышения защищённости. Представленный в работе алгоритм может быть использован для сокрытия платежей с целью защиты информации и предотвращения компрометации отдельных пользователей.

Данное исследование имеет важное значение для развития систем анонимных платежей и обеспечения приватности пользователей. Реализация такой системы может значительно улучшить конфиденциальность и безопасность платежных операций, открывая новые пути для развития и применения криптовалютных технологий. Реализация системы анонимных платежей на базе сети Ethereum с использованием технологии доказательств с нулевым разглашением представляет новые возможности для безопасных и конфиденциальных финансовых операций.

ЛИТЕРАТУРА

[1] F. Fang, Z. Zhou and D. Summers. Confer (Confidential Transactions) // ETH San Francisco 2018. URL: https://devpost.com/software/confer-on-chain-confidential-transactions

[2] Chen T. et al. A review of zk-snarks //arXiv preprint arXiv:2202.06877. – 2022.

[3] Guan Z. et al. BlockMaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs //IEEE Transactions on Dependable and Secure Computing. – 2020. – Т. 19. – №. 3. – С. 1446-1463.

[4] ZoKrates: Introduction // GitHub Pages. URL:https://zokrates.github.io/

Куратор исследования –
ассистент Института физико-математических наук и информационных технологий БФУ им. Канта, Мельничук Евгений Михайлович

Решение проблемы масштабируемости блокчейна с помощью ZK-rollups

А.М. Мамаджанова[1], Я.Е. Дрожжина[1], А.С. Шапоренко[2]
[1]БФУ им. Канта
[2]НИ ТГУ
E-mail: mamadzhanovamadina@gmail.com, 1yana1yt@gmail.com, andrew230820@gmail.com

Аннотация

В работе рассматривается ZK-Rollups как один из возможных способов решения проблемы масштабирования блокчейна. Были рассмотрены наиболее популярные реализации технологии ZK-Rollups, проведён анализ и составлена сравнительная характеристика. На платформе Ethereum было разработано блокчейн-приложение, которое решает проблему масштабирования для задачи перевода токенов, используя наиболее подходящую реализацию ZK-Rollups в виде ZK-Sync.

Ключевые слова: *Ethereum, блокчейн, ZK-Rollups, ZK-Sync, масштабирование, смарт-контракт, транзакция.*

Актуальность

Одной из самых серьезных проблем различных блокчейнов становится проблема масштабируемости. Так, размер основной сети Ethereum уже превышает 1 Тб. Одним из перспективных исследований, направленных на решение проблемы масштабируемости, являются свертки с нулевым разглашением (ZK-Rollups).

Цели и задачи проекта

1. Изучить платформу Ethereum.
2. Исследовать реализации ZK-Rollups. Составить сравнительную характеристику.
3. Разработать смарт-контракты на языке Solidity.
4. Создать приложение с использованием технологии блокчейн.
5. Решить проблему масштабирования с помощью технологии ZK-Rollups.

Ethereum - криптовалюта и платформа для создания децентрализованных онлайн-сервисов на базе блокчейна, работающих на базе смарт-контрактов. Реализована как единая децентрализованная виртуальная машина. Концепт был предложен Виталиком Бутериным в конце 2013 года, сеть была запущена 30 июля 2015 года[1]. В качестве основной платформы для работы над проектом мы выбрали Ethereum, потому что это самая популярная платформа среди пользователей, которая позволяет писать смарт-контракты с расширенными возможностями.

Терминология

Блокчейн - это база данных с транзакциями, состоящая из последовательно выстроенной цепочки цифровых блоков, в каждом из которых хранится информация о предыдущем и следующем блоках [2].

Смарт-котракт - это компьютерная программа, которая отслеживает и обеспечивает исполнение обязательств. Стороны прописывают в нем условия сделки и санкции за их невыполнение, после чего заверяют с помощью цифровой подписи. Смарт-контракт самостоятельно определяет,

выполнены ли условия договора, и принимает решение о дальнейших действиях: завершить сделку и выдать требуемое (деньги, акции, недвижимость), наложить на участников штраф или пеню, закрыть доступ к активам и т.д. [3].

Свертки с нулевым разглашением (ZK-Rollups) - объединяют (или "сворачивают") транзакции в пакеты, которые выполняются вне цепочки. Вычисления вне основного блокчейна уменьшают объем данных, которые должны быть размещены в блокчейне. Операторы ZK-Rollup представляют сводку изменений, необходимых для представления всех транзакций в пакете, а не отправляют каждую транзакцию по отдельности [4].

Транзакция — операция сохранения данных в блокчейне, в ходе которой происходит передача криптоактивов или другой информации между кошельками. Отправка транзакции происходит после её создания в кошельке и подписания цифровой подписью на основе закрытого ключа [5].

Сравнительная характеристика реализаций ZK-Rollups

Для выбора наиболее подходящей реализации были изучены и проанализированы различные реализации ZK-Rollups: Loopring, ZK-Synk, ZKSpace, Aztek [6, 7, 8, 9]. В результате была составлена таблица со сравнительными характеристиками. Основными рассматриваемыми критериями были: возможности инструмента, типы транзакций, размер комиссии, пропускная способность, а также способы вывода средств. Данные критерии очень важны для пользователей. Во-первых, размер комиссии за транзакцию играет ключевую роль при разработке децентрализованных приложений - низкая комиссия привлекает больше пользователей, тем самым делая реализацию доступной. Во-вторых, возможности инструмента, поддерживаемые типы транзакций и пропускная способность характеризуют разнообразие функциональности, которую можно реализовать, и скорость соответственно.

Возможности	Ограничения			
	Транзакции	**Комиссия за 1 транзакцию**	**Пропускная способность**	**Вывод средств**
Loopring				
1)Повышает пропускную способность протокола. **2)**Не возникает риска потери средств. Не требуют доверия и безопасны. **3)**Передача токена вне цепочки требует минимальных затрат на генерацию доказательства обновления дерева Меркла. **4)**Пользователи совершают транзакции с минимальной комиссией.	**1)**Спотовая торговля. **2)**Перевод токена. **3)**Пополнение баланса. **4)**Вывод средств. **5)**Обновление учетной записи. **6)**Обновить AMM. **7)**NFT mint. **8)**Данные NFT.	~ **0,04$**	до **2025** транзакций в секунду.	**1)Обычный выход** Пользователь отправляет запрос на вывод оператору L2, после подтверждения оператором, запрос отправляется на L1. После пользователю доступен вывод на L1 без подтверждения merkle. **2)Принудительный выход** Если пользователь сталкивается с цензурой от оператора, он может отправлять свои запросы на вывод средств прямо на L1. Затем система обязана обработать этот запрос. Как только принудительная операция отправлена и если запрос обслужен, то операция выполняется в порядке общей очереди(в течении 15 дней).
zkSync				
1)Отсутствие доверия. **2)**Безопасность. **3)**Устойчивость. **4)**Возможность разветвления. **5)**Гипермасштабируемость. **6)**Смарт-контракты, совместимые с EVM без разрешения. **7)**Стандартный Web3 API. **8)**Сохранение ключевых функций EVM, таких как возможность компоновки смарт-контрактов. **9)**Представляем новые функции, такие как абстракция учетной записи.	**1)**zkSync поддерживает старые стандарты транзакций платформы ETHEREUM (pre-EIP2718, EIP1559 EIP712.). Транзакции этого типа можно использовать для таких специфичных функций zkSync, как абстракция аккаунтов. Кроме того, смарт-контракты можно развертывать только с этим видом транзакций.	~ **0.001$**	До **2 000** транзакций в секунду	

ZKSpace				3)**Аварийный выход** Если 15дней проходит, а принудительный выход игнорируется, пользователь может перевести систему в режим вывода, запретив дальнейшие обновления состояния. В этом случае каждый может вывести средства, предоставив merkle подтверждение своих средств с помощью транзакции L1.
1)Чрезвычайно быстрые транзакции (L2 могут быть подтверждены немедленно, не дожидаясь создания блока). **2)Низкая плата за газ.** **3)Неограниченная масштабируемость**, TPS может достигать 10000+. **4)Нулевое доверие** (активы пользователей хранятся в их кошельке).	**1)**Все виды транзакций поддерживаемые **zkSync + ряд функций модели AMM**, таких как создание торговых пар, добавление/удаление ликвидности, обмен и т. д. **2)Работа с NFT** (Transfer, withdraw, deposit).	**~ 0,05$**	До **6000** транзакций в секунду	
Aztek				
1)Платежи являются частными - Балансы и идентификационные данные для всех токенов в агрегированном пакете Aztec зашифрованы. Каждая транзакция кодируется как zkSNARK, защищая пользовательские данные. **2)**В случае сбоя секвенсора пользователи могут принудительно включить транзакции в цепочку L2, отправив их на L1. Предложение новых блоков требует создания доказательств ZK. **3)**Если Proposer терпит неудачу, пользователи могут использовать средство проверки с открытым исходным кодом для отправки доказательств на мост L1.	**1)**Конечно, проблема в совместимости с EVM, но, со слов команды, достичь полной приватности с EVM невозможно. Поэтому приходится искать хитрые решения. **И Aztec — это, фактически, блокчейн "VPN".** **2)**Aztec Connect дает возможность **приватного взаимодействия со смарт-контрактами** (торговля, кредитования, NFT, управление) в L1 из сети второго слоя. .	**~0,02$**	До **1800** транзакций в секунду	

Проанализировав полученные результаты, мы пришли к выводу, что ZK-Sync будет самым оптимальным инструментом для создания приложения с технологией ZK-Rollups. В процессе анализа мы установили, что у всех реализаций схожие способы вывода средств. ZK-Sync имеет преимущество над своими аналогами по данным критериям: низкая комиссия, безопасность, устойчивость и обработка большого количества транзакций в секунду.

Идея приложения

В качестве реализации примера для тестирования работы ZK-Rollups рассматривается приложение из документации, которое выполняет ряд базовых функций, а именно:

1. Перевод токенов на уровне L2 в Zk-Sync.
2. Перевод ETH с уровня L1 в L2 (Депозит).
3. Перевод ETH на уровне L2.
4. Вывод ETH с уровня L2 на уровень L1.

Алгоритм работы приложения

1. Разворачивается смарт-контракт в тестовой сети ZK-Sync который реализует вышеперечисленные методы.
2. Разворачивается смарт-контракт в тестовой сети первого уровня, который реализует межсетевое общение.
3. Пользователи осуществляют транзакции на уровне L2, а так же между уровнями L1 и L2.
4. Узел уровня L2 в свою очередь выпоняет базовую проверку поступивших транзакций и объединяет их в один пакет, генерируя для него доказательство с нулевым разглашением.
5. Пакет верифицируется смарт-контрактом первого уровня и этот смарт-контракт помещает пакет с конечными изменениями состояния в блокчейн первого уровня.

Открытые проблемы ZK-Rollups

В результате проведенного исследования были выявлены следующие проблемы:

1. Приложения ZK-Rollups в настоящее время по прежнему ограничены простыми платежами и транзакциями, поскольку необходимо продумать логику сжатия данных во время "свёртки"транзакций в пакеты.
2. На данный момент различные приложения ZK-Rollups не могут взаимодействовать на одном уровне L2.
3. Специфика языка Solidity делает труднодостижимым написание сложных смарт-контрактов, которые бы продолжали удовлетворять свойствам безопасности на уровне L1 и масштабируемости.

Результат

Все поставленные цели и задачи выполнены. В результате создано приложение для платформы Ethereum с использованием сверток с нулевым разглашением ZK-Rollups. Использование данной технологии уменьшило занимаемую память в блокчейне уровня L1 и позволило сэкономить на комиссии за транзакции за счёт более дешёвых цен на уровне L2. Также был проведен анализ реализаций ZK-Rollups для платформы Ethereum и выявлены проблемы существующих систем.

ЛИТЕРАТУРА

[1] Cryptofeed. Ethereum. [Электронный ресурс]. – URL: https://cryptofeed.live/alphabet/ethereum-efirium.

[2] Forbes. Что такое блокчейн: все, что нужно знать о технологии [Электронный ресурс]. – URL: https://www.forbes.ru/mneniya/456381-cto-takoe-blokcejn-vse-cto-nuzno-znat-o-tehnologii.

[3] Pravo. Смарт-контракты: как они работают и зачем нужны. [Электронный ресурс]. – URL: https://pravo.ru/story/205151.

[4] Ethereum documentation. Zero-Knowledge rollups. [Электронный ресурс]. – URL: https://ethereum.org/ru/developers/docs/scaling/zk-rollups.

[5] bits.media. Транзакция. [Электронный ресурс]. – URL: https://bits.media/tranzaktsiya.

[6] l2beat. Loopring. [Электронный ресурс]. – URL: https://l2beat.com/projects/loopring.

[7] l2beat. zkSync Lite. [Электронный ресурс]. – URL: https://l2beat.com/projects/zksync.

[8] l2beat. ZKSwap. [Электронный ресурс]. – URL: https://l2beat.com/projects/zkswap.

[9] l2beat. Aztek. [Электронный ресурс]. – URL: https://l2beat.com/projects/aztec.

Куратор исследования –
Кондырев Д.О. Преподаватель Новосибирского государственного университета

Список участников

1. **АРБЕЙТЕР Максим Андреевич** – Факультет вычислительной техники, Рязанский государственный радиотехнический университет им. В.Ф. Уткина
2. **БАХАРЕВ Денис Александрович** – Механико-математический факультет, Новосибирский государственный университет
3. **БАШКИРОВ Владимир Андреевич** – Институт физико-математических наук, Балтийский федеральный университет им. И. Канта
4. **БЕЛОВ Максим Алексеевич** – Факультет инфокоммуникационных сетей и систем, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича
5. **БЛОХИН Артём Александрович** – Институт прикладной математики и компьютерных наук, Томский государственный университет
6. **БОЛТНЕВ Юрий Федорович** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта
7. **БОЯЗИТОВ Вадим Дмитриевич** – Институт прикладной математики и компьютерных наук, Томский государственный университет
8. **БУЧНЕВ Александр Андреевич** – Факультет компьютерных и инженерных наук, Университет Иннополис
9. **БЫКОВ Денис Александрович** – Механико-математический факультет, Новосибирский государственный университет
10. **БЫСТРЫХ Алексей Викторович** – Институт прикладной математики и компьютерных наук, Томский государственный университет
11. **ВОЛКОВ Кирилл Андреевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта
12. **ГАЛУШКА Екатерина Алексеевна** – участник Летней школы
13. **ГЛУЩЕНКО Николай Александрович** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет
14. **ГОЛОВАНОВ Андрей Андреевич** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО
15. **ГОЛЯШОВ Антон Андреевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта
16. **ГРЕБНЕВ Сергей Владимирович** – ведущий криптограф-исследователь QApp, Российского квантового центра (Москва)
17. **ГУРЬЯНОВ Владимир Олегович** – Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет «МИФИ»

18. **ДАВЫДОВ Вадим Валерьевич** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

19. **ДАЛЕВИЧ Владислав Андреевич** – Факультет информационных технологий, Новосибирский государственный университет

20. **ДРОЖЖИНА Яна Евгеньевна** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

21. **ЕВСЕЕВ Александр Сергеевич** – Институт прикладной математики и компьютерных наук, Томский государственный университет

22. **ЕНИН Данил Васильевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

23. **ЕРЕМЕЕВ Руслан Тахирович** – Институт прикладной математики и компьютерных наук, Томский государственный университет

24. **ЕРОХИНА Анна Александровна** – Физический факультет, Московский государственный университет

25. **ЖДАНКИНА Светлана Андреевна** – Естественно-научный факультет, Дальневосточный государственный университет путей сообщения

26. **ЗАИКИН Олег Сергеевич –** Институт динамики систем и теории управления, специалист в области криптографии и SAT-вычислений, создатель лучшего SAT-решателя в 2019 году (по результатам международного конкурса SATrace)

27. **ЗИМЕНКО Кирилл Иванович** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

28. **ИБРАГИМОВА Алина Наильевна** – Факультет инфокоммуникационных сетей и систем, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

29. **ИОГАНСОН Иван Дмитриевич** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

30. **ИСАЙЧЕВА Алена Валерьевна** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

31. **КАЛГИН Константин Викторович** – Механико-математический факультет, Новосибирский государственный университет

32. **КАЛИНИНА Елизавета Андреевна** -– Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

33. **КИРШАНОВА Елена Алексеевна** -– Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

34. **КИРЬЯНОВА Анастасия Павловна** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

35. **КОВАЛЕВА Софья Андреевна** -– Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

36. **КОЛОМЕЕЦ Николай Александрович** -– Новосибирский государственный университет, специалист в области симметричной криптографии и криптоанализа

37. **КОНДЫРЕВ Дмитрий Олегович** – Новосибирский государственный университет, специалист в области блокчейн-технологий

38. **КОСТОЧКА Светлана Владимировна**– спортивный тренер Летней школы

39. **КОТОВ Даниэль Александрович** – Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет «МИФИ»

40. **КОЧЕТКОВА Валерия Кирилловна** – Новосибирский государственный университет

41. **КРУГЛЯКОВ Роман Сергеевич** – Институт киберфизических систем, Санкт-петербургский государственный университет аэрокосмического приборостроения

42. **КУНИНЕЦ Артем Андреевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

43. **КУЦЕНКО Александр Владимирович** – Новосибирский государственный университет, специалист в области постквантового криптоанализа

44. **ЛАРИОНОВА София Игоревна** – Институт киберфизических систем, Санкт-Петербургский государственный университет аэрокосмического приборостроения

45. **ЛЕ Куок Зунг** – Инженерно-экономический институт, Национально исследовательский университет МЭИ

46. **ЛЕЕВИК Антон Георгиевич** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

47. **ЛЕОНТЬЕВА Юлия Павловна** – Институт компьютерных технологий и информационной безопасности, Южный федеральный институт

48. **ЛЕРНЕР Ксения Александровна** -– Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

49. **МАЛЫГИНА Екатерина Сергеевна** -– Институт высоких технологий, научный сотрудник лаборатории «Математические методы защиты и обработки информации» НОМЦ «Северо-Западный центр математических исследований им. С.Ковалевской», Балтийский федеральный университет им. И. Канта,

50. **МАЛЮТИН Глеб Денисович** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

51. **МАМАДЖАНОВА Адина Мирхалидиновна** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

52. **МАРО Екатерина Александровна** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

53. **МЕЛЬНИЧУК Евгений Михайлович** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

54. **МУДРИЧ Анна Бобановна** – Высшая инженерно-техническая школа, Национальный исследовательский университет ИТМО

55. **МУХОРТОВА Алёна Андреевна** – Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет «МИФИ»

56. **НЕМОВА Ольга Юрьевна** – Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет «МИФИ»

57. **НЕЦВЕТАЙЛОВ Андрей Александрович** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

58. **НОВИКОВА Екатерина Вячеславовна** – Институт прикладной математики и компьютерных наук, Томский государственный университет

59. **НОВОСЕЛОВ Семен Александрович** – младший научный сотрудник лаборатории "Математические методы защиты и обработки информации старший преподаватель с ученой степенью кандидата наук ОНК "Институт высоких технологий Балтийский федеральный университет им. И. Канта

60. **ПИВОВАРОВ Фёдор Евгеньевич** – участник Летней школы

61. **ПОПКОВ Дмитрий Алексеевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

62. **ПОПОВ Никита Андреевич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

63. **РУБАН Егор Алексеевич** – Институт прикладной математики и компьютерных наук, Томский государственный университет

64. **РУДЗЯНСКИЙ Артемий Дмитриевич** – Московский государственный институт электроники и математики, Высшая школа экономики

65. **САЦУТА Анатолий Игоревич** – Высшая школа компьютерных наук и прикладной математики, Балтийский федеральный университет им. И. Канта

66. **СЕРОШТАН Наталья Николаевна** – Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

67. **СОКОЛОВА Диана Анатольевна** – Информационные системы и программирование, Колледж Информационных Технологий

68. **ТОГУМБАЕВ Рустам Денисулы** – Институт прикладной математики и компьютерных наук, Томский государственный университет

69. **ТОКАРЕВА Наталья Николаевна** – Механико-математический факультет, Новосибирский государственный университет, руководитель Криптографического центра (Новосибирск), председатель международной олимпиады Non-StopUniversity CRYPTO

70. **ФРОЛОВ Егор Дмитриевич** — Институт высоких технологий, Балтийский федеральный университет им. И. Канта

71. **ХИЛЬЧУК Ирина Сергеевна** — Механико-математический факультет, Новосибирский государственный университет

72. **ХОРЕВА Юлия Владиславовна** – Факультет инфокоммуникационных сетей и систем, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

73. **ХУЦАЕВА Алтана Феликсовна** – Факультет безопасности информационных технологий, Национальный исследовательский университет ИТМО

74. **ЦАРЕГОРОДЦЕВ Кирилл Денисович** – специалист-исследователь, АО «НПК «Криптонит»

75. **ЧИЖОВ Иван Владимирович** – специалист в области постквантовой криптографии, Московский государственный университет

76. **ЧУБАНЬ Артем Андреевич** — Институт высоких технологий, Балтийский федеральный университет им. И. Канта

77. **ШАПОРЕНКО Александр Сергеевич** – Новосибирский государственный университет

78. **ШАПОРЕНКО Андрей Сергеевич** — Институт прикладной математики и компьютерных наук, Томский государственный университет

79. **ЯКУНИН Никита Александрович** – ГАУ КО ОО Школа-интернат лицей-интернат «ШИЛИ»

80. **ЯКУШЕНОКС Кирилл Петрович** – Московский институт электроники и математики им. А.Н. Тихонова, Высшая школа экономики