Летняя школа-конференция "Криптография и информационная безопасность" 2022

Сборник тезисов

MATHEMATICAL CENTER IN AKADEMGORODOK

НГУ ФИТ

mathematics & mechanics

Новосибирск 2022

Содержание

О школе

Летняя школа-конференция «Криптография и информационная безопасность» памяти С.Ф. Кренделева для студентов и школьников —- традиционное мероприятие, проходящее в стенах НГУ каждый год. Организаторами школы-конференции выступают Криптографический центр (Новосибирск), Международный математический Центр в Академгородке, организаторы международной олимпиады NSUCRYPTO, Факультет информационных технологий и Механико-математический факультет.

Летняя школа - первый этап программы профессиональной переподготовки в области криптографии! У вас есть возможность продолжить исследования в данном направлении и получить диплом государственного образца.

Даты проведения: 27 июня - 11 июля 2022.

Формат участия: очный.

Организована дистанционная секция для иностранных участников!

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

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

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

Лекторы и преподаватели школы:

- **GANGOPADHYA Sugata** – PhD, декан факультета электроники и техники связи Индийского технологического института Рурки (г.Рурки, Индия);
- **АЛЕКСЕЕВ Евгений Константинович** – к.ф.-м.н., начальник отдела криптографических исследований, КРИПТО-ПРО (г.Москва);
- **АТУТОВА Наталья Дмитриевна** – студентка ММФ НГУ;
- **БАХАРЕВ Александр Олегович** – студент ММФ НГУ;
- **БОНИЧ Татьяна Андреевна** – аспирантка ФИТ НГУ;
- **БЫКОВ Денис Александрович** – студент ММФ НГУ;
- **ДОРОНИН Артемий Евгеньевич** – аспирант ФИТ НГУ;
- **ЗЮБИНА Дарья Александровна** – магистрантка ФИТ НГУ;
- **ИДРИСОВА Валерия Александровна** – к.ф.-м.н., н.с. Института математики им. С.Л.Соболева СО РАН;
- **ИЩУКОВА Евгения Александровна** – к.т.н., доцент кафедры безопасности информационных технологий Южного федерального университета (г.Ростов-на-Дону);
- **КАЛГИН Константин Викторович** – к.ф.-м.н., старший преподаватель кафедры параллельного программирования ФИТ НГУ, м.н.с. ИВМиМГ, н.с. ИМ СО РАН;
- **КОЛОМЕЕЦ Николай Александрович** – к.ф.-м.н., старший преподаватель кафедры теоретической кибернетики ММФ НГУ;
- **КОНДЫРЕВ Дмитрий Олегович** – аспирант ФИТ НГУ, ассистент кафедры компьютерных систем ФИТ НГУ;
- **КОСТОЧКА Светлана Владимировна** – преподаватель кафедры физвоспитания НГУ;
- **КУЦЕНКО Александр Владимирович** – к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ;
- **КЯЖИН Сергей Николаевич** – к.ф.-м.н., ведущий инженер-аналитик отдела криптографических исследований, КРИПТО-ПРО (г.Москва);
- **МАКСИМЛЮК Юлия Павловна** – аспирантка ММФ НГУ, ассистент кафедры компьютерных систем ФИТ НГУ;
- **МАЛЫГИНА Екатерина Сергеевна** – к.ф.-м.н., доцент Балтийского федерального университета им. И. Канта (г.Калининград);
- **МЕЛЬНИЧУК Евгений Михайлович** – главный расследователь BLIN.agency, ассистент БФУ им. И. Канта (г.Калининград);
- **МОКРОУСОВ Антон Сергеевич** – магистрант ФИТ НГУ;

- **ПАНФЕРОВ Матвей Андреевич** – аспирант ИМ СО РАН;
- **ПАРФЕНОВ Денис Романович** – магистрант ФИТ НГУ;
- **СУТОРМИН Иван Александрович** – магистрант ММФ НГУ;
- **ТОКАРЕВА Наталья Николаевна** – к.ф.-м.н., доцент кафедры компьютерных систем ФИТ, кафедры теоретической кибернетики ММФ, с.н.с. ИМ СО РАН;
- **ХИЛЬЧУК Ирина Сергеевна** – магистрантка ММФ НГУ;
- **ШАПОРЕНКО Александр Сергеевич** – аспирант ММФ НГУ, ассистент кафедры дискретного анализа и исследования операций ФИТ НГУ.

Организационный комитет:

- Токарева Наталья Николаевна;
- Атутова Наталья Дмитриевна;
- Бонич Татьяна Андреевна;
- Зюбина Дарья Александровна;
- Идрисова Валерия Александровна;
- Коломеец Николай Александрович;
- Кондырев Дмитрий Олегович;
- Косточка Светлана Владимировна;
- Куценко Александр Владимирович;
- Максимлюк Юлия Павловна;
- Шапоренко Александр Сергеевич.

Лекции:

1. Криптография: быстрый старт – Токарева Н.Н.
2. Алгебраический криптоанализ (1 часть) – Куценко А.В.
3. Хэш-функции – Коломеец Н.А.
4. Алгебраический криптоанализ (2 часть) – Куценко А.В.
5. Безопасный пароль: миф или реальность? (1 часть) – Ищукова Е.А.
6. Безопасный пароль: миф или реальность? (2 часть) – Ищукова Е.А.
7. Non-Stop University CRYPTO Olympiad – Идрисова В.А.
8. Блокчейн (основы) – Кондырев Д.О.
9. Блокчейн (смарт-контракты, сетевое взаимодействие, приложения) – Кондырев Д.О.
10. Различные уловки хакеров и как от них защититься – Ищукова Е.А.
11. Криптография на кодах – Малыгина Е.С.
12. Атаки на кодовые криптосистемы – Малыгина Е.С.
13. ARX-шифры – Коломеец Н.А.
14. Булевы функции в криптографии – Шапоренко А.С.
15. Программирование для криптографических задач – Калгин К.В.
16. Квантовые технологии и криптография (1 часть) – Куценко А.В.
17. Lattice based post-quantum cryptography and quantum cryptanalysis (1 часть) – Gangopadhyay S.
18. Lattice based post-quantum cryptography and quantum cryptanalysis (2 часть) – Gangopadhyay S.
19. SAT-решатели – Калгин К.В.
20. Квантовые технологии и криптография (2 часть) – Куценко А.В.
21. Криптография на эллиптических кривых – Малыгина Е.С.
22. Постквантовая криптография – Бахарев А.О.
23. Криптоанализ симметричных шифров – Токарева Н.Н.
24. Уязвимости и взломы смарт-контрактов и defi приложений (1 часть) – Мельничук Е.М.
25. Уязвимости и взломы смарт-контрактов и defi приложений (2 часть) – Мельничук Е.М.
26. О протоколах доказательства с нулевым разглашением на примере Bulletproof – Кяжин С.Н.
27. История криптографии – Токарева Н.Н.
28. Основы Monero, криптография в Monero – Мельничук Е.М.

29. Устройство и работа Ethereum 2.0 – Мельничук Е.М.

30. Что плохого можно сделать, неправильно используя криптоалгоритмы? – Алексеев Е.К.

Программа конференции

11 июля 2022г., начало в 11:15:

1. Н.Н. Токарева **Вступительное слово**
2. В.Д. Боязитов **"Реализация банковского смарт-контракта"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
3. Г.А. Жуков **"Разработка смарт-контракта, позволяющего создавать, покупать и продавать NFT"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
4. И.В. Исаков, Р.А. Маслов, В.В. Скудина, Н.Д. Атутова **"Реализация смарт-контракта, выпускающего NFT"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
5. А.И. Сацута **"Реализация смарт-контракта лотереи на языке Solidity с подтверждением беспристрастности"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
6. Е.И. Соколова, Е.С. Гричин **"Интеграция алгоритмов приватности в код смарт-контрактов"** (куратор – Д.О. Кондырев)
7. А.А. Чубань **"Генерация NFT в реальном времени с подтверждением непредвзятости"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
8. Ю.А. Шаманов **"Распределенный сбор средств с использованием системы блокчейн"** (кураторы – Е.М. Мельничук, Д.О. Кондырев)
9. А.Е. Доронин, Д.А. Зюбина, Е.А. Ищукова, Н.А. Коломеец, А.В. Куценко, Э.А. Пивнева, И.А. Сутормин **"Исследование криптографических свойств новых функций хэширования HBC и HAS01"** (кураторы – Е.А. Ищукова, А.В. Куценко, Н.А. Коломеец)
10. Н.А. Абрамова, Д.А. Быков, А.А. Ерохина, М.А. Панферов, Н.Н. Токарева, А.С. Шапоренко **"Криптографические замки для протокола Шамира"** (кураторы – Н.Н. Токарева, М.А. Панферов, Д.А. Быков, А.С. Шапоренко)
11. ECHIMAM Chineze **"An overview of block ciphers"** (куратор – Ю.П. Максимлюк)
12. Л.К. Бояндин **"Разработка симметричной криптосистемы и её криптоанализ"** (куратор – Ю.П. Максимлюк)
13. Р.И. Бархаткин, К.В. Калгин **"Конструкции APN-функций из векторных bent-функций"** (куратор – К.В. Калгин)
14. И.С. Хильчук **"О корреляционно-иммунных функциях с максимальной алгебраической иммунностью (продолжение)"**
15. С.А. Бобрышев, Н.С. Белянин **"Разностный криптоанализ ARX-шифра"** (кураторы – Н.А. Коломеец, А.С. Мокроусов)
16. И.М. Одуд, А.О. Бахарев, Д.Р. Парфенов, А.В. Куценко **"Новый подход к классификации XS-схем малой размерности"** (кураторы – А.В. Куценко, Д.Р. Парфенов)

17. А.А. Кунинец, Д.К. Воробьёв, В.А. Левин, А.О. Бахарев, Т.А. Бонич, Е.С. Малыгина, А.В. Куценко **"Постквантовые криптосистемы на обобщенных кодах Рида-Соломона: обзор известных атак и модификаций"** (кураторы – А.В. Куценко, А.О. Бахарев, Е.С. Малыгина)

18. Ю.А. Сергеева, А.П. Французова, Н.А. Абрамова, Т.А. Бонич, Д.А. Зюбина, Н.Н. Токарева, И.С. Хильчук, В.Ю. Черкашин, Е.А. Шашкина **"Дешифрование дневников баронессы М. В. Притвиц"** (кураторы – Н.Н. Токарева, Т.А. Бонич, Д.А. Зюбина)

19. **Закрытие школы-конференции**

КРИПТОГРАФИЯ В ИСТОРИИ

Дешифрование дневников баронессы М. В. Притвиц

Ю. А. Сергеева[1], А. П. Французова[1], Н. А. Абрамова[2], Т. А. Бонич[1], Д. А. Зюбина[1], Н. Н. Токарева[1], И. С. Хильчук[1], В. Ю. Черкашин[1], Е. А. Шашкина[3]
[1]Новосибирский государственный университет
[2]СибГУ им. М. Ф. Решетнева
[3]Специализированный учебно-научный центр НГУ
E-mail: I.sergeeva9@g.nsu.ru, apfrantsuzova@gmail.com, boniya1994@bk.ru, t.bonich@g.nsu.ru, d.zyubina@g.nsu.ru, tokareva@math.nsu.ru, i.khilchuk@g.nsu.ru, v.cherkashin@g.nsu.ru, e.shashkina1@school.nsu.ru

Аннотация

В рамках данного проекта были рассмотрены отрывки из двух дневников баронессы М. В. Притвиц. Мы разобрали тексты этих отрывков, перепечатали и перевели с иностранных языков. Хотя в дневнике не использовался специально разработанный шифр, в нем применялись элементы шифрования для достижения конфиденциальности: использование трех иностранных языков, многочисленные сокращения имен и фамилий, мелкий почерк и запись поверх уже написанного текста. Представляется, что результаты данного проекта дополнят сведения о судьбе семьи Притвиц, которая сыграла огромную роль в развитии Академгородка.

Ключевые слова: *дешифрование дневников, Притвиц, перевод дневников.*

Наталья Алексеевна Притвиц была одной из первых сотрудниц Сибирского отделения АН СССР и пресс-секретарем при пяти его руководителях. Она внесла огромнейший вклад в формирование культуры Академгородка. Одним из ее многочисленных достижений является сохранение архива своей семьи, российской ветви древнего немецкого рода Притвиц.

Наша команда посетила Интегральный музей-квартиру живой истории Академгородка, где нам подробно и увлекательно рассказали про судьбу семьи Притвиц, в частности бабушки Н. А. Притвиц, баронессы Марии Викторовны Притвиц. Там же нам показали ее дневники, текст которых был написан необычным образом. Анастасия Безносова-Близнюк, хранительница музея, предложила дешифровать их. Так родилась идея этого проекта, в рамках которого мы разобрали, напечатали и перевели части двух дневников баронессы М. В. Притвиц.

Краткая биография М. В. Притвиц

В архиве Н. А. Притвиц сохранилось много документов, портретов, вырезок из газет и журналов, которые помогли восстановить родословную и описать значимые события из жизни Притвицев. Данный архив находится в свободном доступе онлайн в Свободном архиве СО РАН. Опираясь на сохранившиеся в архиве документы и книгу, написанную Н. Н. Богуненко и Н. А. Притвиц, мы составили краткую биографию М. В. Притвиц, чьи дневники мы изучали.

Мария Викторовна Притвиц, по отцу Губер (Хубер), (02.07.1876 — 06.01.1961), родилась в Кишинёве и всю жизнь считала себя молдаванкой с русским языком как родным.

По материнской линии она состояла в родстве с семьей Богдан, молдавских дворян, к которым приезжал погостить А. С. Пушкин. Мать М. В. Притвиц, Варвара Карповна Котруца, была языковедом, а её дедушка занимался переводом научных текстов. Отец М. В. Притвиц, Виктор Карлович

Рис. 1: Проектная группа в музее

Рис. 2: Личные вещи М. В. Притвиц

Рис. 3: Мария Викторовна Притвиц

Губер (Хубер), был юристом, статским советником. Он происходил из русско-немецкой семьи, в которой его отец, Карл Карлович, был русским по матери и немцем по отцу. Вскоре после рождения М. В. Притвиц, единственного ребёнка, её родители переехали из Кишинёва в Одессу.

С 1886 - 1893 гг. М. В. Притвиц обучалась в Мариинской женской гимназии. По окончании 04 июля 1893 г. она получила свидетельство, дающее ей право работать домашней наставницей и преподавать географию и арифметику, а 15 июля 1893 г. – аттестат об окончании Одесской Мариинской городской Общественной женской гимназии. По всем предметам она преуспела и знала их на отлично, за проведение уроков по географии и арифметике получила хорошую оценку. Примечательно, что в своем жизнеописании 1947 г. она говорит о том, что закончила с золотой медалью и правом преподавания русского языка. Однако, в свидетельствах о выдаче аттестата и присвоении права преподавания подтверждения этому найдены не были.

В период с 1893 г.по 1894 г. М. В. Притвиц проходит двухгодичные курсы в дрезденском женском первом лицее, где она изучала французский, немецкий, английский языки, политэкономию, историю, химию и др.

В 1894 г. она возвращается назад в Одессу, где работает в детской столовой на Куликовом поле и дает уроки английского языка. Она посещает лекции в Новороссийском университете, делает

переводы для профессоров, изучает “Слово о полку Игореве” под руководством переводчика А. В. Лонгинова.

Так она живёт до 1897 г. В этом году она выходит замуж за Аркадия Павловича фон Притвиц и переезжает в Ашхабад, а затем в Варшаву. В этих городах она продолжает преподавать английский язык и делать переводы для разных учреждений вплоть до 1914 г.

Рис. 4: М. В. Притвиц и А. П. Притвиц

Кроме того, в 1908 - 1912 гг. М. В. Притвиц является секретарем ею организованного отдела Всероссийского общества повсеместной помощи пострадавшим на войне солдатам и их семьям. В 1911 г. она провела благотворительный вечер в пользу Ашхабадской общины сестёр милосердия красного креста, ставит танец для балета “Фея кукол”, показанного на этом приеме.

Аркадий Павлович Притвиц, муж Марии Викторовны, является потомком одной из двух ветвей рода Притвицев-унд-Гаффрон, которые оказались в России. Именно брак с ним дал ей статус баронессы. А. П. Притвиц был военным инженером-железнодорожником, повышенным до правителя канцелярии Средне-Азиатской (Закаспийской) ж. д. и Варшаво-Венской ж. д. На службе он сопровождал Николая II во время его перемещений по железной дороге и был с ним в дружеских отношениях. Интересно, что сам Николай II являлся крестным их младшего сына, Алексея, названного так в честь сына Николая II.

Всего у Марии Викторовны и Аркадия Павловича было три сына: Владимир, Виктор и Алексей. После революции Аркадий с двумя старшими сыновьями отправляется за армией А. И. Деникина и затем эмигрирует в Европу.

С началом Первой мировой войны в 1914 г. М. В. Притвиц переезжает с сыном Алексеем в Петербург к матери, где живет и работает до ее ссылки в Уфу в 1935 г. В это время она продолжает переводить, преподает английский и немецкий языки, посещает занятия в студии Малого драматического театра (1919 г.), работает библиографом в библиотеке Геолого-Разведческого нефтяного института (1930 г.).

Во время ее ссылки она активно сотрудничает с Башкирским медицинским институтом (делает для него переводы, консультирует по языкам его сотрудников и обучает иностранным языкам студентов), Башкирским институтом языка и литературы, Украинской академией наук, эвакуированной в Уфу во время ВОВ, делает переводы для промышленных компаний и другого рода переводы (музыкальные, художественные и т. д.).

М. В. Притвиц провела свои последние годы в Уфе и там скончалась в возрасте 85 лет.

Рис. 5: Аркадий Павлович и Мария Викторовна с детьми

Описание дневников

В ходе нашего проекта мы разобрали и перепечатали части дневников М.В. Притвиц, которые она вела в 1893-1894 гг. В это время ей было 17-18 лет. Первый из ее дневников посвящен последнему году обучения в Мариинской женской гимназии (1893 г.), а второй дневник был начат вскоре после её приезда обратно в Одессу после учёбы в Дрездене (1894 г).

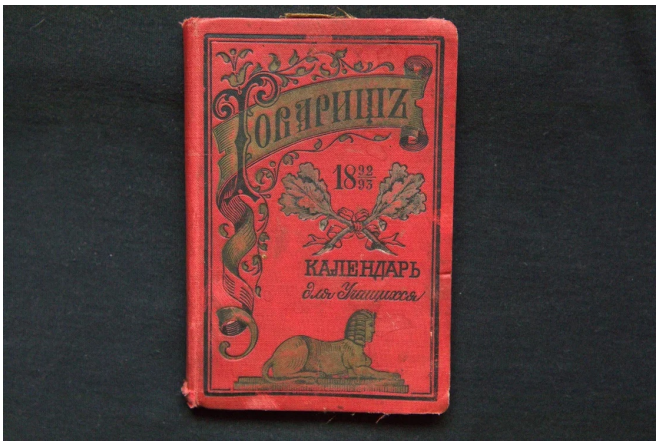

Рис. 6: Дневник для учащихся М. В. Притвиц 1893 г.

Рис. 7: Пропуск НГУ и маленький дневник М. В. Притвиц в сравнении

Первый дневник, далее красный дневник, был написан на русском языке и был школьным дневником (дневник для обучающихся). Второй дневник представляет из себя ежедневник небольшого размера, далее маленький дневник, который М. В. вела на иностранных языках: английском, итальянском и французском. Оба дневника вызвали значительную сложность в прочтении в связи размером рукописного шрифта, а также наложением текста поперек, что изначально было воспринято как шифр. При детальном знакомстве с дневниками выяснилось, что сам текст не зашифрован, однако, для его прочтения необходимо владение несколькими иностранными языками и знание особенностей ее почерка и письменного языка.

В маленьком дневнике каждая страница посвящена одному дню из жизни М. В. Притвиц. Она начинает описание горизонтально, после чего дневник поворачивался на 90 градусов и запись

продолжалась поверх уже написанного текста. Как правило, одна страница была написана на одном языке, иногда с вкраплениями и описывала ее эмоциональные переживания и наиболее яркие события.

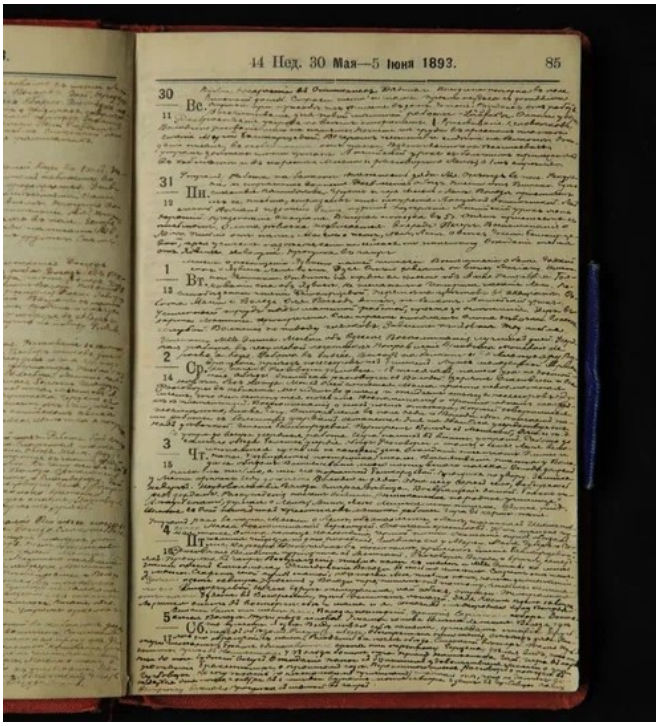

Рис. 8: Страница из красного дневника М. В. Притвиц

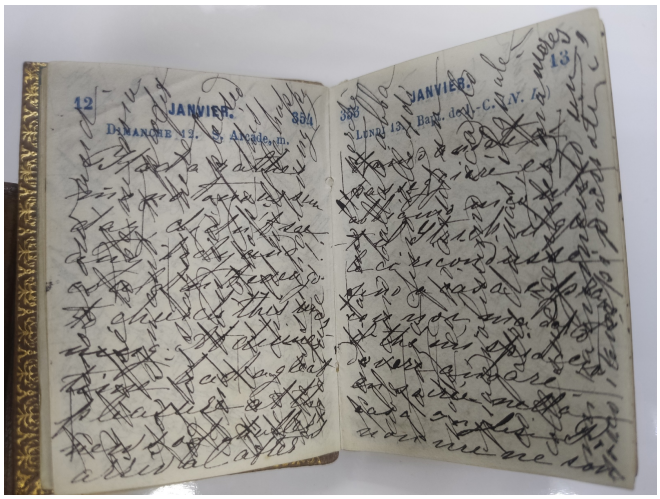

Рис. 9: Разворот из маленького дневника М. В. Притвиц. Левая страница написана на английском языке, а правая - на итальянском.

Работа с дневниками

По нашему предположению, баронесса старалась скрыть свои записи от посторонних глаз. Даже при попадании дневников в чужие руки прочитать их совершенно непросто. В красном дневнике текст настолько мелок, что прочитать его можно только с сильным увеличением. В маленьком дневнике частый переход с одного языка на другой и запись текста в двух направлениях (один текст поверх другого), что очень затрудняет чтение. Более того, нужно действительно хорошо разбираться как минимум в трех европейских языках - английском, итальянском, французском.

Членами нашей проектной группы была осуществлена следующая работа: перепечатано и переведено с английского языка 58 страниц из маленького дневника, перепечатано 40 страниц на итальянском языке и 10 страниц из красного дневника на русском языке. Для распознавания и перевода с английского и итальянского языков нами привлекались носители иностранного языка, в том числе специалисты разбирающиеся в истории того периода.

Страницы на русском языке (Красный дневник 1892-1893 года)

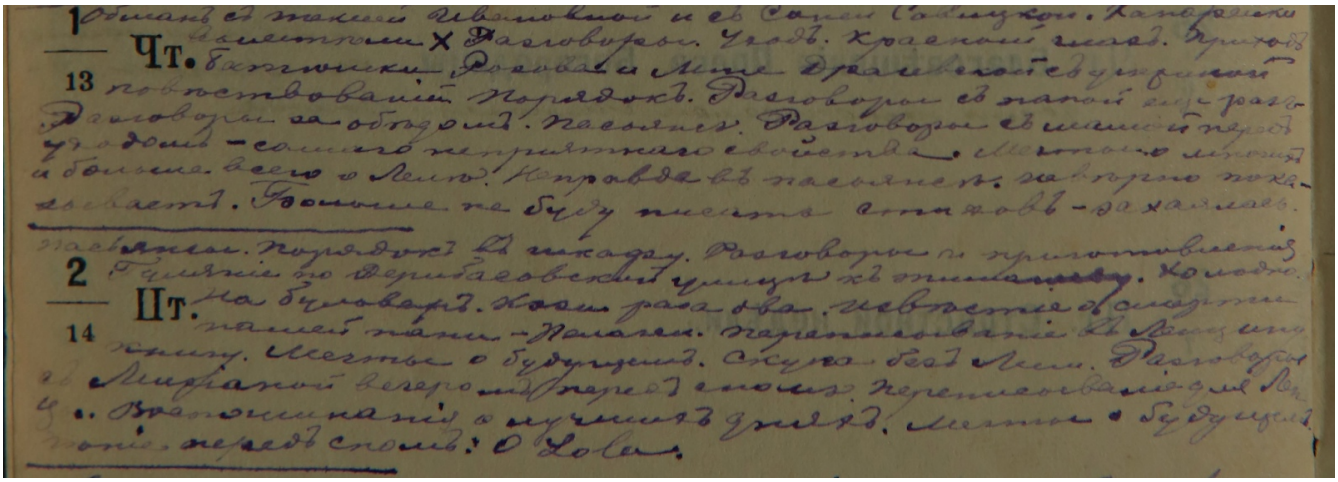

Рис. 10: Отрывок из красного дневника. День 1, Чт.

“Разговоры. Уход. Красные глаза. Приход батюшки. Порядок. Разговоры с папой. Ещё разговоры за обедом. Пасьянсы. Разговоры с мамой перед уходом — самого неприятного свойства. Мечты о многих и больше всего о Леле. Неправда в пасьянсе. Неверно показывает. Больше не буду писать стихов – захаялась.

Порядок в шкафу. Разговоры и приготовления. Гуляние по Дерибасовской улице. . . Холодно. Известие о смерти нашей няни – Пелагеи. Переписывание в Ленцину книгу . Мечты о будущем. Скучаю без Лели. Разговоры вечером перед сном. Переписывание для Ленцы. Воспоминание о лучших днях. Тоже перед сном: O Lola. ”

В книге описываются переживания Марии Викторовны, то как она разговаривала с папой и мамой о своих днях, ее времяпровождение, походы в церковь и в гости к знатным людям, а также изучение английского языка и английской грамматики. Приведенный выше пример хорошо это иллюстрирует.

В красной книге указывается и секретная информация, о которой упоминает М. В. Притвиц: фамилии и имена некоторых людей, которых она хотела бы скрыть. В красной книге Мария Викторовна записывает их в виде инициалов.

Другой день Марии Викторовны был более насыщенный, и там мы находим отсылки на другие любопытные события.

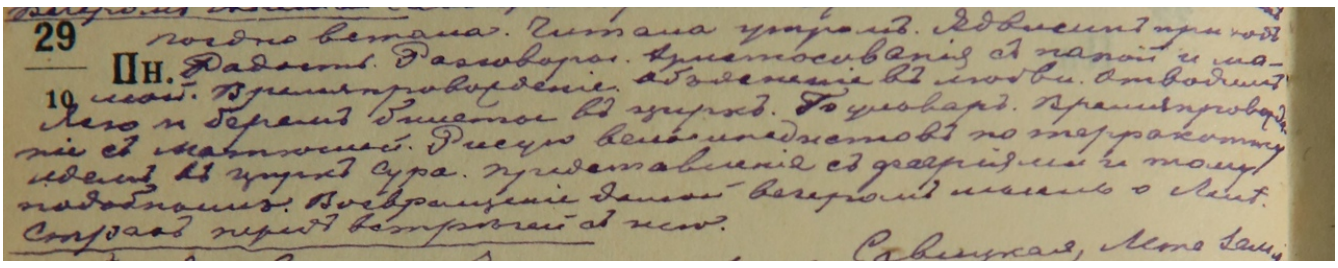

Рис. 11: Отрывок из красного дневника. День 29, Пн.

“29 Пн. Поздно встала. Читала утром. Ядвигин приход. Радость. Разговоры. Христосование с папой и мамой. Времяпровождение. Объяснения в любви. Отводим Лею и берем билеты в цирк. Бульвар. Времяпровождение с Матюшей. Рисую велосипедистов по терракоту. Идем в цирк Сура.

Представление с феериями и тому подобным. Возвращение домой вечером. мысль о Леле. Страх при встрече с нею.“

Вильгельм и Эдуард Сур давали примерно в это время в Одессе цирковые выступления, чему мы нашли подтверждение в одесской газете 1893 г.

Цирк Э. СУРА.

Большое представ. при 2 оркестрахъ музыки.

Гимназисты плат. балконъ и амфит. полов. цѣны.

Сегодня, въ среду, 10 го марта, дано будетъ

большое блестящее представленіе

при 2хъ оркестрахъ музыки; во второй разъ поставлено будетъ на сценѣ цирка „ВЕСЕЛЫЕ ГЕЙДЕЛЬБЕРСКІЕ СТУДЕНТЫ“, большая комическая пантомима въ 8-хъ картинахъ при участіи до 80 дамъ и мужчинъ; при двухъ оркестрахъ музыки поставлено будетъ Альбертомъ Суръ. Выходъ всѣхъ наѣздницъ и наѣздниковъ цирка. Начало въ 8 час. веч. Завтра, 11 марта, дано будетъ большое разнообразное представленіе при совершенно новой программѣ. *Примѣчаніе.* Учащіеся всѣхъ гимназій и реальныхъ училищъ имѣютъ право съ галлерейными билетами входить на балконъ, а съ балконными билетами входить въ амфитеатръ. Цѣна балкона 50 коп. 1—655—1

Рис. 12: Объявление в одесской газете о выступлении цирка Э. Сура. 10 Марта 1893 г.

Страницы на английском языке (маленький дневник 1894 г.)

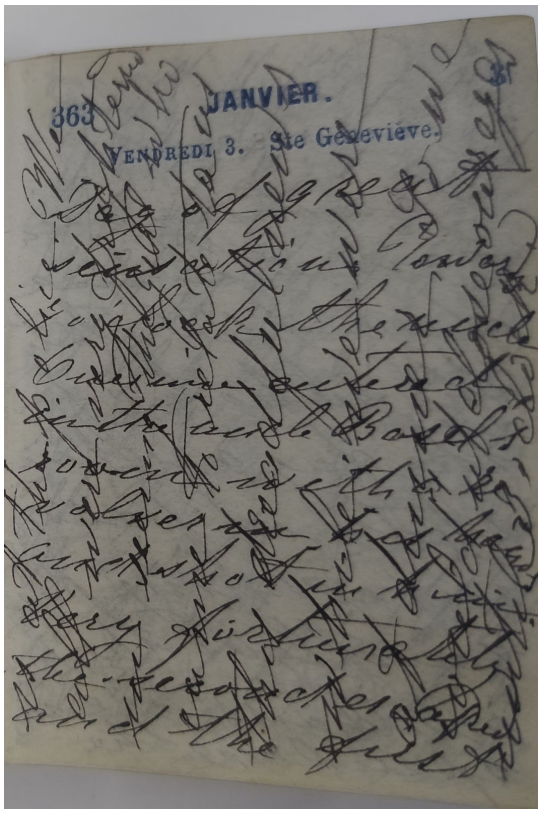

Рис. 13: Один день из маленького дневника, написанный на английском языке

“Day of great sensation. Towards 4* o’clock, the uncle O*** entered in the uncle Basil’s room with a revolver in his hands and shot in him. Very fortunately the second escaped and the first is at the police.

We all were very frightened in seeing the girl who came to relate to us the dreadful news while we were at dinner. So we couldn't close our eyes until 8* o'clock."

Перевод:

"День великой сенсации. Около 4* часов дядя О*** вошел в комнату дяди Василия с револьвером в руках и выстрелил в него. К счастью, второй сбежал, а первый находится в полиции. Мы все были очень напуганы, увидев девушку, которая пришла сообщить нам ужасную новость, пока мы обедали. Так мы не могли закрыть глаза до 8* часов."

В маленьком дневнике, возможно, поскольку объем листа на один день весьма ограничен, автор редко описывает подробности дня. Она скорее описывает в общих чертах, где она провела день, и свои чувства или отношение к пережитому дню. Эта страница - интересное исключение, где описывается действительно сенсационное событие. На следующий же день автор отмечает, что только об этом происшествии говорили весь день, до такой степени, что она устала об этом слушать ("I am rather sick of it.") и отмечает интересную подробность: "And to think that O. tells he only wanted to try his brother." Несколько непонятное употребление здесь слова "try" делает перевод затруднительным, однако, тот факт, что дядя О. и Basil (Василий) были братьями немедленно добавляет драму и глубину этой трагедии. Увы, дальнейшая судьба братьев нам не известна.

Примеры более типичных страниц, где автор не уточняет конкретные события, а скорее на них реагирует, встречаются позже:

"What very strange behaviour H. sometimes has, I really think he must have a very bad temper to have such a very strange behaviour, as I did change my own behaviour towards him. How very funny men are sometimes, they themselves do not know what they wish."

Перевод:

"Как странно иногда ведёт себя Х., мне действительно кажется, что у него должен быть плохой характер, чтобы так странно себя вести, и я изменила своё поведение по отношению к нему. Как же смешны иногда мужчины, они сами не знают, чего они желают."

"Gave my lessons during the day and spent the evening at grandMama's who was so pleased to have me* and gave me a very pretty pink silk dress. The rest of the evening I spent at home in dancing, for I was in very good humours."

Перевод:

"Дала уроки днем и провела вечер у бабушки, которая была так рада меня видеть и подарила мне очень красивое розовое шёлковое платье. Оставшийся вечер я провела дома, танцуя, потому что была в прекрасном настроении."

"Passed the evening at the theatre again but was bored to death by * who was cross because the theatre was nearly empty and on that account I was obliged to be her souffre-douleur."

Перевод: "Снова провела вечер в театре но чуть не умерла от скуки из-за *, которая была сердита, потому что театр был почти пуст, и в связи с этим мне пришлось терпеть её злость."

Страницы на итальянском языке (маленький дневник 1894 г.)

Часть страниц дневника были написаны на итальянском языке с вкраплением французских слов. Отметим, что переход с одного языка на другой происходит, по нашему предположению, по настроению баронессы на данный день. Как правило, в рамках одного дня она пишет на одном языке. Приведем пример страницы на итальянском языке.

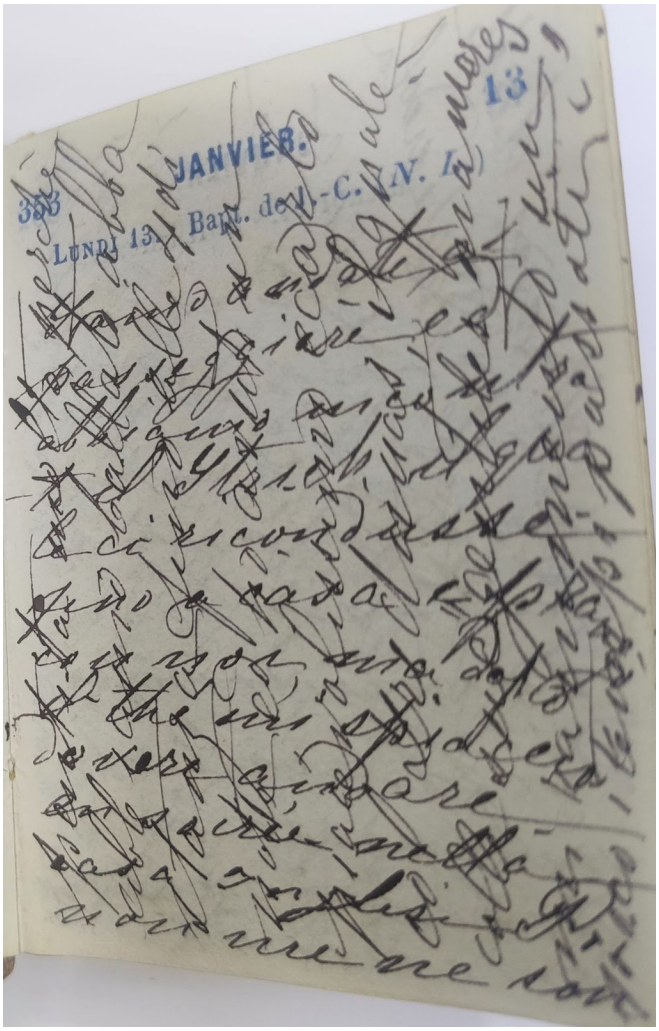

Рис. 14: Страница из маленького дневника. Запись от 13 января. Текст по-горизонтали.

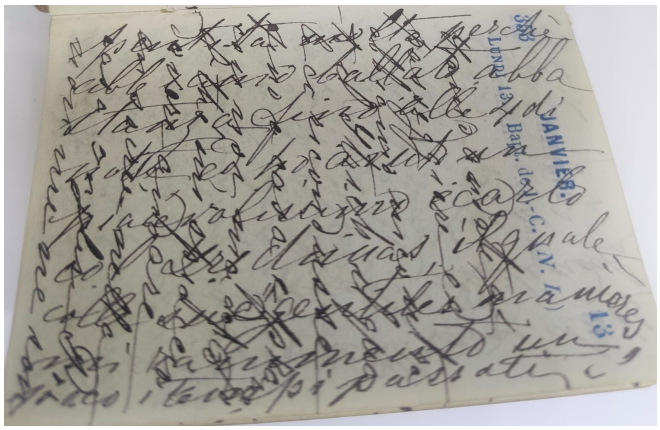

Рис. 15: Страница из маленького дневника. Запись от 13 января. Текст по-вертикали.

Текст по-горизонтали:

“Siamo andate a passeggiare ed abbiamo incontrato il Itrich il quale ci ricondusse fino a casa e pranzo‘ con noi ma dopo il the‘ mi spiaceva dovere andare *en soiree* nella casa inglese non me ne son”

Текст по-вертикали:

“pentita molto perche abbiamo ballato abbastanza fino alle 4 di notte ed ho avuto un piacevolissimo *Ecarte‘* col Itrich il quale colle sue gentiles manieres mi rammento‘ in fuoco i tempi passati.”

Перевод:

“Мы отправились погулять и встретили Итриха, который довёл нас до дома и пообедал с нами, но после чая я расстраивалась, что мне нужно идти *вечером* в английский дом, но в итоге не пожалела об этом, поскольку мы достаточно потанцевали до 4 утра и очень приятно провели время *за картами* с Итрихом, который своими приятными манерами вызвал во мне былой огонь.”

В данном тексте встречаются вкрапления французского языка. Мы отметили их курсивом. Еще стоит отметить одно интересное словосочетание “английский дом”, которое в тот период времени означало танцплощадку.

На нескольких страницах итальянского текста встречаются предложения с английскими словами, это свидетельствует о том, что баронесса была образована и бегло разговаривала на разных языках. Приведем в пример страницу из дневника:

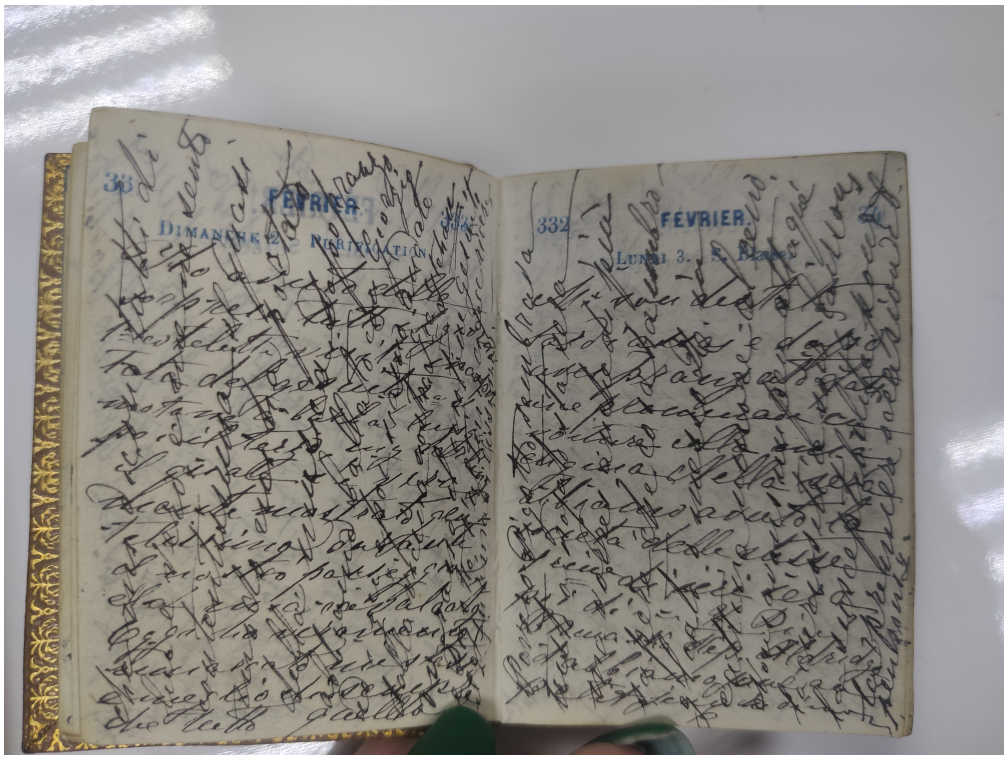

Рис. 16: Страница из маленького дневника. Текст по-горизонтали и по вертикали

Страницы на французском языке (маленький дневник 1894 г.)

Есть случаи, когда переход с одного языка на другой происходит не только при переходе от одной страницы к другой. Так, на одной из страниц первое предложение было написано на французском языке, дальнейший текст был написан на английском.

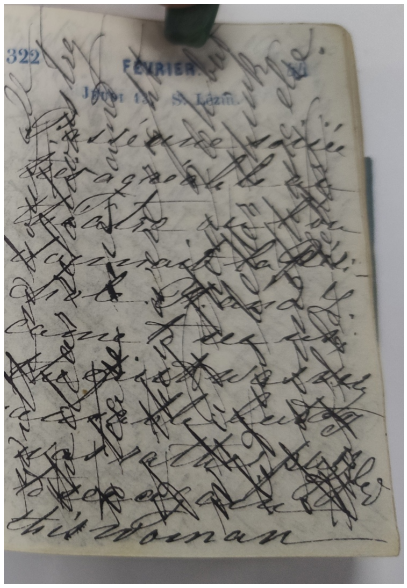

Рис. 17: Страница из маленького дневника. Запись от 13 февраля. Текст по-горизонтали и по вертикали

“Passé une soirée très agréable au théâtre ou lóu donnait la Pilichole. M. and I. came to see us, the first was very aurable but I was puzzled to see again thes’ woman with whom all the dear relations by to tease me, and if they were right, what then? Oh, but let us better think of something else that I’l be better.”

Также присутствует страница полностью на французском. Можно заметить различия между написанием некоторых слов на странице и написанием тех же слов в нынешнее время (например, évênement и événement, troisime и troisième), что может быть связано либо с ошибками в написании слов, либо с изменением самого языка.

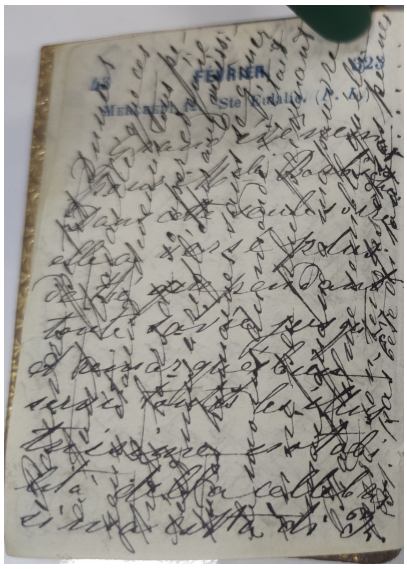

Рис. 18: Страница из маленького дневника. Запись от 12 февраля. Текст по-горизонтали и по вертикали

"Grand événement pour illé elle Barbiche. Dant cotte sèule soirée elle a vèrsé plus dethé que pondant tout savie jus de la jus quel of remarqué buis servent toutes les illes troisime notabilita delta celebrissima esta di Ir. Pauvre petit Dumas a tant travaillé jeux étaient devenus tous petrots. lors du so aperev quie c'est nusa reflichiraux meissitudes humaines sans se gener le moius dumande. etait en ne pen plus prevenant mais quano a ma pauvre. I*ate il me semble qu' elle perd son tem* et les peines Il. pas bele di laut..."

Заключение

Таким образом, нами была изучена биография М. В. Притвиц, перепечатаны и переведены части двух ее дневников. Опираясь на проделанную работу, можно говорить о том, что хотя М. В. Притвиц не использовала какого-то особого шифра при записи ее дневников, они все же не предназначались для посторонних глаз, о чем свидетельствует использование трех кодов - иностранных языков, многочисленные сокращения имен и фамилий, размер почерка и запись поверх уже написанного текста. Результаты данного проекта преумножат уже накопленные сведения о семье Притвиц, которая сыграла далеко не последнюю роль в развитии новосибирского Академгородка.

Благодарность

От лица нашей команды мы хотим выразить благодарность Интегральному музею-квартире живой истории Академгородка за предоставленную возможность заняться этим увлекательным проектом, а также за оказанную помощь в его осуществлении. Мы выражаем свою признательность Анастасии Безносовой-Близнюк и Александру Степанову за неоценимый труд в сохранении истории и культуры Академгородка, который для многих участников нашего проекта является малой Родиной и очень милым сердцу уголком.

ЛИТЕРАТУРА

[1] Открытый архив СО РАН. Дата обращения: 07.07.2022. Режим доступа: http://odasib.ru/openarchive;

[2] Ермиков, В. Д. Наталья Алексеевна Притвиц: Хранитель знаний. - Новосибирск: СО РАН, 2020 г. - 520 с.;

[3] Притвиц, Н. А., Н. Н. Богуненко. Русская ветвь: фон Притвицы в России. - Барнаул: ИД Барнаул, 2020. - 162 с.

Кураторы исследования –
к.ф.-м.н., с.н.с. Института математики им. С. Л. Соболева СО РАН, Наталья Николаевна Токарева;
аспирантка ФИТ НГУ, м.н.с. Международного математического центра (ММЦ), Татьяна Андреевна Бонич;
магистрантка ФИТ НГУ, инженер Международного математического центра (ММЦ), Дарья Александровна Зюбина.

ХЭШ-ФУНКЦИИ

Исследование криптографических свойств новых функций хэширования HBC и HAS01

А. Е. Доронин[1], Д. А. Зюбина[1], Е. А. Ищукова[2], Н. А. Коломеец[1], А. В. Куценко[1], Э. А. Пивнева[1], И. А. Сутормин[1]
[1]Новосибирский государственный университет
[2]Южный федеральный университет
E-mail: artem96dor@gmail.com, d.zyubina@g.nsu.ru, jekky82@mail.ru, kolomeec@math.nsc.ru, alexandrkutsenko@bk.ru, e.pivneva@g.nsu.ru, ivan.sutormin@gmail.com

Аннотация

В данной работе представлены результаты исследования новых хэш-функции HBC и HAS01 (модифицированная версия), основанных на функции сжатия, которая построена на основе блочного шифра, и конструкции «криптографическая губка» соответственно. Данные хэш-функции были реализованы и проверены с помощью статистических тестов NIST для валидации случайности их выхода. Временные замеры скорости выработки хэш-суммы для оригинальной и модифицированной версий HAS01 совпадают, так как используемая в настоящем исследовании модификация меняет только порядок использования элементов массива состояния, но не меняет количество совершаемых операций. Для модифицированной версии HAS01 была обнаружена слабость, приводящая к коллизиям. Показано, что для оригинальной функции HAS01 данная слабость отсутствует.

Ключевые слова: *хэш-функция, блочный шифр, конструкция «губка».*

Введение

В рамках проекта команде было предложено рассмотреть два алгоритма хэширования, разработанных командами из Казахстана: алгоритм HBC-256, представленный в работе [1], и HAS01, представленный в [2]. Задачей номер один стала программная реализация данных хэш-функций на языке программирования C/C++ и получение замеров времени их работы. Разработчики HAS01 все еще вносят правки в алгоритм, поэтому далее в тексте будут упоминания о версии функции. Задача номер два — проверка выходных данных хэш-функций с помощью статистических тестов NIST. И третья задача — анализ стойкости функций к методам построения коллизий.

Описание алгоритмов хэширования

Функция хэширования HBC-256

Алгоритм хэширования данных HBC-256 (*Hash based on Block Cipher*) [1] разработан на основе функции сжатия CF (*Compression Function*), которая создана на базе блочного шифра. На сегодняшний день подход с использованием функции сжатия является популярным и устоявшимся. На входе функция сжатия принимает два 128-битных блока: блок сообщений m^j и раундовый ключ шифрования. На выходе функция выводит промежуточный 128-битный хэш-код. Для построения хэш-функции используется конструкция Меркля-Дамгора с наиболее распространенной модификацией wide-pipe [3], способной противостоять атаке удлинением сообщения (*length extension*

attack). Для создания конечного n-битного хэш-кода размер блока сообщений и размер промежуточного хэш-кода должны быть одинаковой длины w бит, причем $n < w$. Чтобы обеспечить требования модификации wide-pipe, в одном цикле хэширования одновременно выполняем k раз CF для разных m^j, $j = 0, \ldots, k-1$. Поэтому длина промежуточного хэш-кода w равна $128 \cdot k$ бит.

Для противодействия нахождению коллизий используется схема Дэвиса-Мейера [4], где выход CF суммируется (операция XOR) с результатом предыдущей итерации хэширования h_{i-1}^j. Значение p^j является результатом i-ой итерации хэш-функции на основе схемы Девиса-Мейера. Данная схема используется в алгоритмах хэширования на основе блочных шифров и выступает в качестве односторонней функции сжатия.

Алгоритм хэширования HBC-256

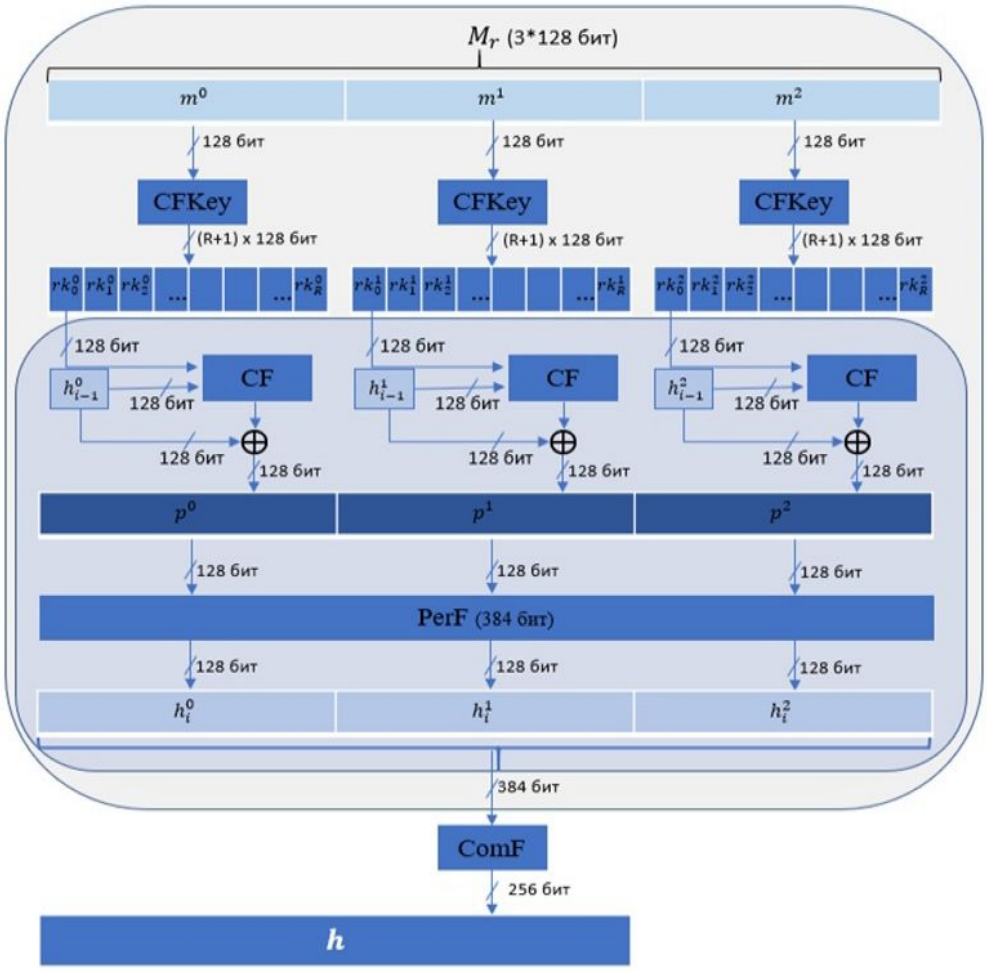

Рис. 1: Общая схема хэширования алгоритма HBC

Алгоритм HBC-256 итеративно обрабатывает $128 \cdot k$-битные блоки входного сообщения M. Если длина кратна $128 \cdot k$, то в конце добавляется еще один $128 \cdot k$-битный блок, состоящий из нулевых битов, за исключением первого и последнего бита, которые равны единице. Если длина не кратна $128 \cdot k$, то дополняется таким количеством битов, чтобы оно было кратно $128 \cdot k$. Для хэширования дополненное сообщение разделяем на t блоков размером $128 \cdot k$ бит каждый следующим образом: $M = M_0 \parallel M_1 \parallel \cdots \parallel M_{t-1}$. Далее, берется блок M_0, состоящий из первых 384 бит сообщения , и делится на три 128-битовые части m^0, m^1, m^2. На основе каждого m^j генерируются отдельные раундовые ключи rk_i^j, используя алгоритм развертывания раундовых ключей CFKey, где

$i = 1, 2, \ldots, R_1$ ($R_1 = 4$), и $j = 0, 1, \ldots, k-1$.

Алгоритм развертывания ключей CFKey состоит из преобразований StageKey-1, StageKey-2 и StageKey-3. Представленный алгоритм развертывания раундовых ключей схематически показан на рис. 2, также с данным алгоритмом более подробно можно ознакомиться в [1].

В самом начале инициализационный вектор или хэш-код принимает значение 0, т.е. $h_0^j = 0^{128}$. Далее для всех трех частей одновременно выполняется алгоритм шифрования CF, принимающий в качестве входных данных раундовый ключ rk_0^j и h_0^j.

Алгоритм шифрования CF относится к классу симметричного блочного шифрования с длиной блоков и ключей 128 бит. В алгоритме используются как линейные (сложение по модулю 2, циклические левые сдвиги), так и нелинейные (четыре S-блока замены) преобразования. Структура шифра является вариантом подстановочно-перестановочной сети (SP-сети) с четырьмя раундами (R_1). Один раунд шифрования состоит из трех преобразований, называемых Stage-1, Stage-2 и Stage-3 и представлен на рис. 3.

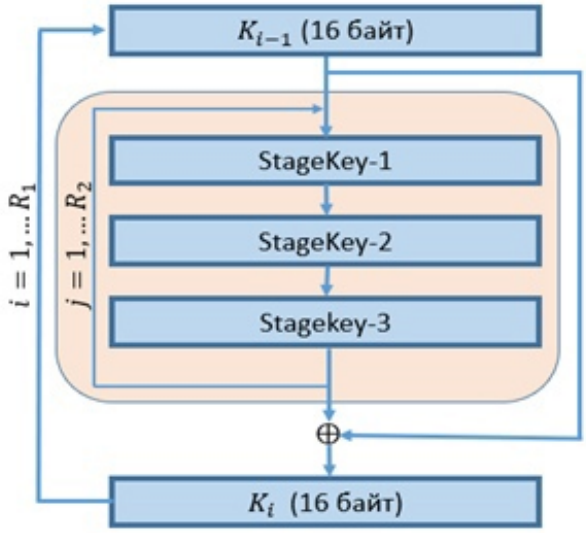

Рис. 2: Схема генерации раундовых ключей CFKey

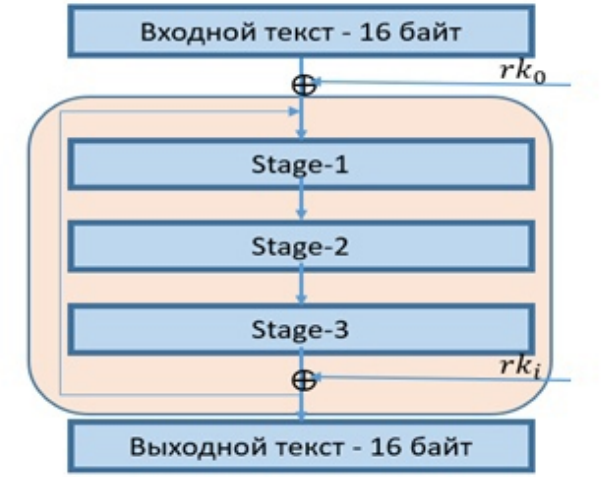

Рис. 3: Общая схема алгоритма CF

Авторы хэш-функции отмечают, что для CFKey, и для CF, преобразования Stage-1 и Stage-3 одинаковые. В них применяется процедура SBOX: задаются четыре подстановки S_0, S_1, S_2, S_3, где $S_i : Z_2^4 \to Z_2^4$, $i = 0, \ldots, 3$. Для работы авторы хэш-алгоритма выбрали четыре "золотых"S-блока: S_0 взят из шифра Serpent [5], S_1 из шифра Hummingbird-1 [6] и S_2, S_3 из шифра Hummingbird-2 [7] (рис. 4).

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
$S_0(x)$	0	F	B	8	C	9	6	3	D	1	2	4	A	7	5	E	Serpent, S_3
$S_1(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D	HB-1, S_2
$S_2(x)$	7	C	E	9	2	1	5	F	B	6	D	0	4	8	A	3	HB-2, S_0
$S_3(x)$	4	A	1	6	8	F	7	C	3	0	E	D	5	9	B	2	HB-2, S_1

Рис. 4: Четыре «золотых» S-блока

Далее по схеме Дэвиса-Мейера получаем p^j как результат суммирования h_1^j и h_0^j по модулю 2.

После этого с помощью процедуры PerF производится перестановка местами значений всех трех p^j, которые затем делятся на три части длиной 128 бит каждая. Процедура байтовой перестановки PerF осуществляется по формуле: $h_{ki+j} = h_{i+16j}, i = 0, \ldots, 15; j = 0, \ldots, k$.

Окончательный хэш-код определяется через процедуры ComF (Compression Function). В нашем случае в качестве окончательного хэш-кода берется значений первого и второго блока, длина, которая равна 256 бит или 32 байт: $h = h_1^0 \parallel h_1^1$.

Алгоритм хэширования HAS01

Алгоритм хэширования HAS01 [2] основан на конструкции «губки», представленной на рис. , который преобразует открытый текст произвольной длины 192-битными (или 24-байтовыми) блоками в хэш-значение размером 256 или 512 битов.

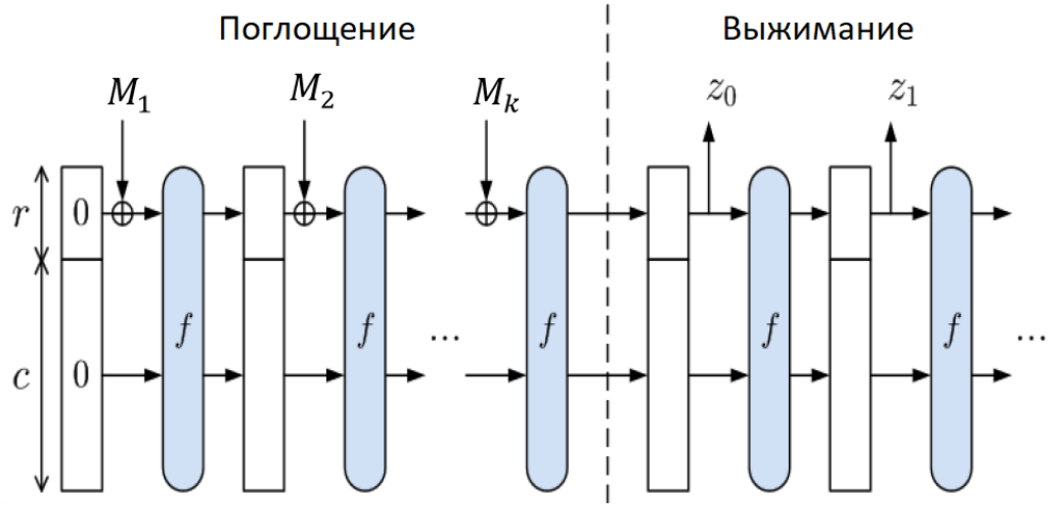

Рис. 5: Классическая конструкция губки для хэш-функций.

Конструкция губки представляет собой простую итеративную конструкцию с входными данными переменной длины и выходными данными произвольной длины на основе преобразования f, работающего с фиксированным числом битов b, где b называется шириной *(width)*. Конструкция губки работает с данными состояния, состоящими из $b = r + c$ бит, где r — параметр скорости *(bitrate)*, а c — параметр безопасности *(capacity)*.

Первоначально все b битов состояния принимают нулевые значения, а входное сообщение дополняется и делится на блоки по r бит каждый. Выполнение функции губки проходит в две стадии: поглощение *(absorbing)* и выжимание *(squeezing)*.

Блок входных данных M_i имеет размер 24 байта и может быть представлен в виде 24-байтовой последовательности $m_1, m_2, \ldots, m_{24}$. Внешняя часть состояния хэша r_i имеет тот же размер, что и размер блока входных данных (24 байта). Внутренняя часть состояния хэша c_i имеет размер 40 байт. Таким образом, текущее состояние хэша y_i представляет собой объединение внешней и внутренней части состояния хэша, то есть $y_i = r_i||c_i$.

Математическое описание алгоритма HAS01

В начале алгоритм хэширования разбивает исходное сообщение M с открытым текстом на отдельные блоки $M_1, M_2, \ldots, M_k$ размером 24 байта: $M = M_1||M_2|| \ldots ||M_{k-1}||M_k$. Если длина сообщения M не кратна длине блока, то выполняется дополнение последнего блока единицей с

последующими нулями. Если длина сообщения M кратна длине блока, выполняется добавление нового блока M_{k+1}, состоящего из единицы с последующими нулями. Начальное состояние хэша y_0 инициализируется нулевыми значениями. На стадии поглощения выполняются следующие преобразования:

$f(M_1, 0) = F(F(F(F((M_1 \oplus 0)||0)))) = y_1$

$f(M_2, y_1) = F(F(F(F((M_2 \oplus r_1)||c_1)))) = y_2$

$f(M_3, y_2) = F(F(F(F((M_3 \oplus r_2)||c_2)))) = y_3$

...

$f(M_k, y_{k-1}) = F(F(F(F((M_k \oplus r_{k-1})||c_{k-1})))) = y_k$,

где $F : [8 \times 8] \to [8 \times 8]$ — функция преобразования текущего состояния хэша, с которым более подробно можно ознакомиться в [2].

На каждой итерации стадии выжимания формируется очередной элемент z_i значения хэша, представляющий собой 8-байтовую последовательность, получаемую путём копирования элементов определённого столбца текущего состояния хэша x_i, Для значения хэша длиной 256 бит копируются элементы столбца с индексом 5, а хэша длиной 512 бит копируются элементы столбца с индексом 3:

$$z_i \leftarrow (x_{0,5}, x_{1,5}, x_{2,5}, x_{3,5}, x_{4,5}, x_{5,5}, x_{6,5}, x_{7,5}) \text{ для хэша длиной } L = 256 \text{ бит.}$$

$$z_i \leftarrow (x_{0,3}, x_{1,3}, x_{2,3}, x_{3,3}, x_{4,3}, x_{5,3}, x_{6,3}, x_{7,3}) \text{ для хэша длиной } L = 512 \text{ бит.}$$

Таким образом, хэш-значение $h(M)$ представляет собой последовательность байтов $z_1, z_2, \ldots, z_l$:

$$h(M) = z_1||z_2|| \ldots ||z_l,$$

где $l = 4$ при длине хэша 256 бит и $l = 8$ при длине хэша 512 бит.

Тесты NIST

Генерация данных для тестов NIST

Входное сообщение состоит из двух константных блоков произвольной длины (< $|M|/2 - 2$, где M — входное сообщение), блока из нулей и числа от 0 до 32767 в случае HBC и HAS01 с размером хэш-значения 256 бит, и от 0 до 16383 в случае HAS01 с размером хэш-значения 512 бит. Размер сообщения для HBC – 48 байт (384 бита), для HAS01 – 24 байта (192 бита).

Формирование входных данных происходит следующим образом:

Константные блоки задаются с помощью функции rand(). В начало сообщения записывается первый константный блок, затем число. Далее идет блок из нулей и в конце записывается второй константный блок. Также тестировались сообщения вида:

1. константный блок – блок из нулей – число;
2. константный блок – число – блок из нулей – константный блок (константные блоки одинаковые);
3. блок из нулей – число – блок из нулей.

Нам было необходимо получить файл с хэш-значениями размером 1МБ, для этого было сформировано 32768 сообщений для HBC и HAS01 с размером хэш-значения 256 бит (256 бит*32768 = 1МБ), и 16383 - в случае HAS01 с размером хэш-значения 512 бит (512 бит*16383 = 1МБ) для каждого константного блока.

Временные показатели генерации данных

Для замеров скорости работы выполнения алгоритмов использовался персональный компьютер со следующими характеристиками: Intel Core i7-8550U, 4-х ядерный процессор на 2.2 ГГц, с оперативной памятью 8 ГБ на 2133 ГГц.

Размер входных данных	HBC-256	HAS01-256	HAS01-512
1МБ	0,9 с	0,87 с	0,8 с
5МБ	3,1 с	4,3 с	3,8 с
10МБ	6,3 с	8,6 с	7,7 с
20МБ	12,5 с	17,2 с	15,6 с

Результаты тестов NIST

Над сгенерированными данными были проведены статистические тесты NIST для валидации генераторов случайных чисел и генераторов псевдослучайных чисел для криптографических приложений [8]. Был использован весь набор тестов, запущенных со стандартными параметрами. Обе хэш-функции успешно прошли все тесты.

Коллизии HAS01

В рамках текущего исследования для функции хэширования HAS01 была использована модифицированная версия. В оригинальной версии HAS01 функция F состоит из нелинейного преобразования $f1$ и линейных преобразований $f2$ и $f3$. Нелинейная функция $f1$ выполняет преобразование каждого байта матрицы в соответствии с формулами из [2] следующим образом. Каждая строка матрицы преобразуется в зависимости от четырех байтов. При этом вторая строка преобразуется только в зависимости от элементов исходного состояния. Остальные строки принимают на вход как байты исходного состояния, так и уже измененные байты.

В рамках настоящей работы была использована модифицированная версия алгоритма HAS01, в которой для функции $f1$ использовалась рекурсия при выработке. То есть для обработки каждого следующего байта состояния использовались ранее обработанные значения байтов состояния. Такая модификация алгоритма оказалась неудачной: из 192 входных бит каждого обрабатываемого блока 9 бит не используются за счет применения операции циклического сдвига влево на 3 позиции: старшие три бита в байтах $A00$, $A10$ и $A20$ (рис. 6).

Любые два исходных сообщения, которые отличаются только в этих позициях в любых блоках, будут давать одинаковые значения хэш-суммы.

Так, например, если на вход подать два сообщения (сообщения приведены в 16-ричном виде):

Сообщение 1 : 537472696e6720310d0a537472696e6720320d0a53747269

Сообщение 2 : 537472696e6720318d0a537472696e6720320d0a53747269

Будут выработаны одинаковые хэш-суммы:

Hash-256: 3451d136af58c3bade317fbded1ebae1bb07ad4773ca5361b7bc9689cbe6

Hash-512: 2ea85a7ce2932e75f7625ed4533cf38b9a36aa37b83d162775de28ebb24b90422886c7c23df82a4f36711abafd6827029fac4732ffaa439ec7fe4c888146fc

Таким образом, можно подобрать коллизии практически к любому тексту. Теоретически и экспериментально показано, что введенное дополнение в виде рекурсии ослабляет исходную функцию

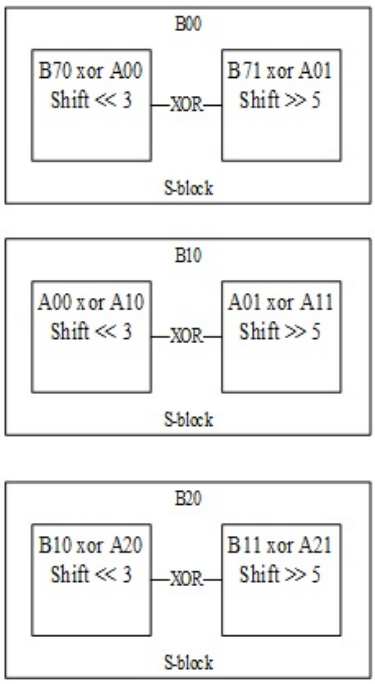

Рис. 6: Преобразование байтов $A00$, $A10$ и $A20$.

хэширования HAS01 и делает ее непригодной для использования в модифицированном виде, в то время как оригинальное построение функции HAS01 защищает ее от подобного рода коллизий.

ЛИТЕРАТУРА

[1] **Sakan K., Nyssanbayeva S., Kapalova N., Algazy K., Khompysh A., Dyusenbayev D.** Development and analysis of the new hashing algorithm based on block cipher // Eastern-European Journal of Enterprise Technologies. Ukraine. 2022. № 2/9(116). P. 60–73.

[2] **Сакан К.С., Дюсенбаев Д.С., Алгазы К.Т., Лизунов О.А., Ардабек Хомпыш** Разработка и анализ алгоритма хэширования «HAS01» // Сборник статей IV международной научно-технической конференции «Минские научные чтения-2021». – Минск. 09 декабря 2021 г. – Т.3., С. 181-187.

[3] **Stefan L.** Design Principles for Iterated Hash Functions // Cryptology ePrint Archive, Paper 2004/253, 2004, `https://eprint.iacr.org/2004/253`

[4] **Preneel B.** Davies–Meyer Hash Function // In: van Tilborg, H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA, 2005

[5] **Ross A., Eli B. and Lars K.** The Case for Serpent // 2002

[6] **Fan X., Hu H., Gong G., Smith E.M., Engels D.** Lightweight Implementation of Hummingbird Cryptographic Algorithm on 4-Bit Microcontroller // The 1st International Workshop on RFID Security and Cryptography 2009 (RISC'09), pp. 838–844, 2009.

[7] **Engels D., Saarinen MJ.O., Schweitzer P., Smith E.M.** The Hummingbird-2 Lightweight Authenticated Encryption Algorithm // In: Juels, A., Paar, C. (eds) RFID. Security and Privacy. RFIDSec 2011. Lecture Notes in Computer Science, vol 7055. Springer, Berlin, Heidelberg, 2012.

[8] **Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert N., Dray J., Vo S., Bassham L.**, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2010, `https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf`

Кураторы исследования –

к.т.н., доц. каф. БИТ ЮФУ, Евгения Александровна Ищукова;

к.ф.- м.н., асс. каф. ТК ММФ НГУ, н.с. ИМ СО РАН, Александр Владимирович Куценко;

к.ф.- м.н., ст.п. каф. ТК ММФ НГУ, н.с. ИМ СО РАН, Николай Александрович Коломеец.

ДИЗАЙН ШИФРОВ

Криптографические замки для протокола Шамира

Н. А. Абрамова[1], Д. А. Быков [2], А. А. Ерохина[3], М. А. Панферов[4], Н. Н. Токарева[2,4], А. С. Шапоренко[2]

[1] Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнёва

[2]Новосибирский государственный университет

[3]Московский государственный университет имени М.В. Ломоносова

[4]Институт математики им. С. Л. Соболева СО РАН

E-mail: boniya1994@gmail.com, d.bykov1@g.nsu.ru, erokhina.aa19@physics.msu.ru, m.panferov@g.nsu.ru, tokareva@math.nsc.ru, a.shaporenko@g.nsu.ru

Аннотация

Протокол Шамира — способ безопасной передачи информации без необходимости распространения ключей шифрования. В данной работе рассматривается возможность реализации трёхэтапного протокола Шамира с помощью симметричного шифрования. Описаны различные подходы к решению этой задачи, такие как использование известных симметричных криптографических систем (DES, одноразовый блокнот) и гомоморфного шифрования, а также применение свойств подгрупп группы всевозможных взаимооднозначных функций.

Ключевые слова: *протокол Шамира, симметричная криптография, гомоморфное шифрование.*

Введение

В криптографии трёхэтапный протокол для отправки сообщений представляет собой алгоритм, который позволяет организовать обмен секретными сообщениями по незащищенной линии связи без обмена секретными ключами или их распространения. Основная концепция трёхэтапного протокола заключается в том, что каждая сторона имеет закрытый ключ зашифрования и закрытый ключ расшифрования. Отправитель и получатель обмениваются тремя зашифрованными сообщениями. При этом они используют свои ключи независимо — сначала для шифрования сообщения, а затем для его расшифровки.

Первый трёхэтапный протокол был разработан Ади Шамиром примерно в 1980 году. Рассмотрим этот алгоритм подробнее.

Пусть Алиса желает передать Бобу секретное сообщение m, где m — натуральное число. Для этого она выбирает достаточно большое случайное простое число p, $p > m$, и открыто передает его Бобу. Своим секретным ключом Алиса выбирает два числа c_A и d_A такие, что $c_A d_A \equiv 1 (\mathrm{mod}\ p-1)$. Аналогичные действия проделывает Боб.

	Открытый ключ	Секретный ключ
Алиса	простое число p	числа c_A и d_A такие, что $c_A d_A \equiv 1 (\mathrm{mod}\ p-1)$
Боб	простое число p	числа c_B и d_B такие, что $c_B d_B \equiv 1 (\mathrm{mod}\ p-1)$

Таблица 1: Ключи для трёхэтапного протокола Шамира

Алгоритм передачи сообщения m от Алисы к Бобу:

- Алиса вычисляет $x_1 = m^{c_A} \bmod p$ и отправляет его Бобу.
- Боб, получив число x_1, вычисляет число $x_2 = x_1^{c_B} \bmod p$ и отправляет его Алисе.
- Алиса вычисляет $x_3 = x_2^{d_A} \bmod p$ и отправляет его Бобу.
- Боб вычисляет число $x_4 = x_3^{d_B} \bmod p$.

В конце алгоритма Боб вычисляет x_4, которое равно исходному секретному сообщению m.

$$x_4 = x_3^{d_B} \bmod p = (x_2^{d_A})^{d_B} \bmod p = ((x_1^{c_B})^{d_A})^{d_B} \bmod p =$$
$$(((m^{c_A})^{c_B})^{d_A})^{d_B} \bmod p = (m^{c_A d_A})^{c_B d_B} = (m^{1+k_1(p-1)})^{1+k_2(p-1)} \bmod p.$$

Теорема 1 (Малая теорема Ферма)**.** *Если целое число a не делится на простое число p, то* $a^{p-1} \equiv 1 (\bmod p)$.

Далее в работе рассмотрим, можем ли мы реализовать трёхэтапный протокол с помощью других систем симметричного шифрования без использования алгоритма возведения в степень, предложенного Шамиром.

Использование симметричных шифров в протоколе Шамира

Симметричный шифр представляет из себя семейство взаимно однозначных отображений, параметризованное ключом. Далее рассмотрим группу всевозможных взаимно однозначных отображений и выделим в ней функции, коммутирующие со всеми функциями из семейства, соответствующего некоторому симметриному шифру.

Коммутирующие отображения в трёхэтапном протоколе

Введем основные определения теории групп.

Определение 1. Множество G с бинарной операцией $\cdot$ называется *группой*, если:

1. Операция *ассоциативна*, т.е. $(ab)c = a(bc)$ для любых a, b, c из G.
2. Операция гарантирует единицу, т.е. в G существует такой элемент e — он называется *единицей*, — что $ae = ea = a$ для любого a из G.
3. Операция гарантирует обратные элементы, т.е. для любого a из G существует такой элемент $x \in G$ — он называется *обратным* к a, — что $ax = xa = e$.

Пусть $N = \{1, 2, \ldots, n\}$, где $n \geq 1$.

Определение 2. Перестановкой π называется взаимно однозначное отображение $\pi : N \to N$.

Группу всех перестановок n элементов обозначим как S_n. Пусть $F : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ является взаимно однозначной функцией и G_n — группа всех таких функций с операцией суперпозиции.

Определение 3. Подгруппа K группы G называется *нормальной* тогда и только тогда, когда для $\forall f \in K$ и $\forall g \in G$ выполняется $gfg^{-1} \in K$.

Определение 4. *Центр* $Z(G)$ группы G — множество всех элементов $z \in G$ таких, что $\forall g \in G$ выполнено $gz = zg$.

Отметим, что центр $Z(G)$ является нормальной подгруппой группы G. Обозначим $C_n = Z(G_n)$.

Если в протоколе Шамира в качестве секретного преобразования Алиса выберет функцию $f \in C_n$, а Боб $g \in G_n$, то f и g будут коммутировать и алгоритм передачи сообщения m от Алисы к Бобу будет выглядеть следующим образом:

- Алиса вычисляет значение $f(m)$ и отправляет его Бобу.
- Боб вычисляет значение $g \circ f(m)$ и отправляет его Алисе.
- Алиса вычисляет значение $f^{-1} \circ g \circ f(m)$ и отправляет его Бобу.
- Боб вычисляет значение $g^{-1} \circ f^{-1} \circ g \circ f(m) = g^{-1} \circ g \circ f^{-1} \circ f(m) = m$.

G_n изоморфна симметрической группе перестановок S_{2^n}. Известно, что центр симметрической группы перестановок тривиален. Следовательно, имеет смысл рассматривать некоторое семейство взаимно однозначных функций, а не всю группу.

Заметим, что симметричный шифр можно рассматривать как параметризованное семейство взаимно однозначных отображений, где ключ является параметром. В таком случае задачу применения симметричного шифра в протоколе Шамира можно свести к следующему вопросу.

Вопрос 1. Можно ли найти такой симметричный шифр, что при фиксировании некоторого множества ключей, мы получим некоторое подмножество группы G_n, замыкание относительно операции суперпозиции которого будет иметь достаточно большой центр.

Использование известных симметричных систем в трёхэтапном протоколе

Для примера рассмотрим симметричное шифрование с помощью криптосистемы DES с одним раундом. Этот алгоритм реализуется с использованием ключа K и раундовой функции F. Раундовая функция является открытой информацией, а ключи выбираются Алисой и Бобом произвольно. В этом случае Алиса шифрует сообщение m с помощью своего ключа K_1 и отправляет Бобу значение $Y_1 = DES(m, K_1)$. Боб шифрует сообщение ещё раз со своим ключом и отправляет Алисе $Y_2 = DES(Y_1, K_2) = DES(DES(m, K_1), K_2)$. Следующим шагом Алиса должна снять шифрование с помощью своего ключа. При этом из-за нелинейности операций алгоритма мы не можем утверждать, что получившееся значение удовлетворит требованиям и при последующей расшифровке с использованием своего ключа Боб получит изначальное сообщение. Единственным решением представляется передача собственных ключей Алисы и Боба друг другу, но это противоречит условию поставленной перед нами задачи.

Использование одноразового блокнота также не является подходящим решением. Несмотря на то, что эта система удовлетворяет требованиям протокола, она не обеспечивает секретность. Алиса шифрует сообщение с помощью своего ключа K_1 и отправляет Бобу. Он в ответ отправляет сообщение, зашифрованное его ключом K_2. Теперь Алиса должна использовать свой ключ K_1 и отправить расшифрованное сообщение. Однако по первой и второй пересылке восстанавливается ключ K_2. А затем с его помощью из третьей пересылки злоумышленник восстанавливает исходное сообщение.

Гомоморфное шифрование для протокола Шамира

Гомоморфное шифрование – это форма шифрования, которая позволяет пользователям выполнять определённые математические операции с зашифрованными данными без их предварительной расшифровки и получать в результате шифртекст, соответствующий этим операциям над открытым текстом.

Пусть $E_{Sym}^{K_1}$ является функцией зашифрования некоторого симметричного шифра C_{Sym} на ключе K_1, а $E_{FHE}^{K_2}$ – функция зашифрования некоторого полностью гомоморфного шифра C_{FHE} (не уточняем пока симметричного или асимметричного) на ключе K_2. При этом преобразования битов открытого текста в биты шифртекста для C_{Sym} можно выразить с помощью операций, для которых C_{FHE} является гомоморфным. Благодаря этому, имея $E_{FHE}^{K_2}(E_{Sym}^{K_1}(m))$ и $E_{FHE}^{K_2}(K_1)$, можно выразить $E_{FHE}^{K_2}(m)$, где m – открытый текст. Иными словами, возможно расшифровать C_{Sym} внутри гомоморфного C_{FHE}. Пример представления алгоритмов шифрования упрощенного Кузнечика в виде КНФ для такого расшифрования можно найти в работе [2]. Отметим, что в указанной статье гомоморфное шифрование предполагается асимметричным, но это требование важно для представленного там протокола и не влияет на выкладки, поэтому его можно убрать.

Гомоморфное шифрование выглядит применимым для встраивания в схему Шамира в качестве одного из шифров, так как позволяет снять внутреннее шифрование, не снимая само гомоморфное. При этом нам требуется, чтобы оно было симметричным. В случае использования асимметричного шифра теряется смысл трёхэтапного протокола Шамира, так как можно передать сообщение за одну пересылку с использованием этого асимметричного шифра. Примеры симметричных полностью гомоморфных шифров можно найти в [3, 4].

В таком случае первые шаги протокола будут включать следующее:

- Алиса вычисляет $E_{Sym}^{K_1}(m)$ и отправляет Бобу.
- Боб вычисляет $E_{FHE}^{K_2}(E_{Sym}^{K_1}(m))$ и отправляет Алисе.

Далее Алисе необходимо вычислить $E_{FHE}^{K_2}(m)$. Как упоминалось выше, чтобы для этой задачи использовать свойства гомоморфного шифра, Алисе необходимо знать $E_{FHE}^{K_2}(K_1)$. Однако тогда либо Алиса должна знать ключ K_2 (или как минимум какую-то информацию о нем) и сама вычислять $E_{FHE}^{K_2}(K_1)$, либо Боб должен знать ключ K_1 и вычислять шифртекст $E_{FHE}^{K_2}(K_1)$, чтобы затем отправить его Алисе. Оба предположения противоречат идее трёхэтапного протокола Шамира, согласно которой стороны шифруют данные на своих ключах, информацией о которых не делятся друг с другом.

Таким образом, указанную выше возможность снятия внутреннего шифрования нельзя использовать для применения полного гомоморфного шифрования в трёхэтапном протоколе Шамира.

Трёхэтапный протокол на основе перестановок

В этой секции мы представим успешное обобщение протокола Шамира, которое было представлено в 2020 году [1]. Все операции данного протокола проводятся над перестановками.

Пусть S_M — множество перестановок на множестве $X = \{0, \ldots, M-1\}$.

- Алиса и Боб выбирают перестановку $\alpha \in S_M$, которая будет являться открытым ключом. Кроме того, им известно разложение данной перестановки в произведение непересекающихся циклов $\alpha = \prod_{i=1}^{n(\alpha)} \alpha_i$.

- Алиса генерирует случайный вектор $\bar{s} = (s_1, \ldots, s_{n(\alpha)})$ размерности $n(\alpha)$. Более того, $1 \leqslant s_i \leqslant \ell(\alpha_i) - 1$, где $\ell(\alpha_i)$ — порядок цикла α_i в разложении в произведение непересекающихся циклов $\alpha = \prod\limits_{i=1}^{n(\alpha)} \alpha_i$. Затем Алиса формирует перестановку $\sigma_A = \prod\limits_{i=1}^{n(\alpha)} \alpha_i^{s_i}$. Она также вычисляет обратную перестановку $\sigma_A^{-1} = \prod\limits_{i=1}^{n(\alpha)} (\alpha_{n(\alpha)+1-i}^{s_{n(\alpha)+1-i}})^{-1}$. Вектор $\bar{x}$ и перестановки σ_A, σ_A^{-1} держатся в секрете.

- Аналогичным образом Боб формирует $\bar{r}$, σ_B и σ_B^{-1}. Помимо этого, он дополнительно генерирует случайную перестановку $\chi_B \in S_M$. Она должна удовлетворять следующему условию: все непересекающиеся циклы χ_{jB} в разложении перестановки $\chi_B = \prod\limits_{j=1}^{n(\chi_B)} \chi_{jB}$ должны иметь разную длину, т.е. $\ell(\chi_{iB}) \neq \ell(\chi_{jB}))$ для $\forall i \neq j$. Боб хранит $\bar{r}$, σ_B, σ_B^{-1} и χ_B в секрете.

- Алиса хочет передать сообщение m. Этому отображению ставится в соответствие перестановка $\pi \in S_M$. Отметим, что для этого требуется, чтобы $2^k \leqslant M!$, где k количество бит в сообщение m. Алиса генерирует шифртекст $Y_1 = \sigma_A \cdot \pi$ и отправляет его Бобу.

- Получив шифртекст Y_1, Боб вычисляет $Y_2 = \sigma_B \cdot Y_1 \cdot \chi_B$ и отправляет его Алисе.

- Алиса вычисляет перестановку π^{-1} обратную к π. Затем Алиса «снимает» свой ключ с полученного шифртекста Y_2 вычисляя $Y_3 = \sigma_A^{-1} \cdot Y_2 \cdot \pi^{-1}$. Затем она отправляет Y_3 Бобу.

- Боб получает Y_3 и «снимает» свой секретный ключ σ_B вычисляя $Y_4 = \sigma_B^{-1} \cdot Y_3$. Затем Боб раскладывает Y_4 в произведение непересекающихся циклов $Y_4 = \prod\limits_{k=1}^{n(Y_4)} Y_{k4}$.

Если мы представим циклы χ_{jB} как $\chi_{jB} = (\chi_{1jB}, \ldots, \chi_{\ell(\chi_{jB})jB})$, то перестановку π можно восстановить из перестановки $Y_4 = \prod\limits_{j=1}^{n(\chi)} \big(\pi(\chi_{1jB}), \ldots, \pi(\chi_{\ell(\chi_{jB})jB})\big)$.

Последнее выражение определяет перестановку π. Стоит отметить, что после восстановления π Боб может получить секретный ключ Алисы $\sigma_A = Y_1\pi^{-1}$. Следовательно, секретный ключ нужно менять после каждого применения протокола.

Заключение

В рамках работы были исследованы возможности использования симметричных шифров и полного гомоморфного шифрования в трёхэтапном протоколе Шамира. Было показано, что один из подходов к поиску шифров, которые могут быть использованы в протоколе, – изучение центров подгрупп группы всех взаимно однозначных отображений. Кроме того, показано, что использование полного гомоморфного шифрования в трёхэтапном протоколе Шамира нецелесообразно.

ЛИТЕРАТУРА

[1] S. Anatoly, F. Emil and L. Olha. Three-Pass Cryptographic Protocol Based on Permutations // ATIT 2020 - Proceedings: 2020 2nd International Conference on Advanced Trends in Information Theory. 2020. P. 281–284.

[2] Л. К. Бабенко, Е.А. Толоманенко. Гибридное шифрование на основе использования симметричных и гомоморфных шифров // Известия ЮФУ. Технические науки. 2021. №2(219). С. 6–18.

[3] Ф. Б. Буртыка. Симметричное полностью гомоморфное шифрование с использованием неприводимых матричных полиномов // Известия ЮФУ. Технические науки. 2014. №8(157). С. 107–122.

[4] K. Hariss, H. Noura, A.E. Samhat. An efficient fully homomorphic symmetric encryption algorithm // Multimedia Tools and Applications. 2020. V. 79. P. 12139–12164.

Кураторы исследования –
к.ф.-м.н., с.н.с. Института математики им. С. Л. Соболева СО РАН, Наталья Николаевна Токарева;
аспирант Института математики им. С.С. Соболева СО РАН, Матвей Андреевич Панферов;
бакалавр ММФ НГУ, лаборант Института математики им.С.Л.Соболева СО РАН, Денис Александрович Быков;
аспирант ММФ НГУ, м.н.с Института математики им.С.Л.Соболева СО РАН, Александр Сергеевич Шапоренко.

Разработка симметричной криптосистемы и её криптоанализ

Л. К. Бояндин[1]
[1]СУНЦ НГУ
E-mail: l.boyandin@school.nsu.ru

Аннотация

В данной работе представлена конструкция симметричного блочного итерационного шифра с использованием криптографических примитивов на основе сети Фейстеля. Проведен алгебраический криптоанализ данной криптосистемы (полный анализ одного раунда и частичный для двух раундов, однако, системы булевых уравнений полностью составлены в обоих случаях). Также в работе показано, что тесты на случайность демонстрируют положительный результат, а проведение криптоанализ заметно усложняется с увеличением количества раундов.

Ключевые слова: *блочный шифр, симметричный шифр, сеть Фейстеля, расписание ключей, алгебраический криптоанализ*

Симметричная шифрсистема — система шифрования, в которой ключи зашифрования и расшифрования совпадают, либо легко определяются один по другому[5].

Симметричные шифрсистемы более стойкие при меньших размерах ключа, чем асимметричные (для того, чтобы из открытого ключа асимметричного шифра было бы трудно получить закрытый ключ, ключи должны иметь относительно большие размеры). Также они быстрее работают (асимметричные криптосистемы, такие, как RSA, используют дорогостоящие операции (умножение и возведение в степень по модулю)).

Блочный шифр состоит из нескольких *раундов*, на каждом из которых происходит обратимое преобразование блока длины n в новый блок той же длины. Выходной блок i-го раунда называется *промежуточным шифртекстом* и обозначается C_i. Он является входным блоком следующего $i+1$-го раунда. Входным блоком первого раунда служит $C_0 = P$. *Шифртекстом* (ciphertext) C называется выходной блок последнего раунда, $C = C_r$. Число раундов r (на практике, от 8 до 64) должно быть, с одной стороны, достаточно большим для обеспечения высокой криптостойкости шифра, а с другой стороны, достаточно малым для того, чтобы шифрование было быстрым. Часто на всех раундах шифра используется одно и то же преобразование, зависящее лишь от разных битов ключа. А именно из ключа формируется r *подключей* $K_0, \ldots, K_{r-1}$, каждый из которых используется на определённом раунде[5].

Сетью Фейстеля (рис.1). называется блочный шифр, устроенный следующим образом. Пусть входной блок C_i каждого раунда делится на левую L_i и правую R_i половины. Тогда блок C_{i+1} получается объединением $L_{i+1} = R_i$ и $R_{i+1} = L_i \oplus F(R_i, K_i)$, где F — *раундовая функция* и K_i — подключ данного раунда. Сеть Фейстеля примечательна тем, что при её расшифровании не требуется обращать раундовую функцию F, т. е. находить функцию F^{-1}. Действительно, значения L_i и R_i могут быть восстановлены как $L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$ и $R_i = L_{i+1}$. Это свойство существенно. Оно означает, что в сети Фейстеля функция F может быть устроена как угодно сложно (обратная функция F^{-1} может д˙аже не существовать!), что тем не менее не усложняет процедуру расшифрования по сравнению с шифрованием. Примерами сетей Фейстеля являются шифры DES, ГОСТ 28147-89, TwoFish, RC-6, MARS, Camellia[5].

Шифр, сконструируемый в ходе работы (далее - *123F*), основан на данной схеме из-за её простоты и отсутствия необходимости в обратимости раундовой функции. Ключ имеет длину 64 бит, блоки - 32, в шифре 32 раунда. Последняя характеристика была выбрана из тестирования, проведённого при помощи тестировочной программы NIST (в качестве блоков подавался "счётчик числа

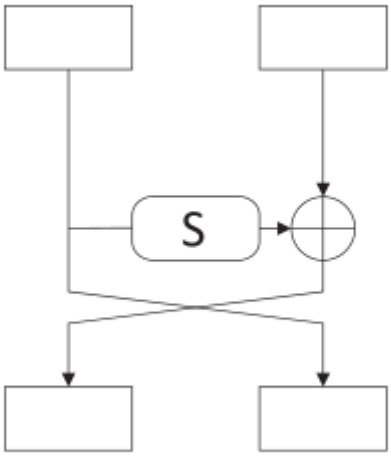

Рис. 1: Сеть Фейстеля

от 0 до $2^{32}-1$, и выход проверялся на случайность): требовалось получить максимально случайный шифртекст от максимально неслучайного открытого текста. При ключе, равном 6576, все тесты прошли успешно. Логические операции дают нелинейность, XOR - вполне распространённая операция для прямого добавления битов ключа. Циклический сдвиг, расположенный перед логическими операциями, даёт возможность не использовать биты одной и той же позиции в этих операциях. Номер раунда используется в расписании ключей как счётчик для более значительного изменения битов ключа. Также проводилось исследование "лавинного эффекта того, как меняется выход при изменении одного бита входа (конкретнее, расстояние Хэмминга между выходами). Среднее значение оказалось равным 0.509766 при ключе, равном 0, 0.492188 при ключе, равном 6576, 0.489258 при ключе, равном 12979524099108840000, 0.494141 при ключе, равном 9263590764808563000. Все результаты близки к 0.5.

Проведём алгебраический криптоанализ данного шифра. Составим систему булевых уравнений для одного раунда текста:

$$(1) \quad L_{i+1}[k] = R_i[k].$$

$$R_{i+1}[0] = (K_{2_i}[0]R_i[6]) \oplus (R_i[11] \oplus (K_{1_i}[0] \oplus R_i[0]) \oplus (K_{1_i}[0] \oplus R_i[0])R_i[11]) \oplus L_i[0]$$

$$R_{i+1}[1] = (K_{2_i}[1]R_i[7]) \oplus (R_i[12] \oplus (K_{1_i}[1] \oplus R_i[1]) \oplus (K_{1_i}[1] \oplus R_i[1])R_i[12]) \oplus L_i[1] \oplus J_1.$$

Здесь и далее вводятся следующие переменные, отвечающие за добавочный бит при сложении по модулю:

$$J_{0_i} = 0$$

и

$$J_{k_i} = J_{k-1_i}R_i[((k-1)+6) \mod 16]K_{2_i}[k-1] \oplus J_{k-1_i}(R_i[((k-1)-5) \mod 16](K_{1_i}[k-1] \oplus R_i[k-1])$$

$$\oplus (K_{1_i}[k-1] \oplus R_i[k-1]) \oplus R_i[((k-1)-5) \mod 16]) \oplus R_i[((k-1)+6) \mod 16]K_{2_i}[k-1]$$

$$(R_i[((k-1)-5) \mod 16](K_{1_i}[k-1] \oplus R_i[k-1]) \oplus (K_{1_i}[k-1] \oplus R_i[k-1]) \oplus R_i[((k-1)-5) \mod 16]) \Leftrightarrow$$

$$\Leftrightarrow (2) \quad J_{k_i} = J_{k-1_i}R_i[(k+5) \mod 16]K_{2_i}[k-1] \oplus J_{k-1_i}(R_i[(k-6) \mod 16](K_{1_i}[k-1] \oplus R_i[k-1])$$

$$\oplus (K_{1_i}[k-1] \oplus R_i[k-1]) \oplus R_i[(k-6) \mod 16]) \oplus R_i[(k+5) \mod 16]K_{2_i}[k-1]$$

$$(R_i[(k-6) \mod 16](K_{1_i}[k-1] \oplus R_i[k-1]) \oplus (K_{1_i}[k-1] \oplus R_i[k-1]) \oplus R_i[(k-6) \mod 16])$$

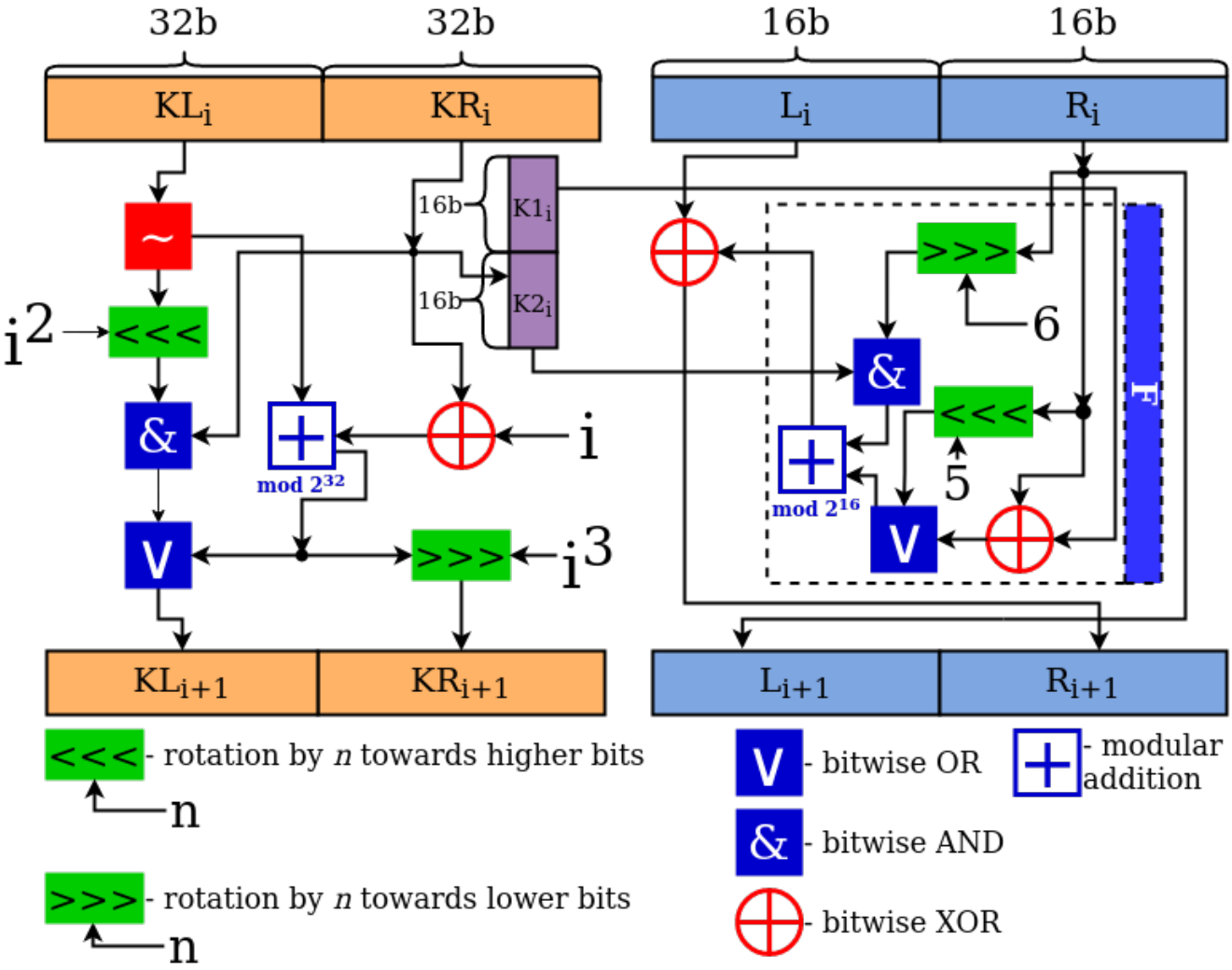

Рис. 2: *Один раунд шифра 123F*

для $k \geq 1$. Аналогично до $k = 15$ получаем уравнения вида

$$(3) \quad R_{i+1}[k] = (K_{2_i}[k]R_i[(k+6) \mod 16]) \oplus (R_i[(k-5) \mod 16] \oplus$$

$$\oplus (K_{1_i}[k] \oplus R_i[k]) \oplus (K_{1_i}[k] \oplus R_i[k])R_i[(k-5) \mod 16]) \oplus L_i[k] \oplus J_{k_i}.$$

Допустим, что шифр состоит из одного раунда. Таким образом мы можем найти K_{R_0}, т.е. старшие биты ключа. Выберем случайным образом ключ, равный 3219841561881812000 (случайное число от 0 до 1, умноженное на $2^{64} - 1$). В двоичном представлении он равен 0010110010101111001011001101011101010001000000010001110000100000. Представим, что мы не знаем этот ключ.

Подадим на вход $P_a = 0$ (используя программную реализацию на языке C++). На выходе получаем $C_a = 00101100110101110000000000000000$. Так как $L_0[k] = R_0[k] = 0$, (1) превращается в

$$(4) \quad J_{k_0} = J_{k-1_0}(K_{1_0}[k-1] \oplus R_0[k-1])$$

(Заметим, что тогда по индукции все $J_{k_0} = 0$ т.к. $J_{0_0} = 0$) а (2) переходит в

$$R_1[k] = K_{1_0}[k].$$

Теперь мы знаем K_{1_0}, младшие 16 бит старших 32 бит ключа, полностью. Действительно,

$$0010110011010111 = 0010110011010111.$$

Далее необходимо найти K_{2_0}. Заметим, что по (2) для этого необходимо рассмотреть такие k, при которых $R_i[(k+6) \mod 16] = 1$. Можно попробовать рассмотреть

$$P_b = 11111111111111110000000000000000_2 = 4294901760_{10}.$$

Тогда равенство выполняется при каждом $k \in \{0, \ldots, 15\}$. На выходе получаем

$$C_b = 00101100101011101111111111111111.$$

В этом случае (2) превращается в

$$J_{k_0} = J_{k-1_0} K_{2_0}[k-1] \oplus J_{k-1_0} \oplus K_{2_0}[k-1]$$

Тогда получаем, что

$$J_1 = K_{2_0}[0].$$

(3) при этом превращается в

$$(7) \quad R_1[k] = K_{2_0}[k] \oplus 1 \oplus J_{k_0}.$$

Так как $J_0 = 0$, имеем $R_1[0] = K_{2_0}[0] \oplus 0 \oplus 1 \Leftrightarrow 0 = K_{2_0}[0] \oplus 1 \Leftrightarrow K_{2_0}[0] = 1$. Тогда $J_1 = 1$. Заметим, что (6) есть ни что иное, как алгебраическая нормальная форма дизъюнкции, т.е. это выражение эквивалентно $J_k = J_{k-1} \vee K_{2_0}[k-1]$. Тогда методом мат. индукции по k можно доказать, что $\forall k \geq 1 \;\; J_k = 1$: $\square$ Равенство $J_1 = 1$ выведено несколькими строчками выше (база). Если для произвольного $k > 1$ $J_k = 1$, то $J_{k+1} = 1 \vee K_{2_0}[k] = 1$ (индукционный переход). $\blacksquare$ Но с помощью этого факта (7) значительно упрощается, а именно:

$$\forall k \geq 1 \;\; R_1[k] = K_{2_0}[k] \oplus 1 \oplus 1 \Leftrightarrow R_1[k] = K_{2_0}[k].$$

Сравнив $R_1 = 0010110010101110$ и $K_{2_0} = 0010110010101111$, убедимся, что мы правильно определили K_{2_0}, а значит, и весь K_{R_0}.

Теперь представим, что шифр имеет два раунда, и одновременно "забудем"и промежуточный шифртекст, и ту половину ключа, которую мы только что нашли.

Снова подадим на вход $P_a = 0_{10}$. На выходе получаем $C_a = 00010111100001110010110011010111$. Тогда для $i = 0$ (4) остаётся тем же, а (3) переходит в

$$(8) \quad R_1[k] = K_{1_0}[k].$$

Также

$$L_1[k] = R_0[k].$$

Для $i = 1$

$$J_{k_1} = J_{k-1_1} R_1[(k+5) \mod 16] K_{2_1}[k-1] \oplus J_{k-1_1}(R_1[(k-6) \mod 16](K_{1_1}[k-1] \oplus R_1[k-1])$$

$$\oplus (K_{1_1}[k-1] \oplus R_1[k-1]) \oplus R_1[(k-6) \mod 16]) \oplus R_1[(k+5) \mod 16] K_{2_1}[k-1]$$

$$(R_1[(k-6) \mod 16](K_{1_1}[k-1] \oplus R_1[k-1]) \oplus (K_{1_1}[k-1] \oplus R_1[k-1]) \oplus R_1[(k-6) \mod 16])$$

$$R_2[k] = (K_{2_1}[k] R_1[(k+6) \mod 16]) \oplus (R_1[(k-5) \mod 16] \oplus$$

$$\oplus (K_{1_1}[k] \oplus R_1[k]) \oplus (K_{1_1}[k] \oplus R_1[k]) R_1[(k-5) \mod 16]) \oplus L_1[k] \oplus J_{k_1}.$$

$$(9) \quad L_2[k] = R_1[k].$$

Из (8) и (9) получаем

$$(10) \quad K_{1_0}[k] = L_2[k].$$

Из (9) получаем

$$(11) \quad J_{k_1} = J_{k-1_1} L_2[(k+5) \mod 16] K_{2_1}[k-1] \oplus J_{k-1_1} (L_2[(k-6) \mod 16](K_{1_1}[k-1] \oplus L_2[k-1])$$

$$\oplus (K_{1_1}[k-1] \oplus L_2[k-1]) \oplus L_2[(k-6) \mod 16]) \oplus L_2[(k+5) \mod 16] K_{2_1}[k-1]$$

$$(L_2[(k-6) \mod 16](K_{1_1}[k-1] \oplus L_2[k-1]) \oplus (K_{1_1}[k-1] \oplus L_2[k-1]) \oplus L_2[(k-6) \mod 16])$$

$$(12) \quad R_2[k] = (K_{2_1}[k] L_2[(k+6) \mod 16]) \oplus (L_2[(k-5) \mod 16] \oplus$$

$$\oplus (K_{1_1}[k] \oplus L_2[k]) \oplus (K_{1_1}[k] \oplus L_2[k]) L_2[(k-5) \mod 16]) \oplus J_{k_1}.$$

Теперь подадим на вход $P_b = 11111111111111111111111111111111$, получив на выходе $C_b = 11010000111101001101001101010001$. Теми же останутся уравнения $L_1[k] = R_0[k](= 1)$ и $L_2[k] = R_1[k]$. Для $i = 0$

$$(13) \quad J_{k_0} = J_{k-1_0} K_{2_0}[k-1] \oplus J_{k-1_0} \oplus K_{2_0}[k-1],$$

$$\text{из чего } J_{1_0} = K_{2_0}[0],$$

$$(14) \quad L_2[k] = K_{2_0}[k] \oplus J_{k_0}.$$

для $i = 1$ (11) не изменяет свой вид,

$$(15) \quad R_2[k] = (K_{2_1}[k] L_2[(k+6) \mod 16]) \oplus (L_2[(k-5) \mod 16] \oplus$$

$$\oplus (K_{1_1}[k] \oplus L_2[k]) \oplus (K_{1_1}[k] \oplus L_2[k]) L_2[(k-5) \mod 16]) \oplus 1 \oplus J_{k_1}.$$

Подадим на вход $P_c = 00000000000000001111111111111111$, получив на выходе $C_c = 0110101010101010\ 1101001100101000$. Теми же останутся уравнения $L_1[k] = R_0[k](= 0)$ и $L_2[k] = R_1[k]$. Для $i = 0$

$$(16) \quad J_{k_0} = J_{k-1_0} K_{1_i}[k-1] = 0$$

$$(17) \quad L_2[k] = K_{1_0}[k] \oplus 1;$$

для $i = 1$ (11) и (12) останутся теми же.

Подадим на вход $P_d = 11111111111111110000000000000000$, получив $C_d = 1010100110000000010110010101110$. Теми же останутся уравнения $L_1[k] = R_0[k](= 1)$ и $L_2[k] = R_1[k]$. Для $i = 0$ (13) остаётся тем же (как и $J_{1_0} = K_{2_0}[0]$),

$$(19) \quad L_2[k] = K_{2_0}[k] \oplus 1 \oplus J_{k_0}.$$

для $i = 1$ (15) и (11) остаются теми же.

По (10) сразу находим $K_{1_0} = 0010110011010111$. Рассмотрим (19): для C_d $L_2 = 0010110010101110$, а значит, т.к. $J_{0_0} = 0$, $K_{2_0}[0] = L_2[0] \oplus 1 = 0 \oplus 1 = 1$. Тогда $J_{1_0} = 1$, а (13) эквивалентно $J_{k_0} = J_{k-1_0} \vee K_{2_0}[k-1]$, и тогда мы, абсолютно аналогично рассуждению из предыдущего случая, приходим к тому, что

$$\forall k \geq 1 \;\; L_2[k] = K_{2_0}[k] \oplus 1 \oplus 1 \Leftrightarrow L_2[k] = K_{2_0}[k].$$

Сравнив $L_2 = 0010110010101110$ и $K_{2_0} = 0010110010101111$, убеждаемся, что данная часть ключа определена правильно. Таким образом мы определили $K_{R_0} = 00101100101011110010110011010111$. Следующим шагом будет определение K_{R_1}, состоящего из K_{1_1} и K_{2_1}. Полезными для нас в данном

случае оказываются системы (11) и $(15)/(12)$. Начнём с $P_c = 00000000000000001111111111111111$. Пользуясь программой, получаем уравнения для J_{k1}:

$$J_{1_1} = J_{0_1} K_{1_1}[0],$$

$$J_{2_1} = J_{1_1} K_{1_1}[1],$$

$$J_{3_1} = J_{2_1} K_{2_1}[2] \oplus J_{2_1} K_{1_1}[2] \oplus K_{2_1}[2] K_{1_1}[2],$$

$$J_{4_1} = J_{3_1} K_{2_1}[3] \oplus J_{3_1}(K_{1_1}[3] \oplus 1) \oplus K_{2_1}[3](K_{1_1}[3] \oplus 1),$$

$$J_{5_1} = J_{4_1} K_{1_1}[4],$$

$$J_{6_1} = J_{5_1}(K_{1_1}[5] \oplus 1),$$

$$J_{7_1} = J_{6_1} K_{2_1}[6] \oplus J_{6_1} K_{1_1}[6] \oplus K_{2_1}[6] K_{1_1}[6],$$

$$J_{8_1} = J_{7_1} K_{1_1}[7],$$

$$J_{9_1} = J_{8_1} K_{2_1}[8] \oplus J_{8_1} \oplus K_{2_1}[8],$$

$$J_{10_1} = J_{9_1} K_{2_1}[9] \oplus J_{9_1}(K_{1_1}[9] \oplus 1) \oplus K_{2_1}[9](K_{1_1}[9] \oplus 1),$$

$$J_{11_1} = J_{10_1},$$

$$J_{12_1} = J_{11_1} K_{1_1}[11],$$

$$J_{13_1} = J_{12_1}(K_{1_1}[12] \oplus 1),$$

$$J_{14_1} = J_{13_1} K_{2_1}[13] \oplus J_{13_1} \oplus K_{2_1}[13],$$

$$J_{15_1} = J_{14_1}.$$

Вторая система:

$$0 = K_{1_1}[0] \oplus J_{0_1},$$

$$1 = K_{1_1}[1] \oplus J_{1_1},$$

$$0 = K_{2_1}[2] \oplus K_{1_1}[2] \oplus J_{2_1},$$

$$1 = K_{2_1}[3] \oplus (K_{1_1}[3] \oplus 1) \oplus J_{3_1} \Rightarrow 0 = K_{2_1}[3] \oplus K_{1_1}[3] \oplus J_{3_1},$$

$$0 = K_{1_1}[4] \oplus J_{4_1} \Rightarrow K_{1_1}[4] = J_{4_1},$$

$$1 = (K_{1_1}[5] \oplus 1) \oplus J_{5_1} \Rightarrow 0 = K_{1_1}[5] \oplus J_{5_1} \Rightarrow K_{1_1}[5] = J_{5_1},$$

$$0 = K_{2_1}[6] \oplus K_{1_1}[6] \oplus J_{6_1},$$

$$1 = K_{1_1}[7] \oplus J_{7_1},$$

$$0 = K_{2_1}[8] \oplus 1 \oplus J_{8_1} \Rightarrow 1 = K_{2_1}[8] \oplus J_{8_1},$$

$$1 = K_{2_1}[9] \oplus (K_{1_1}[9] \oplus 1) \oplus J_{9_1} \Rightarrow 0 = K_{2_1}[9] \oplus K_{1_1}[9] \oplus J_{9_1},$$

$$0 = 1 \oplus J_{10_1} \Rightarrow J_{10_1} = 1,$$

$$1 = K_{1_1}[11] \oplus J_{11_1},$$

$$0 = (K_{1_1}[12] \oplus 1) \oplus J_{12_1} \Rightarrow 1 = K_{1_1}[12] \oplus J_{12_1},$$

$$1 = K_{2_1}[13] \oplus 1 \oplus J_{13_1} \Rightarrow 0 = K_{2_1}[13] \oplus J_{13_1},$$

$$1 = 1 \oplus J_{14_1} \Rightarrow J_{14_1} = 0,$$

$$0 = K_{2_1}[15] \oplus (K_{1_1}[15] \oplus 1) \oplus J_{15_1} \Rightarrow 1 = K_{2_1}[15] \oplus K_{1_1}[15] \oplus J_{15_1},$$

Так как $J_{0_1} = 0$, то

$$K_{1_1}[0] = 0,$$

$$J_{1_1} = 0 \Rightarrow K_{1_1}[1] = 1,$$

$$J_{2_1} = 0 \Rightarrow 0 = K_{2_1}[2] \oplus K_{1_1}[2] \Rightarrow K_{2_1}[2] = K_{1_1}[2],$$

$$J_{3_1} = K_{2_1}[2]K_{1_1}[2] \Rightarrow 0 = K_{2_1}[3] \oplus K_{1_1}[3] \oplus K_{2_1}[2]K_{1_1}[2].$$

Так как $J_{10_1} = 1$, то

$$J_{11_1} = 1 \Rightarrow K_{1_1}[11] = 0,$$

$$J_{12_1} = K_{1_1}[11] = 0 \Rightarrow K_{1_1}[12] = 1,$$

$$J_{13_1} = 0 \Rightarrow K_{2_1}[13] = 0.$$

Так как $J_{14_1} = 0$, то

$$J_{15_1} = 0 \Rightarrow 1 = K_{2_1}[15] \oplus K_{1_1}[15] \Rightarrow \left[\begin{array}{l} \begin{cases} K_{1_1}[15] = 0 \\ K_{2_1}[15] = 1 \end{cases} \\ \begin{cases} K_{1_1}[15] = 1 \\ K_{2_1}[15] = 0 \end{cases} \end{array}\right.$$

Так как $1 = K_{1_1}[7] \oplus J_{7_1}$, то либо $K_{1_1}[7] = 0$ и $J_{7_1} = 1$, либо $K_{1_1}[7] = 1$ и $J_{7_1} = 0$. В обоих случаях произведение $J_{7_1}K_{1_1}[7] = 0$, но тогда

$$J_{8_1} = 0 \Rightarrow K_{2_1}[8] = 1,$$

$$J_{9_1} = K_{2_1}[8] = 1 \Rightarrow 1 = K_{2_1}[9] \oplus K_{1_1}[9] \Rightarrow \left[\begin{array}{l} \begin{cases} K_{1_1}[9] = 0 \\ K_{2_1}[9] = 1 \end{cases} \\ \begin{cases} K_{1_1}[9] = 1 \\ K_{2_1}[9] = 0 \end{cases} \end{array}\right.$$

Так как $K_{1_1}[5] = J_{5_1}$, то либо

$$\begin{cases} K_{1_1}[5] = 0 \\ J_{5_1} = 0 \end{cases} \Rightarrow J_{6_1} = 0 \times (0 \oplus 1) = 0,$$

либо

$$\begin{cases} K_{1_1}[5] = 1 \\ J_{5_1} = 1 \end{cases} \Rightarrow J_{6_1} = 1 \times (1 \oplus 1) = 1 \times 0 = 0.$$

В обоих случаях $J_{6_1} = 0$, но тогда

$$J_{7_1} = K_{2_1}[6]K_{1_1}[6],$$

$$0 = K_{2_1}[6] \oplus K_{1_1}[6].$$

При этом

$$1 = K_{1_1}[7] \oplus K_{2_1}[6]K_{1_1}[6].$$

Но тогда возможны два случая:

$$\left[\begin{array}{l} \begin{cases} K_{2_1}[6] = 0 \\ K_{1_1}[6] = 0 \\ K_{1_1}[7] = 1 \end{cases} \\ \begin{cases} K_{2_1}[6] = 1 \\ K_{1_1}[6] = 1 \\ K_{1_1}[7] = 0 \end{cases} \end{array}\right.$$

Так как $K_{2_1}[2] = K_{1_1}[2]$, то либо

$$\begin{cases} K_{1_1}[2] = 0 \\ K_{2_1}[2] = 0 \end{cases} \Rightarrow J_{3_1} = 0 \Rightarrow \begin{cases} J_{4_1} = K_{2_1}[3](K_{1_1}[3] \oplus 1) \\ K_{2_1}[3] = K_{1_1}[3] \end{cases} \Rightarrow J_{4_1} = 0 \Rightarrow \begin{cases} K_{1_1}[4] = 0 \\ J_{5_1} = 0 \end{cases} \Rightarrow \begin{cases} K_{1_1}[4] = 0 \\ K_{1_1}[5] = 0 \end{cases},$$

либо

$$\begin{cases} K_{1_1}[2] = 1 \\ K_{2_1}[2] = 1 \end{cases} \Rightarrow J_{3_1} = 1 \Rightarrow \begin{cases} J_{4_1} = K_{2_1}[3] \oplus (K_{1_1}[3] \oplus 1) \oplus K_{2_1}[3](K_{1_1}[3] \oplus 1) \\ K_{2_1}[3] \oplus K_{1_1}[3] = 1 \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} J_{4_1} = K_{2_1}[3] \vee (K_{1_1}[3] \oplus 1) \\ \left[\begin{array}{l} \begin{cases} K_{2_1}[3] = 0 \\ K_{1_1}[3] = 1 \end{cases} \\ \begin{cases} K_{2_1}[3] = 1 \\ K_{1_1}[3] = 0 \end{cases} \end{array} \right. \end{cases} \Rightarrow J_{4_1} = 1 \Rightarrow \begin{cases} K_{1_1}[4] = 1 \\ J_{5_1} = 1 \end{cases} \Rightarrow \begin{cases} K_{1_1}[4] = 1 \\ K_{1_1}[5] = 1 \end{cases}.$$

Тогда

$$\left[\begin{array}{l} \begin{cases} K_{1_1}[2] = 0 \\ K_{2_1}[2] = 0 \\ K_{1_1}[4] = 0 \\ K_{1_1}[5] = 0 \\ K_{2_1}[3] = K_{1_1}[3] \end{cases} \\ \begin{cases} K_{1_1}[2] = 1 \\ K_{2_1}[2] = 1 \\ K_{1_1}[4] = 1 \\ K_{1_1}[5] = 1 \\ K_{2_1}[3] \oplus K_{1_1}[3] = 1 \end{cases} \end{array} \right..$$

Итого

$$\begin{cases} K_{1_1}[0] = 0 \\ K_{1_1}[1] = 1 \\ K_{1_1}[11] = 0 \\ K_{1_1}[12] = 1 \\ K_{2_1}[8] = 1 \\ K_{2_1}[13] = 0 \end{cases}, \left[\begin{array}{l} \begin{cases} K_{2_1}[6] = 0 \\ K_{1_1}[6] = 0 \\ K_{1_1}[7] = 1 \end{cases} \\ \begin{cases} K_{2_1}[6] = 1 \\ K_{1_1}[6] = 1 \\ K_{1_1}[7] = 0 \end{cases} \end{array} \right., \left[\begin{array}{l} \begin{cases} K_{1_1}[2] = 0 \\ K_{2_1}[2] = 0 \\ K_{1_1}[4] = 0 \\ K_{1_1}[5] = 0 \\ K_{2_1}[3] = K_{1_1}[3] \end{cases} \\ \begin{cases} K_{1_1}[2] = 1 \\ K_{2_1}[2] = 1 \\ K_{1_1}[4] = 1 \\ K_{1_1}[5] = 1 \\ K_{2_1}[3] \oplus K_{1_1}[3] = 1 \end{cases} \end{array} \right., \left[\begin{array}{l} \begin{cases} K_{1_1}[15] = 0 \\ K_{2_1}[15] = 1 \end{cases} \\ \begin{cases} K_{1_1}[15] = 1 \\ K_{2_1}[15] = 0 \end{cases} \end{array} \right., \left[\begin{array}{l} \begin{cases} K_{1_1}[9] = 0 \\ K_{2_1}[9] = 1 \end{cases} \\ \begin{cases} K_{1_1}[9] = 1 \\ K_{2_1}[9] = 0 \end{cases} \end{array} \right..$$

Для справки, $K_{1_1} = 0001000010110110$, $K_{2_1} = 1101101110101110$. Составим систему уравнений для битов ключа:

$$K_{R_{i+1}}[j] = K_{R_i}[(j + i^3) \mod 32] \oplus i \oplus K_{L_i}[(j + i^3) \mod 32] \oplus 1 \oplus G_j$$

$$K_{L_{i+1}}[j] = (K_{R_i}[j] \oplus i \oplus K_{L_i}[j] \oplus 1 \oplus G_j)K_{R_i}[j](K_{L_i}[(j - i^2) \mod 32] \oplus 1) \oplus$$
$$\oplus K_{R_i}[j](K_{L_i}[(j - i^2) \mod 32] \oplus 1) \oplus K_{R_i}[j] \oplus i \oplus K_{L_i}[j] \oplus 1 \oplus G_j$$

При этом $G_0 = 0$, а $G_{j+1} = G_j(K_{R_i}[j] \oplus i) \oplus G_j(K_{L_i}[j] \oplus 1) \oplus (K_{R_i}[j] \oplus i)(K_{L_i}[j] \oplus 1)$.

Подставим $i = 0$ (в данном случае - единственное используемое значение, K_{L_1} не используется):

$$K_{R_1}[j] = K_{R_0}[j] \oplus K_{L_0}[j] \oplus 1 \oplus G_j \Rightarrow K_{L_0}[j] = K_{R_1}[j] \oplus K_{R_0}[j] \oplus 1 \oplus G_j,$$

$$G_{j+1} = G_j K_{R_0}[j] \oplus (G_j \oplus K_{R_0}[j])(K_{L_0}[j] \oplus 1).$$

Как видно, криптоанализ значительно усложняется при увеличении количества раундов. Предполагается, что сложность увеличится во много раз относительно наблюдаемой у "игрушечного"варианта при увеличении раундов.

Итого, был создан симметричный итеративный блочный шифр на основе сети Фейстеля с применением логических операций, циклических сдвигов, исключающего "или"и сложения по модулю. Его выход для максимально неслучайных входных данных был протестирован на случайность при помощи программы от NIST. Был проведён алгебраический криптоанализ для одного раунда, были составлены необходимые системы булевых уравнений для проведения криптоанализа для двух раундов (и использованы для частичного выявления битов ключа).

ЛИТЕРАТУРА

[1] Токарева Н. Н. Симметричная криптография. Краткий курс.-Новосибирский государственный университет. – 2012.

Кураторы исследования –
Максимлюк Юлия Павловна.

БУЛЕВЫ ФУНКЦИИ В КРИПТОГРАФИИ

Конструкции APN-функций из векторных bent-функций

Р. И. Бархаткин, К.В. Калгин
Новосибирский государственный университет
E-mail: r.barkhatkin@g.nsu.ru

Аннотация

В блочных и поточных шифрах бент-функции и их векторные аналоги способствуют предельному повышению стойкости этих шифров к линейному, а APN-функции к дифференциальному криптоанализу. В работе исследовалась возможность построения APN функции из двух векторных бент-функций. В результате перебора на компьютере параметров конструкции Мэйорана-МакФарланда получены функции от 6 переменных.

Ключевые слова: *APN-функция, бент-функция, ортоморфизм.*

Нелинейностью булевой функции f от n переменных называется расстояние Хэмминга N_f от данной функции до множества всех афинных функций, а именно $N_f = min\ dist(f, l_{a,b})$, где $a \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2$ и $l_{a,b}(x) = \langle a, x\rangle \oplus b$ [1].

Функцию $F : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^m$ называют *векторной булевой функцей*. Нелинейностью таких функций называется минимальная из нелинейностей булевых функций $f_b = \langle b, F(y)\rangle$ при различных значениях $b \in \mathbb{Z}_2^m$, $b \neq 0$.

Векторная булева функция $F : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^n$ называется *дифференциально δ-равномерной*, если при любом ненулевом векторе a и произвольном векторе b уравнение $F(x) \oplus F(x \oplus a) = b$ имеет не более δ решений, где δ-целое число.

Бент-функция - булева функция от n переменных с максимальной нелинейностью, где n четное.

Векторными бент-функциями называются функции $F : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^m$ с максимальной нелинейностью, где $m \leq n/2$.

Векторная булева функция называется *APN-функцией*, если она дифференциально равномерна с минимальным возможным значением δ. В этом случае $\delta = 2$.

Ортоморфизм $f(x)$ - взаимно-однозначное отображение $\mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^n$ такое, что отображение $f(x) + x$ тоже является взаимно-однозчным.

Конструкция булевой бент-функции Мэйорана-МакФарланда. Булева функция от n переменных вида

$$f(x, y) = \langle x, h(y)\rangle \oplus g(y),$$

где $x, y \in \mathbb{Z}_2^{n/2}$, $h(y)$ - произвольная перестановка на множестве $\mathbb{Z}_2^{n/2}$, $g(y)$ - произвольная булева функция от $n/2$ переменных, является бент функцией.

Утверждение. Пусть векторная бент функция $F(x) = (f_1, \ldots, f_{n/2})$,где k-ая координата имеет вид $f_k(x, y) = \langle x, h_k(y)\rangle$ и h_0 - тождественная функция, тогда остальные перестановки h_k, где $k > 0$, являются ортоморфизмами.

Так как при $m \leq 5$ все ортоморфизмы линейные [2], то биективная перестановка m элементов эквивалентна действию обратимой матрицы размера $m \times m$, $h_k(x) \equiv A_k x$.

В настоящей работе осуществляется поиск APN-функций от 6 переменных, представимых в виде комбинации двух векторных бент функций $F(x, y) = (F1, F2)$. Ниже приводятся две конструкции APN-функций, найденные в ходе перебора различных матриц A_k и перестановки $g(x)$.

Конструкция 1. В этой конструкции аргументы у векторных бент-функций F_k меняются местами и используется одна векторная добавочная функция $g(x)$:

$$F(x,y) = (F_1(x,y) \oplus g(y), F_2(y,x) \oplus g(x))$$

$$F_1(x,y) = (f_1(x,y), f_2(x,y), f_3(x,y))$$
$$F_2(x,y) = (f_4(x,y), f_5(x,y), f_6(x,y))$$
$$f_k(x,y) = \langle x, A_k y \rangle,$$

где $g(x) : Z_2^{n/2} \longrightarrow Z_2^{n/2}$ - произвольная функция, $A_1 = A_4 = E$ - тождественные преобразования, $A_2, A_3, A_2 + A_3, A_5, A_6, A_5 + A_6$ - ортоморфизмы.

$$A_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad (1)$$

$$A_5 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}, A_6 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \quad (2)$$

Матрицы A_2, A_3, A_5, A_6 связаны замечательными соотношениями: $A_2 = A$, $A_3 = A^2$, $A_5 = B \cdot A^3 \cdot B$, $A_6 = B \cdot A^4 \cdot B$, где матрица $B = B^{-1} = B^T$:

$$B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad (3)$$

$$g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 5 & 0 & 4 & 7 & 6 \end{pmatrix} \quad (4)$$

- Заметим, что здесь приводится лишь один пример функции g, однако нашлось достаточно много подходящих функций.

Построенная *APN-функция F* имеет следующий вектор значений: F=(0 29 54 46 38 63 23 11 0 50 57 14 20 34 42 25 40 7 39 13 26 49 18 60 0 48 36 17 63 11 28 45 32 13 50 26 57 16 44 0 56 58 37 34 19 21 9 10 48 47 27 1 61 38 17 15)

Конструкция 2. В этой конструкции аргументы $x_1, \ldots, x_6$ в бент функции F_1 идут в том же порядке, что и в F, а в функции F_2 аргументы перегруппированы по принципу четности:

$$F(x_1 \ldots x_6) = (F_1(x_1x_2x_3, x_4x_5x_6) \oplus g(x_4x_5x_6), F_2(x_2x_4x_6, x_1x_3x_5) \oplus g(x_1x_3x_5) \oplus g(x_4x_5x_6))$$

$$F_1(x,y) = (f_1(x,y), f_2(x,y), f_3(x,y))$$
$$F_2(x,y) = (f_4(x,y), f_5(x,y), f_6(x,y))$$
$$f_k(x,y) = \langle x, A_k y \rangle,$$

где $g(x) : Z_2^{n/2} \longrightarrow Z_2^{n/2}$ - взаимно-однозначная перестановка , $A_1 = A_4 = E$ - тождественные преобразования, $A_2, A_3, A_2 + A_3, A_5, A_6, A_5 + A_6$ - ортоморфизмы.

$$A_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad (5)$$

$$A_5 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, A_6 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \tag{6}$$

$$g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 1 & 0 & 4 & 2 & 6 & 3 & 5 \end{pmatrix} \tag{7}$$

- Заметим, что здесь приводится лишь один пример функции g, однако нашлось достаточно много подходящих функций.

Построенная *APN-функция F* имеет следующий вектор значений: F=(6 34 39 1 43 29 4 48 38 49 19 6 59 62 0 7 6 36 16 48 47 31 55 5 14 31 12 31 23 20 27 26 14 6 25 19 48 42 41 49 62 5 61 4 48 25 61 22 62 48 30 18 4 24 42 52 38 27 18 45 44 3 22 59).

ЛИТЕРАТУРА

[1] Токарева Н. Н. Симметричная криптография. Краткий курс. //Новосибирский государственный университет. – 2012.

[2] Dai Z. D., Golomb S. W., Gong G. Generating all linear orthomorphisms without repetition //Discrete Mathematics. – 1999. – Т. 205. – №. 1-3. – С. 47-55.

Куратор исследования – к.ф.-м.н., н.с. ИМ СО РАН, ст.преп. НГУ, К.В. Калгин.

О корреляционно-иммунных функциях с максимальной алгебраической иммунностью (продолжение)

И. С. Хильчук
Новосибирский государственный университет
E-mail: i.khilchuk@g.nsu.ru

Аннотация

В данной работе представлено продолжение исследования способа построения булевых функций от большего числа переменных на основе функций от меньшего числа переменных с сохранением показателей алгебраической и корреляционной иммунности исходных функций. Показано, что при использовании способа построения с использованием геометрического представления булевых функций возможно сохранить показатели алгебраической и корреляционной иммунности, но нельзя добиться повышения показателя алгебраической иммунности до максимально возможного.

Ключевые слова: *булевы функции, алгебраическая иммунность, корреляционная иммунность, геометрическое представление.*

Булевы функции являются основными компонентами блочных и поточных симметричных шифров, и стойкость всей шифрсистемы во многом зависит от криптографических свойств используемых функций. Так как количество различных функций от n переменных равно 2^{2^n}, полный перебор множества булевых функций для поиска функций, обладающих нужными свойствами, затруднён, а значит, интерес представляют различные способы построения булевых функций с заданными свойствами.

Булева функция f — это произвольное отображение из $\mathbb{Z}_2^n$ в $\mathbb{Z}_2$.

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

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

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

Алгебраической иммунностью булевой функции f от n переменных называется минимальное число d такое, что существует не тождественно равная нулю булева функция g от n переменных степени d, для которой выполняется $f \cdot g = 0$ или $(f + 1)g = 0$[1]. Высокая алгебраическая иммунность обеспечивает стойкость шифра к алгебраическому криптоанализу. Известна из [2] оценка сверху алгебраической иммунности булевой функции от n переменных: $AI(f) \leq \lceil \frac{n}{2} \rceil$.

Булева функция f от n переменных является корреляционно-иммунной порядка $n-k, 1 \leq k \leq n$, если любой грани $\Gamma_{i_1,\ldots,i_{n-k}}^{a_1,\ldots,a_{n-k}}$ булева куба $\mathbb{E}^n$ размерности k принадлежит одинаковое число точек носителя функции f, а именно $wt(f) \cdot 2^{-(n-k)}$ точек [3].

В [4] был исследован способ построения функций от 6 переменных на основе функций от 4 переменных и показано, что с его помощью можно сохранить показатели алгебраической и корреляционной иммунности исходных функций. Для этого все векторы булева куба $\mathbb{E}^4$ были заменены на грани размерности 2, и векторы, принадлежащие носителю функции от 4 переменных, заменялись на грани, содержащие одинаковое число элементов носителя функции от 6 переменных. В случае, если векторы, принадлежащие носителю функции от 4 переменных, заменяются на грани размерности 2, которым принадлежит различное число элементов носителя, необходимо соблюдать следующие условия:

- вес полученной функции от 6 переменных должен быть чётным;
- для веса такой функции должно выполняться следующее неравенство

$$wt(f) \geq \sum_{i=0}^{2} \binom{6}{i}$$

С помощью программы были построены все возможные функции от 6 переменных на основе функций от 4 переменных с максимальной алгебраической иммунностью и максимальным показателем корреляционной иммунности, равным единице, и было получено следующее утверждение.

Теорема 1. *С помощью комбинаций различных способов расположения элементов носителя в грани размерности* 2 *(с учётом указанных выше ограничений) возможно построить функцию от шести переменных с сохранением показателей алгебраической и корреляционной иммунности исходной булевой функции от четырёх переменных. Однако, ни в одном случае нельзя добиться роста алгебраической иммунности до максимально возможного показателя* $AI(f) = 3$ *для функций от шести переменных.*

ЛИТЕРАТУРА

[1] Carlet C., “Boolean Functions for Cryptography and Coding Theory”, Cambridge University Press, 2021.

[2] Courtois N., Meier W. Algebraic Attacks on Stream Ciphers with Linear Feedback // Proceedings of Eurocrypt 2003, Lecture Notes in Computer Sciences. 2003. V. 2656. P. 345 – 359.

[3] Siegenthaler T., “Correlation-immunity of nonlinear combining functions for cryptographic applications”, IEEE Trans. Inform. Theory, 30:5 (1984), 776–780

[4] Хильчук И.С., Зюбина Д.А., Токарева Н.Н. О корреляционно-иммунных функциях с максимальной алгебраической иммунностью // Труды конференции SIBECRYPT’22. Сдано в печать.

КРИПТОАНАЛИЗ СИММЕТРИЧНЫХ ШИФРОВ

Разностный криптоанализ ARX-шифра

С. А. Бобрышев[1], Н. С. Белянин[1]
[1]ИКТИБ ИТА ЮФУ
E-mail: bobryshev@sfedu.ru, belyanin@sfedu.ru

Аннотация

В рамках данной работы рассмотрен разностный криптоанализ усеченного до 4 раундов алгоритма поточного шифрования Salsa20, где достигается близкая к полученной теоретически вероятность появления необходимых разностей при правильном подборе последних 160 битов ключа, который суммарно занимает 256 битов, и суммарная сложность подбора ключа снижается от 2^{256} до $2^{160}+2^{96}$. В оригинальной версии алгоритма, используется 20 раундов, что заключено в названии. Salsa20 относится к классу ARX-шифров, то есть содержит в себе комбинацию следующих операций:

- побитовый XOR двоичных векторов длины n, обозначается как «$\oplus$»;
- сложение двух чисел по модулю 2^n, в рамках работы обозначается как «$+$»;
- циклический сдвиг битов на, обозначается как «$\lll$» или «$\ggg$».

Помимо этого будут описаны ошибки, найденные в работе P. Crowley 2005-го года.

Ключевые слова: *ARX-шифр, Salsa20, разностный криптоанализ.*

Введение

Данная работа подразумевает достижение следующего ряда целей.

- Провести атаку на усеченный до 4 раундов поточный шифр Salsa20.
- Убедиться в правильности выводов, сделанных P. Crowley в работе 2005 года [5].

Для достижения поставленных целей, необходимо выполнить следующие задачи.

- Создать собственную программную реализацию алгоритма шифрования Salsa20, на основе спецификации, представленной в [13].
- Реализовать эвристический алгоритм поиска максимально вероятной разности, проходящей через нелинейную операцию «$+$», при фиксированных входных разностях, частично описанный в [6].
- Написать программную реализацию генератора необходимых пар входных данных, определенных правилами, составленными для разностного криптоанализа усеченного Salsa20.
- Проанализировать результаты анализа при правильно и неправильно подобранных 160 битах ключа.

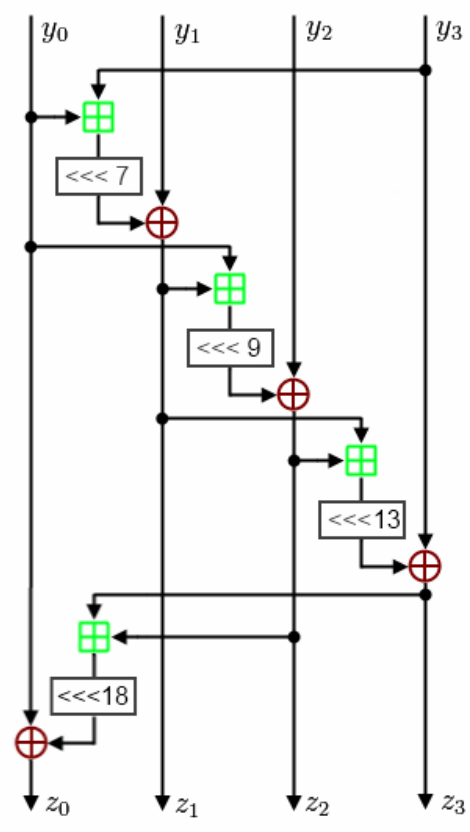

Рис. 1: Схема преобразования *quarterround* (Источник: en.wikipedia.org)

Структура генератора гаммы Salsa20

Входные данные

Входные данные будем обозначать через матрицу y, которая имеет следующий вид:

$$y = \begin{pmatrix} y_0 & y_1 & y_2 & y_3 \\ y_4 & y_5 & y_6 & y_7 \\ y_8 & y_9 & y_{10} & y_{11} \\ y_{12} & y_{13} & y_{14} & y_{15} \end{pmatrix}. \tag{8}$$

Функциональное назначение элементов матрицы показаны на рис. 2 и 3.

Так, на главной диагонали находятся константы, предопределенные создателями шифра, y_2 - y_5 и y_{12} - y_{15} занимают биты ключа, в y_7 - y_8 находятся сессионные параметры, y_9 - y_{10} занимает номер блока гаммы, что позволяет получить произвольный достук к i-му блоку гаммы и повышает удобство шифра.

Преобразование quarterround

В основе Salsa20 лежит функция $quarterround(y_0, y_1, y_2, y_3)$, которая показана на рис. 1.

При помощи данной функции строятся преобразования *rowround* и *columnround*.

Преобразования rowround, columnround

Преобразование $rowround(y)$ последовательно применяет $quarterround(y_0, y_1, y_2, y_3)$ к строкам матрицы y, каждый раз начиная нумерацию с диагонального элемента, и может быть описано

следующим образом:

$$\begin{aligned}(z_0, z_1, z_2, z_3) &= quarterround(y_0, y_1, y_2, y_3),\\ (z_5, z_6, z_7, z_4) &= quarterround(y_5, y_6, y_7, y_4),\\ (z_{10}, z_{11}, z_8, z_9) &= quarterround(y_{10}, y_{11}, y_8, y_9),\\ (z_{15}, z_{12}, z_{13}, z_{14}) &= quarterround(y_{15}, y_{12}, y_{13}, y_{14}),\end{aligned} \tag{9}$$

а преобразование $columnround$ последовательно применяет $quarterround(y_0, y_1, y_2, y_3)$ к столбцам матрицы y, каждый раз начиная нумерацию с диагонального элемента:

$$\begin{aligned}(z_0, z_4, z_8, z_{12}) &= quarterround(y_0, y_4, y_8, y_{12}),\\ (z_5, z_9, z_{13}, z_1) &= quarterround(y_5, y_9, y_{13}, y_1),\\ (z_{10}, z_{14}, z_2, z_6) &= quarterround(y_{10}, y_{14}, y_2, y_6),\\ (z_{15}, z_3, z_7, z_{11}) &= quarterround(y_{15}, y_{12}, y_{13}, y_{14}).\end{aligned} \tag{10}$$

Их композиция используется для реализации преобразования $doubleround$, которое может быть описано как:

$$doubleround(y) = rowround(columnround(y)) \tag{11}$$

Использование ключа

Salsa20 предполагает возможность использования двух видов ключа: размерностью 128 и 256 битов. В рамках данной работы будет рассматриваться атака на ключ длиной 256 битов. При заполнении матрицы y независимо от размера ключа под него выделяется 256 битов, и, в зависимости от размера ключа, выбирается определенный набор констант, заполняющих главную диагональ. На рисунках 2 и 3 представлены отображения матрицы y для ключей длины 256 и 128 битов соответственно.

"expa"	Key_1	Key_2	Key_3
Key_4	"nd 3"	Nonce	Nonce
Pos.	Pos.	"2-by"	Key_5
Key_6	Key_7	Key_8	"te k"

Рис. 2: Назначение элементов для ключа длины 256 битов

"expa"	Key_1	Key_2	Key_3
Key_4	"nd 1"	Nonce	Nonce
Pos.	Pos.	"6-by"	Key_1
Key_2	Key_3	Key_4	"te k"

Рис. 3: Назначение элементов для ключа длины 128 битов

Генерация гаммы

Итоговый вид гаммы, генерируемой Salsa20 для полного алгоритма меет вид:

$$Salsa20(y) = x + doubleround^{10}(y), \tag{12}$$

а для усеченного до 4 раундов:

$$Salsa20/4(y) = x + doubleround^{2}(y). \tag{13}$$

Анализ статьи [5]

Для проведения разностного криптоанализа алгоритма Salsa20 мы взяли за основу статью [5], где описывалась подобная атака на версию Salsa20, усеченную до 5-ти раундов.

Изучив приведенную работу, мы обнаружили, что при разворачивании Salsa20 для 5 раундов вначале применяется функция $rowround^{-1}$, затем транспонируется полученная матрица, хотя должна быть использована операция $columnround^{-1}$. Учитывая вышесказанное, для устранения ошибки, но при этом сохранив вид атаки, мы предлагаем атаку не на 5, а на 4 раунда шифрования.

Основная идея, предложенная в приведенной работе в рамках разностного криптоанализа, заключается в том, чтобы рассчитать максимальные вероятности разностей в определенных позициях после преобразования $doubleround$, и таким образом создать условие, при котором можно убедиться в верности 160 последних битов ключа, не зная его полностью.

Расчет разностей для преобразования doubleround

В структуре ARX-шифров присутствует лишь одна операция, через которую разность проходит неоднозначно, а именно сложение по модулю 2^{32}. Это обосновывается тем, что применение данной операции к одинаковым входным разностям может приводить к различным разностям на выходе. Таким образом, прохождение разности через эту операцию, является вероятностным.

Для нахождения вероятности получения разностей, через сложение по модулю 2^{32}, было принято решение воспользоваться алгоритмом из [6], что позволило находить вероятность за O(log n) по времени. С целью нахождения максимально вероятной выходной разности, была использована эвристика, которая основывалась на том, что вероятность появления разности тем выше, чем меньше битов в ней отличаются от битов операндов сложения по модулю 2^{32}.

Ниже на рис. 4 и 5 представлены расчеты вероятностей после операции $columnround(y)$ и $rowround(y)$ соответственно. Под вопросами у нас скрываются те биты, которые не используются для нахождения максимальной вероятности, но при их учете понижают ее.

Генерация входных пар и сбор статистики с правильным и неправильным подбором последних 160 битов ключа

Для сбора статистики были зашифрованы 10000 пар случайных текстов, гамма для которых генерировалась так, чтобы в матрице y в ячейке y_9 было различие только в старшем разряде, то есть дифференциал в этом элементе был равен 0x80000000, а остальные элементы матрицы получались одинаковыми, то есть их разности равнялись 0.

После генерации пар шифртекстов они складывались по модулю 2 с исходными сообщениями и приводились к парам матриц вида y. Поскольку 3 последние строки матрицы на входе состоят из контролируемых нами в рамках анализа параметров и подбираемых битов ключа, можем считать неизвестной только первую строку входной матрицы. Далее каждый элемент получившихся матриц складывался с аддитивно обратным элементом по модулю 2^{32} для входа. Таким образом из пар «шифртекст-открытый текст» выводилась гамма, сгенерированная алгоритмом Salsa, усеченным до 4 раундов, для каждого из открытых текстов. На основе пары гамм вычислялся их дифференциал — $\Delta\gamma$.

В означенных условиях нам известны три последние строки матрицы каждого дифференциала, поэтому мы можем узнать элементы y_{12} и y_{15} матрицы $doubleround^{-1}(\gamma)$. В получившейся после

```
Start matrix

0x0000_0000     0x0000_0000     0x0000_0000     0x0000_0000
0x0000_0000     0x0000_0000     0x0000_0000     0x0000_0000
0x0000_0000     0x8000_0000     0x0000_0000     0x0000_0000
0x0000_0000     0x0000_0000     0x0000_0000     0x0000_0000

ColumnRound

MatrixColumn = 1

y0 = 0x0000_0000
y1 = 0x8000_0000
y2 = 0x0000_0000
y3 = 0x0000_0000

(y0 +  y3)mod(2**32) = 0x0000_0000 + 0x0000_0000 = 0x0000_0000 (probability=1)
z1 = y1 xor ((y0 + y3) << 7) = 0x8000_0000 (probability=1)

(z1 +  y0)mod(2**32) = 0x8000_0000 + 0x0000_0000 = 0x8000_0000 (probability=1)
z2 = y2 xor ((z1 + y0) << 9) = 0x0000_0100 (probability=1)

(z2 +  z1)mod(2**32) = 0x0000_0100 + 0x8000_0000 = 0x8000_0100 (probability=0.5)
z3 = y3 xor ((z2 + z1) << 13) = 0x0020_1000 (probability=0.5)

(z3 +  z2)mod(2**32) = 0x0020_1000 + 0x0000_0100 = 0x0020_1100 (probability=0.125)
z0 = y0 xor ((z3 + z2) << 18) = 0x4400_0080 (probability=0.125)

After ColumnRound matrix (Probability=0.5))

0x0000_0000     0x0020_1000     0x0000_0000     0x0000_0000
0x0000_0000     ??????????      0x0000_0000     0x0000_0000
0x0000_0000     0x8000_0000     0x0000_0000     0x0000_0000
0x0000_0000     0x0000_0100     0x0000_0000     0x0000_0000
```

Рис. 4: Расчет вероятностей после операции $columnround(y)$

этого преобразования матрице велся подсчет частоты встречаемости элементов y_{12} и y_{15}, для которых ранее была рассчитана наибольшая вероятность появления, равная 2^{-3}.

На рис. 6 и 7 продемонстрированы результаты нашей программы в случае если ключ подобран правильно и неправильно соответственно.

```
Line 3
MatrixColumn = 1

y0 = 0x0000_0000
y1 = 0x0000_0000
y2 = 0x0000_0100
y3 = 0x0000_0000

(y0 +  y3)mod(2**32) = 0x0000_0000 + 0x0000_0000 = 0x0000_0000 (probability=1)
z1 = y1 xor ((y0 + y3) << 7) = 0x0000_0000 (probability=1)

(z1 +  y0)mod(2**32) = 0x0000_0000 + 0x0000_0000 = 0x0000_0000 (probability=1)
z2 = y2 xor ((z1 + y0) << 9) = 0x0000_0100 (probability=1)

(z2 +  z1)mod(2**32) = 0x0000_0100 + 0x0000_0000 = 0x0000_0100 (probability=0.5)
z3 = y3 xor ((z2 + z1) << 13) = 0x0020_0000 (probability=0.5)

(z3 +  z2)mod(2**32) = 0x0020_0000 + 0x0000_0100 = 0x0020_0100 (probability=0.25)
z0 = y0 xor ((z3 + z2) << 18) = 0x0400_0080 (probability=0.25)

After RowRound matrix (Probability=0.125))

??????????     ??????????     ??????????     ??????????
??????????     ??????????     ??????????     ??????????
??????????     ??????????     ??????????     ??????????
0x0000_0000    ??????????     ??????????     0x0400_0080
```

Рис. 5: Расчет вероятностей после операции $rowround(y)$

```
differences_3_0
[('0x0000_0000', 1.0)]

differences_3_3
[('0x0400_0080', 0.1623),
 ('0x0400_0180', 0.1002),
 ('0x0c00_0080', 0.0811),
 ('0x0400_0380', 0.0597),
 ('0x0c00_0180', 0.0518)]
```

Рис. 6: Верно подобранные последние 160 битов ключа

Выводы

Детально изучив статью по разностному криптоанализу усеченного алгоритма Salsa20 для ее 5-ти раундовой версии мы заметили значительные ошибки и на их основе смогли повторить предло-

```
differences_3_0
[('0xa840_99ae', 0.0001),
 ('0xa356_3df9', 0.0001),
 ('0xe4cb_e497', 0.0001),
 ('0xa653_7805', 0.0001),
 ('0x6eb4_0411', 0.0001)]

differences_3_3
[('0xb5a1_985f', 0.0001),
 ('0x4ba2_ffc2', 0.0001),
 ('0x27fc_d5a4', 0.0001),
 ('0xf73d_5112', 0.0001),
 ('0xaa91_e50f', 0.0001)]
```

Рис. 7: Неправильно подобранные последние 160 битов ключа

женнный в [5] метод перебора не всех 256 битов ключа, а лишь последних 160 битов, получив при этом оценку сложность порядка $2^{160} + 2^{96}$.

ЛИТЕРАТУРА

[1] P. Crowley, *Truncated differential cryptanalysis of five rounds of Salsa20*, 2005, https://eprint.iacr.org/2005/375

[2] D. Bernstein, *Salsa20 specification*, 2005, https://eprint.iacr.org/2005/375

[3] H. Lipmaa, S. Moriai, *Efficient Algorithms for Computing Differential Properties of Addition*. In: Matsui, M. (eds) Fast Software Encryption. FSE 2001. Lecture Notes in Computer Science, vol 2355. Springer, Berlin, Heidelberg, 2002

Кураторы исследования:

Куратор 1 – к.ф.-м.н., н.с. ИМ СО РАН, Н.А. Коломеец.;

Куратор 2. - магистрант ФИТ НГУ, А.С. Мокроусов

Новый подход к классификации XS-схем малой размерности

И. М. Одуд[1], А. О. Бахарев[1], Д. Р. Парфенов[1], А. В. Куценко[2]
[1]Новосибирский государственный университет
[2]Институт математики им. С. Л. Соболева СО РАН
E-mail: i.odud@g.nsu.ru, a.bakharev@g.nsu.ru, d.parfenov@g.nsu.ru, alexandrkutsenko@bk.ru

Аннотация

Гарантированное число активаций является важной характеристикой, позволяющей получить оценку стойкости шифра к разностному криптоанализу. В данной работе предложена оптимизация существующего алгоритма поиска гарантированного числа активаций, предполагающая замену вычисления ранга соответствующей матрицы на альтернативную проверку. Была выполнена проверка выдвинутых ранее гипотез о возможности классификации XS-схем на основе веса разности задающих их векторов [1]. С помощью анализа данных, полученных с использованием оптимизированной версии алгоритма, были обобщены раннее замеченные закономерности между значением гарантированного числа активаций и структурой XS-схем.

Ключевые слова: *гарантированное число активаций, XS-схема, разностный криптоанализ, оптимизация алгоритма вычисления гарантированного числа активаций.*

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

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

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

XS-схему можно представить в виде расширенной матрицы

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

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

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

столбцы которой имеют следующий вид:

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

где b — последний столбец матрицы B.

Для каждого шифра можно найти число активаций.

Определение 1. *Число активаций шифра* — количество ненулевых разностей, попавших на вход S-блоков.

Определение 2. *Гарантированное число активаций шифра (guaranteed activation number, GNA)* — наименьшее из всех минимальных чисел активаций.

Отметим, что поиск связан с исследованием линейного кода определенного вида, который строится на основе рассматриваемого шифра.

Одним из подходов к поиску гарантированного числа активаций является алгоритм, представленный в работе [3]. Основная идея алгоритма — полный перебор разбиения матрицы G на G_0 и G_1, так чтобы:

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

Для краткости формулировок введём следующее определение

Определение 3. Будем называть разбиение пар столбцов матрицы G на матрицы G_0 и G_1 **«неподходящим»** если в G_1 содержится столбец, линейно зависящий от столбцов матрицы G_0.

Раннее была предложена оптимизация к данному алгоритму вычисления гарантированного числа активаций на основе интерпертации полного перебора как обхода двочного дерева [1]. Однако, всё ещё остаётся возможность для улучшения. Например, существует способ проверить узел дерева на соответствие «неподходящему» разбиению, используя структуру дерева перебора.

Предложение 1. *Пусть p — путь в дереве перебора от корня до узла, соответствующему «неподходящему» разбиению. Тогда любой путь, суффиксом которого является p, также соответствует «неподходящему» разбиению.*

Данная оптимизация позволяет дополнительно ускорить вычисление гарантированного числа активаций схемы. Её эффект намного заметнее на схемах небольших размерностей ($n = 2, 3, 4$) в силу того, что для таких размерностей множество линейно зависимых векторов имеет небольшую мощность, что позволяет заменить практически все вычисления рангов на проверку суффикса.

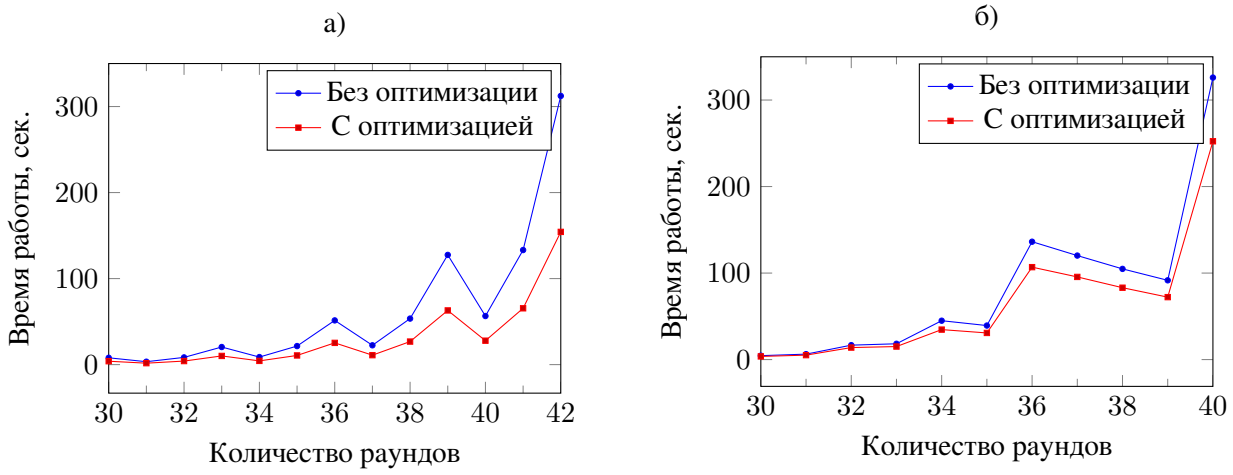

Рис. 1: Время вычисления GNA для а) beltWBL-2 ($n = 2$) б) beltWBL-8 ($n = 8$)

Как видно из графиков, для размерности 8 ускорение росло с ростом количества раундов и составляло от 5% до 30%, в то время как для размерности 2 практически всё время наблюдается двухкратное ускорение. При этом, для размерности 2 ранг вычисляется лишь один раз, а все остальные отсечения совершаются по суффиксу.

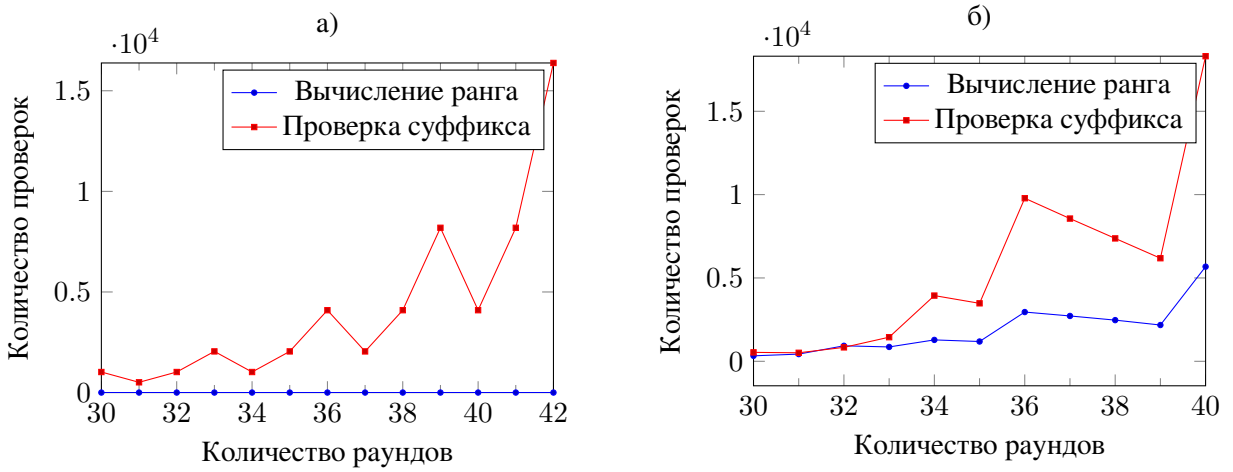

Рис. 2: Количество проверок на линейную зависимость для а) beltWBL-2 ($n = 2$) б) beltWBL-8 ($n = 8$)

Помимо оптимизации алгоритма, в данной работе было проверено несколько предположений, представленных ранее [1].

Весом Хэмминга $wt(x)$ вектора $x \in \mathbb{F}^n$ называется количество ненулевых кординат в данном векторе:

$$wt(x) = |\{i \in \{1, 2, \ldots, n\} | x_i \neq 0\}|.$$

Гипотеза 1. XS-схемы размерности n, для которых выполняется $wt(a-b)=n$, имеют одинаковое число активаций.

Гипотеза 2. XS-схемы размерности n, для которых выполняется $wt(a-b)=\mathrm{const}$, имеют одинаковое число активаций.

Гипотеза 3. Пусть XS-схема задаётся парой векторов (a,b). Тогда её гарантированное число активаций совпадает с гарантированным числом активаций XS-схемы, заданной парой (b,a).

Введём следующее определение:

Определение 4. *Скорость роста гарантированного числа активаций* ξ – действительная неотрицательная величина, определяемая для t раундов по формуле:

$$\xi=\frac{1}{t-1}\sum_{i=0}^{t-1}\frac{GNA_{i+1}-GNA_i}{i+1-i}=\frac{GNA_t-GNA_0}{t-1}$$

Рассмотрим две XS-схемы: (a,b) и (a',b'). Каждой из них соответствует свой набор $(GNA_i,i)_t$, где GNA_i — гарантированное число активаций на i-ом раунде. Далее, набор $(GNA_i,i)_t$ будет называться *кинетикой схемы* из t раундов. Пусть ξ — скорость роста GNA для схемы (a,b), а ξ' — для схемы (a',b'). Тогда, если $\xi>\xi'$, то за одинаковое число раундов GNA схемы (a,b) будет больше, чем для (a',b').

Для всевозможных векторов a и b размерности $n=2,3,4,5,6$ были вычислены значения скоростей ξ. На рисунке 3 приведены значения ξ для нечётной ($n=5$) и чётной ($n=6$) размерности, распределённые на классы по весу разности векторов a и b ($wt(a-b)$).

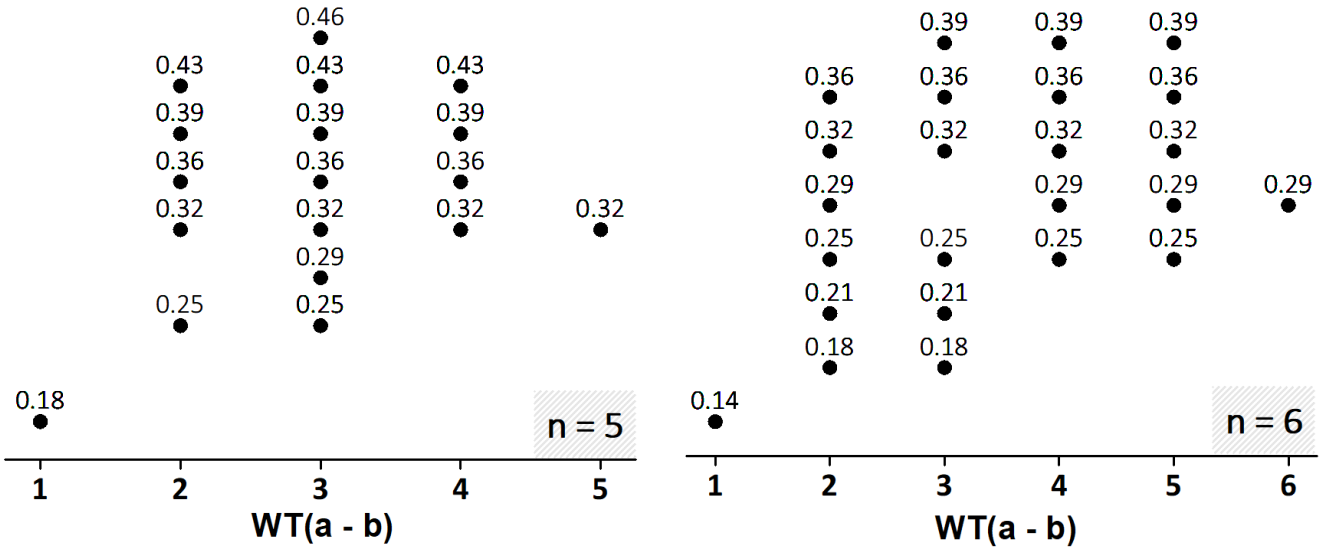

Рис. 3: Значения ξ для нечётной ($n=5$) и чётной ($n=6$) размерности в зависимости от веса разности векторов a и b ($wt(a-b)$)

Определение 5. $Var(wt(a-b))$ – число различных значений ξ в классах по весу разности $wt(a-b)$.

Как видно из рисунка 3, величина $Var(wt(a-b))$ для разных значений $wt(a-b)$ в рамках одной размерности n меняется не случайным образом. А именно, для классов с $wt(a-b)=1$ и n всегда $Var(wt(a-b))=1$, а для $wt(a-b)=\lfloor\frac{n}{2}\rfloor+1$ достигается максимальное значение $wt(a-b)$. Особенно хорошо это видно, когда размерность n нечётная ($n=5$ рисунке 3).

При анализе вычисленных значений была выдвинута следующая гипотеза:

Гипотеза 4. Число различных элементов $Var(wt(a-b))$ в классе $wt(a-b)$ определяется числом сочетаний из $n-1$ по $wt(a-b)-1$ в рамках одной размерности XS-схемы n:

$$C_{n-1}^{wt(a-b)-1} = \frac{(n-1)!}{(wt(a-b)-1)!(n-wt(a-b))!} \tag{14}$$

Данная гипотеза была проверена для размерностей $n = 2, 3, 4, 5, 6$ (рисунок 4). Как видно из рисунка 4, для размерностей меньше 6, с точностью до одного различного значения в классе, экспериментальные данные согласуются с предложенными соотношениями. Однако для классов с $n = 6$ наблюдается значительное отклонение от теоретической модели.

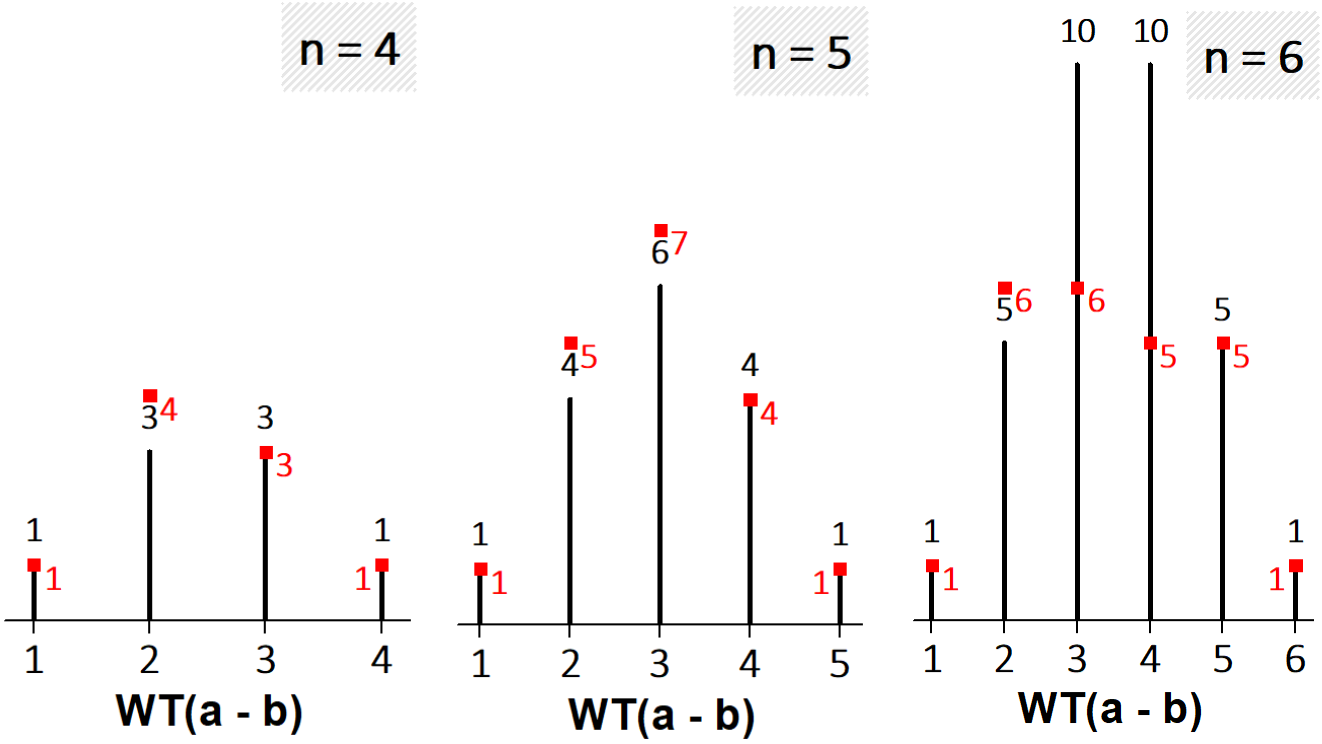

Рис. 4: Сравнение числа различных классов, полученных непосредственно при анализе ξ (красные точки) и рассчитанных согласно формуле 14 (чёрные линии).

Определение 6. $Sum(GNA, t)$ – сумма гарантированных чисел активации с 0-го раунда и до t-го раунда:

$$Sum(GNA, t) = \sum_{i=0}^{t} GNA_i$$

Далее мы пронаблюдали за случайностью величин GNA. А именно, каждой кинетике (GNA, t) сопоставили величину $Sum(GNA, t)$ и посчитали количества одинаковых значений (рисунок 3). По графику на рисунке 5 видно, что распределение $Sum(GNA, t)$ не изотропно. Однако данных не достаточно, чтобы сделать конкретное предположение о виде распределения кинетик GNA для различных XS-схем.

В результате гипотезы 2 и 3 были опровергнуты, и попытка разбить XS-схемы одной размерности на классы, исходя из того что гарантированное число активаций схем с одинаковым $wt(a-b)$ растёт с одинаковой скоростью, оказалась неудачной. В то же время, гипотеза 1 опровергнута не была и получила ещё одно численное подтверждение.

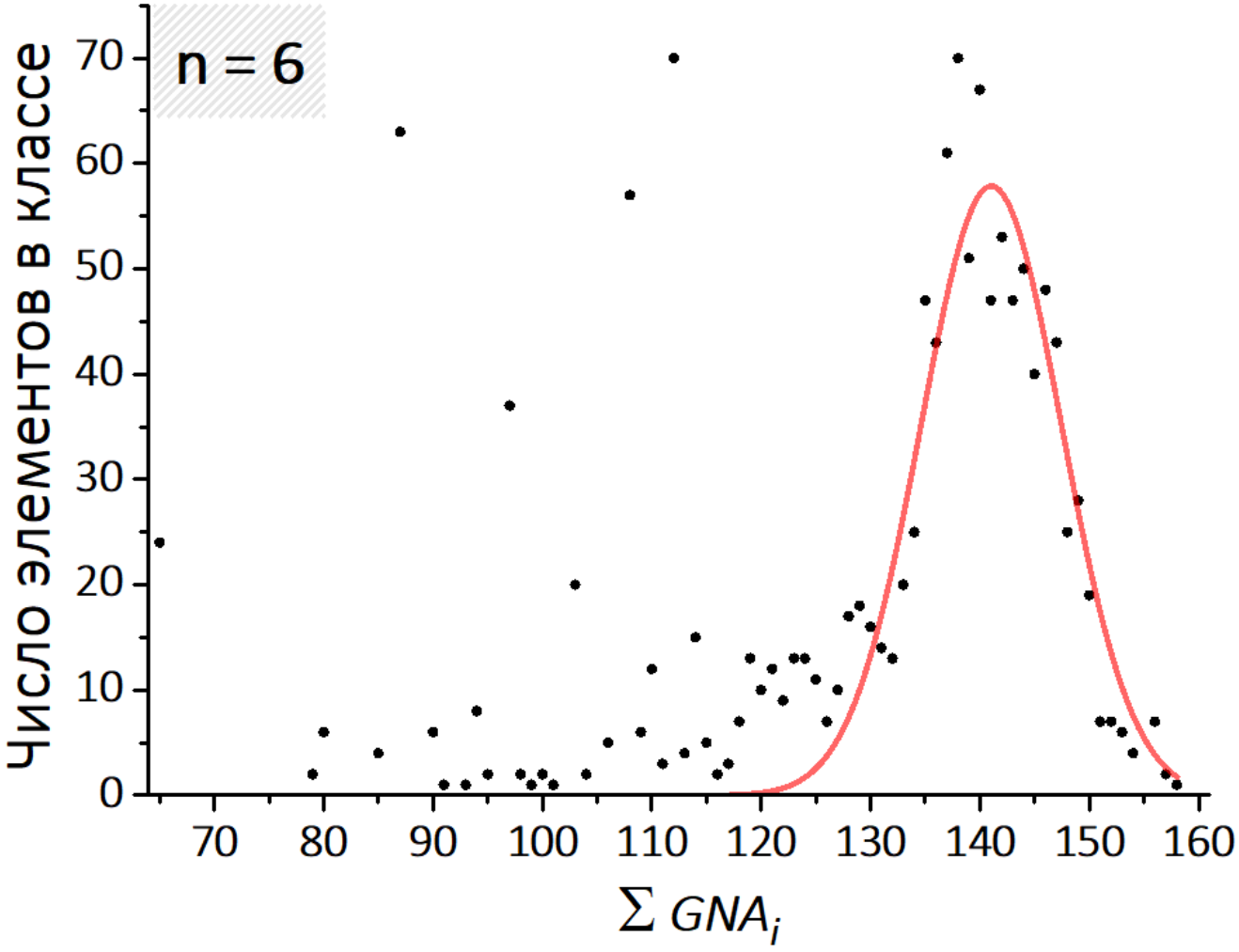

Рис. 5: Распределение кинетик $Sum(GNA, t)$ для размерности $n = 6$ (чёрные точки), красная линия — интерполяция гауссового распределения с фиксированным центром.

ЛИТЕРАТУРА

[1] D. Parfenov, A. Bakharev, A. Kutsenko, A. Belov, N. Atutova – Effective algorithm to compute guaranteed number of activations in XS-circuits and it's application to the cipher design // In: Preproceedings of the 11th Workshop on Current Trends in Cryptology (CTCrypt-2022, June 6–9, 2022), 2022, pp. 66–86.

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

[3] S. Agievich – On the guaranteed number of activations in XS-circuits // Mat. Vopr. Kriptogr., 12:2 (2021), pp. 7–20.

Кураторы исследования –
к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, н.с. ИМ СО РАН, Куценко Александр Владимирович;
магистрант ФИТ НГУ Парфенов Денис Романович;

ПОСТКВАНТОВАЯ КРИПТОГРАФИЯ

Постквантовые криптосистемы на обобщенных кодах Рида-Соломона: обзор известных атак и модификаций

А. А. Кунинец[3], Д. К. Воробьёв[3], В. А. Левин[1], А. О. Бахарев[1,2], Т. А. Бонич[1,2], Е. С. Малыгина[3], А. В. Куценко[1,2]

[1]Новосибирский государственный университет

[2]Институт математики им. С. Л. Соболева СО РАН

[3]Балтийский федеральный университет имени Иммануила Канта

E-mail: artkuninets@yandex.ru, DKiVorobev@stud.kantiana.ru, v.levin@g.nsu.ru, a.bakharev@g.nsu.ru, t.bonich@g.nsu.ru, alexandrkutsenko@bk.ru

Аннотация

В данной работе представлены краткое введение в теорию полей, теорию кодирования, основные классы кодов и их характеристики, а также рассмотрено применение обобщённых кодов Рида-Соломона в криптографии на примере криптосистем Мак-Элиса, Нидеррайтера, BBCRS. Также описана атака Сидельникова-Шестакова на криптосистему Нидеррайтера. Далее рассмотрены модификация обобщённых кодов Рида-Соломона с использованием подкодов, предложенная Thierry P. Berger и Pierre Loidreau. Данная модификация является устойчивой к атаке Сидельникова-Шестакова. Изучена модифицированная атака на криптосистему, основанную на использовании подкодов обобщённого кода Рида-Соломона.

Ключевые слова: *Постквантовая криптография, коды, исправляющие ошибки, криптосистемы на кодах, обобщенные коды Рида-Соломона, атака Сидельникова-Шестакова*

Введение

В настоящее время повсеместно используется асимметричная криптография для зашифрования сообщений, обмена ключами или создания электронной подписи. Асимметричная криптография является типом шифрования, в котором ключи, используемые для зашифрования и расшифрования сообщения, различаются: для зашифрования используется *открытый ключ*, а для расшифрования — *закрытый ключ*. При этом, по открытому ключу сложно вычислить закрытый. Этот формат отличен от принципов симметричной криптографии, в которой один и тот же ключ применяется для обеих операций процесса шифрования.

Большинство алгоритмов асимметричной криптографии основаны на сложности вычисления дискретного логарифма, а также сложности факторизации, то есть разложении больших чисел на простые множители. Ярким примером служит широко используемая криптосистема RSA [12], в которой частью закрытого клюа является пара простых чисел n, q, а в открытый ключ входит их произведение $n = pq$. Все известные на данный момент алгоритмы решения задач дискретного логарифмирования и факторизации имеют экспоненциальную сложность. Ввиду развития компьютерных технологий, которые влекут за собой увеличение количества операций, выполняемых компьютером за определённое время, размер ключей, параметров постоянно увеличивается.

Однако, начиная с 80-х годов прошлого века активно разрабатывается концепция квантовых вычислений. Сегодня квантовый компьютер уже не является абстрактным понятием, так как существуют его рабочие модели. Развитие квантовых технологий и вычислений влечёт за собой опасность как для симметричной, так и для асимметричной криптографии. На данный момент

существуют алгоритм Шора [13], который позволяет раскладывать числа на простые множители за полиномиальное время, а также алгоритм Гровера [7], который позволяет найти решение уравнения $f(x) = 1$, $0 \leqslant x < N$ за время $O(\sqrt{N})$. Следовательно, криптосистема RSA, а также другие системы, основанные на сложности вычисления дискретного логарифма или факторизации, перестанут обеспечивать достаточную защиту, а на определённом этапе развития квантовых вычислений и вовсе станут совершенно бесполезными. Однако существуют классы криптосистем, основанных на сложности решения таких задач, которые, вероятно, не будут иметь эффективного алгоритма решения даже при появлении достаточно мощного квантового компьютера. Эти криптосистемы относят к классу постквантовой криптографии. В данной работе пойдёт речь о криптосистемах, основанных на кодах исправляющих ошибки, которые являются кандидатами для постквантовой криптографии.

В главе 2 будут рассмотрены основы теории полей и теории кодирования, а также важнейшие классы кодов, исправляющих ошибки. Далее, в главе 3, будут представлены криптосистемы Мак-Элиса и Нидеррайтера, а в главе 4 детально разобрана атака Сидельникова-Шестакова на данные криптосистемы. В главе 5 разбирается модификация обобщённых кодов Рида-Соломона с использованием подкодов, устойчивая к атаке Сидельникова-Шестакова, также представлена модифицированная атаку на предложенную криптосистему. В главе 6 представлена схема, основанная на маскировании матриц обобщённых кодов Рида-Соломона, которая также устойчива к данной атаке.

Предварительные сведения

Данная глава посвящена введению основных терминов и определений из теории полей Галуа и теории кодирования, которые будут использованы нами в этой работе. Они необходимы для понимания построения различных классов кодов, а также криптосистем и атак, рассмотренных в главах 3, 4, 5, соответственно.

Сведения из алгебры

Для начала введём базовые определения из теории групп. Затем рассмотрим некоторые характеристики и понятия из данной области.

Определение 1. *Группа* G — это множество с одной бинарной операцией ('*'), для которой выполняются свойства:

1. $(a * b) * c = a * (b * c), \forall a, b, c \in G$
2. $\exists e \in G \,|a * e = e * a = a, \forall a, b \in G$
3. $\forall a \in G \;\; \exists b \in G \mid a * b = b * a = e$

Определение 2. *Кольцо* R — это множество элементов с двумя бинарными операциями ('+','*'), такое что для его элементов выполняются свойства: $\forall a, b, c \in R$

1. $a + b = b + a$
2. $(a + b) + c = a + (b + c)$
3. $\exists - a \in R \mid a + (-a) = 0$

4. $\exists\, 0 \mid a + 0 = a$

5. $a * (b * c) = (a * b) * c$

6. $a * (b + c) = a * b + b * c$

7. $(b + c) * a = b * a + c * a$

Определение 3. Подкольцо I кольца R называется *идеалом* кольца R, если для $\forall r \in R$ и $\forall i \in I$ $i * r; r * i \in I$.

Определение 4. Пусть $I = (i)$ — идеал кольца R. Множество классов вычетов по модулю a образует кольцо $R/I = R/(i)$ с операциями $(r_1 + I) + (r_2 + I) = (r_1 + r_2) + I$ и $(r_1 + I)(r_2 + I) = r_1 r_2 + I$

Определение 5. *Полем* называется коммутативное кольцо с единицей, в котором каждый ненулевой элемент обратим.

Определение 6. *Характеристикой поля* $\mathbb{F}$ называется наименьшее положительное целое число p такое, что для $\forall \alpha \in \mathbb{F}$ справедливо равенство $p \cdot \alpha = 0$

В теории кодирования рассматриваются конечные поля с $q = p^m$ элементами. Все обратимые элементы поля образуют группу $\mathbb{F}_q^* = \mathbb{F}_q / \{0\}$ по умножению, которая называется мультипликативной. Умножение в ней происходит по модулю q. *Порядком* мультипликативной группы поля является число его элементов, т.е. $|\mathbb{F}_q^*| = q - 1$.

Следующие две теоремы подводят к способу построения полей, их расширений и помогают лучше понять структуру данных конструкций.

Теорема 1. *Произвольное поле $\mathbb{F}$ содержит простое подполе K, где $K \cong \mathbb{Q}$ или $K \cong \mathbb{Z}_p$, где p - простое число.*

Легко видеть, что каждое конечное поле $\mathbb{F}$ есть расширение некоторого поля классов вычетов по простому модулю, покажем, как посчитать число элементов конечного поля. Рассмотрим произвольное конечное поле $\mathbb{F}$, в этом поле существует наибольшая система линейно-независимых элементов: $\alpha_1 \cdot a_1 + \ldots + \alpha_m \cdot a_m, \alpha_i \in \mathbb{Z}_p$. Заключаем, что данное поле состоит из всех элементов такого вида, а их существует в точности p^m. Конечное поле определяется своей характеристикой и степенью (числом m) однозначно, с точностью до изоморфизма.

Теорема 2. *Следующие важные факты теории полей являются эквивалентными: мультипликативная группа поля $\mathbb{F}_q$ циклична $\Leftrightarrow$ все элементы поля являются степенями одного элемента, называемого примитивным элементом поля $\Leftrightarrow$ каждый элемент поля $\mathbb{F}_q$ является корнем $q - 1$ степени из единицы.*

Покажем построение конечного поля присоединением корня неприводимого многочлена. Рассмотрим неприводимый над $\mathbb{Z}_p$ многочлен $f(x) = \alpha_m \cdot x^m + \ldots + \alpha_1 \cdot x^1 + \alpha_0$ и пусть некоторый элемент β таков, что $f(\beta) = 0$. Тогда $\mathbb{Z}_p(\beta) = [\alpha_0 \cdot \beta^0, \alpha_1 \cdot \beta^1, \ldots, \beta^m = -\alpha_m^{-1} \cdot (\alpha^{m-1} \cdot \beta^{m-1} + \alpha^1 \cdot \beta^1 + \alpha^0)]$. Таким образом показываем, что степени корня неприводимого многочлена с нулевой по $(m-1)$-ю линейно-независимы над $\mathbb{Z}_p$, а всякая степень элемента β старше $(m-1)$-й линейно выражается через предыдущие. Построенное множество является полем Галуа (конечным полем) $\mathbb{F}_{p^m}$. Многочлен $f(x)$ называется *минимальным многочленом* элемента β. Отметим также, что фактор-кольцо по идеалу многочлена f является полем, изоморфным построенному только что: $\mathbb{Z}_p[x]/(f(x)) \cong \mathbb{F}_{p^m}$. На практике минимальный многочлен подбирают так, чтобы он имел как можно более 'простой вид', потому что, как легко видеть, выбор многочлена задаёт 'правило понижения степени' для практической реализации арифметики полей Галуа.

Определение 7. *Порядком* элемента α мультипликативной группы поля называется наименьшее k такое, что $\alpha^k = 1$. Если $ord(\alpha) = q - 1$, то α называют примитивным элементом поля $\mathbb{F}_q$

Основы теории кодирования

В этой части будут рассмотрены такие понятия как линейный код, его характеристики и некоторые элементы алгебры, непосредственно связанные с кодами.

Определение 8. *Линейный код* — это линейное подпространство векторного пространства V^n, где n - длина кода. В таком случае говорят, что C — код длины n, а $c \in C$ называют кодовым словом.

В данной работе будут рассматриваться линейные блочные коды над алфавитом $A = \mathbb{F}_q$, где q — степень простого числа, $\mathbb{F}_q$ — конечное поле с q элементами. При блочном кодировании, входящие биты разделяются на блоки длины k.

Определение 9. *Размерностью кода* называется его размерность, как векторного пространства. Таким образом, $k = \dim(C)$ равно количеству информационных символов в кодовом слове.

Из определений 8,9 следует, что код C является линейным подпространством $\mathbb{F}_q^n$. Таким образом, кодирование можно определить как применение линейной функции $f_C : \mathbb{F}_q^k \to \mathbb{F}_q^n$, обладающей свойством инъективности, к исходному информационному блоку длины k.

Определение 10. Пусть $a, b \in \mathbb{F}_q^n$, $a = (a_1, a_2, \ldots, a_n)$, $b = (b_1, b_2, \ldots, b_n)$. Тогда $d(a, b) = |\{i : a_i \neq b_i\}|$ называется *расстоянием Хэмминга* между векторами a и b.

Определение 11. *Весом Хэмминга* вектора a называется число, равное $w(a) := |\{i : a_i \neq 0\}|$. Очевидно, что $w(a) = d(a, 0)$.

Определение 12. *Кодовое расстояние* $d = d(C)$ — это число $\min\{d(a, b) \mid a, b \in C\ ;\ a \neq b\}$

Во многих источниках (например [8]) запись $[n, k, d]$ обозначает линейный код C длиной n, размерностью k и кодовым расстоянием d. Далее в работе используется именно это обозначение.

В следующих теоремах вводится одна из важнейших характеристик кода, а именно количество ошибок, которые он может исправить, а также границы для параметров кода.

Теорема 3. *Пусть C — $[n, k, d]$-код. Тогда код C исправляет $t = \lfloor \frac{d-1}{2} \rfloor$ ошибок.*

Теорема 4. *(граница Синглтона). Пусть C — $[n, k, d]$-код. Тогда $k + d \leqslant n + 1$.*

Определение 13. Пусть C — $[n, k]$-код. *Порождающая матрица* G кода C — это матрица размерности $k \times n$, чьи строки являются базисом кода C (как линейного пространства)

$$G = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{k1} & a_{k2} & \ldots & a_{kn} \end{pmatrix}$$

Определение 14. Для $x, y \in \mathbb{F}_q^n$ обозначим $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$.

Определение 15. Пусть $C \subseteq \mathbb{F}_q^n$ — $[n, k]$-код. Тогда $C^\perp = \{u \in \mathbb{F}_q^n | \langle u, c \rangle = 0, \forall c \in C\}$ — *дуальный код* к C.

$C^\perp = [n, n-k]$-код, $(C^\perp)^\perp = C$

Определение 16. Порождающая матрица H кода $C^\perp$ называется *проверочной матрицей* кода C. $\mathrm{rank}(H) = n - k = r$

Проверочная матрица H используется для проверки принадлежности вектора $[n, k]$-коду C. Любой вектор $x \in C$ удовлетворяет $H \cdot x^T = 0$, т.е.

$$\begin{cases} h_{11}x_1 + h_{12}x_2 + \ldots + h_{1n}x_n = 0 \\ h_{21}x_1 + h_{22}x_2 + \ldots + h_{2n}x_n = 0 \\ \ldots \\ h_{r1}x_1 + h_{r2}x_2 + \ldots + h_{rn}x_n = 0 \end{cases}$$

Существует простой способ нахождения порождающей матрицы по проверочной и наоборот.

Теорема 5. *Если проверочная матрица H кода C задана в каноническом виде $H = [A_{n-k,k}|E_{n-k}]$, то порождающая матрица этого кода имеет вид $G = [E_k| - A^T_{n-k,k}]$*

Важные классы кодов

Существуют различные классы кодов, которые предназначены для решения разных задач. Коды, рассмотренные в этой части хорошо изучены, для всех описаны границы параметров, а для некоторых выведены их точные значения. Далее в главе 3 будут использоваться теоремы и определения, связанные с обобщёнными кодами Рида-Соломона.

Циклические коды

Определение 17. Линейный код длины n называется *циклическим*, если для любого кодового слова $(x_1, x_2, \ldots, x_n)$ вектор $(x_2, \ldots, x_n, x_1)$ также является кодовым словом.

Пусть $\mathbb{F}_q$ — конечное поле. $\mathbb{F}_q[x]$ — кольцо многочленов с коэффициентами из $\mathbb{F}_q$. Элементы циклического кода соответствуют элементам идеала $\mathbb{F}_q/(x^n-1)$. Для удобства кодовое слово можно представить в виде многочлена $c(x) \in \mathbb{F}_q[x]$. Тогда циклический сдвиг координат на одну позицию вправо равносилен умножению $c(x)$ на $x \mod x^n - 1$.

Порождающий многочлен $g(x)$ циклического кода C представляет собой унитарный многочлен наименьшей степени из C. Следовательно $g(x)|(x^n-1)$, поэтому размерность кода $k = n - deg(g(x))$

Порождающая матрица циклического $[n, k]$-кода с порождающим многочленом $g(x) = g_0 + g_1x + \ldots + g_{n-k-1}x^{n-k-1}$ имеет вид:

$$G = \begin{pmatrix} g_0 & g_1 & \ldots & g_{n-k-1} & 0 & \ldots & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots & g_{n-k-1} & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & g_0 & g_1 & \ldots & g_{n-k-1} & 0 \\ 0 & \ldots & \ldots & 0 & g_0 & g_1 & \ldots & g_{n-k-1} \end{pmatrix}$$

Определение 18. Многочлен $h(x)$ такой, что $g(x) \cdot h(x) = x^n - 1$ называется *проверочным многочленом* циклического кода. Его степень $k = n - \deg(g(x))$

Следовательно, порождающая матрица циклического $[n, k]$-кода с проверочным многочленом $h(x) = h_0 + h_1 x + \ldots + h_k x^k$ имеет вид:

$$H = \begin{pmatrix} 0 & \ldots & \ldots & 0 & h_k & \ldots & h_1 & h_0 \\ 0 & \ldots & 0 & h_k & \ldots & h_1 & h_0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & h_k & \ldots & h_1 & h_0 & 0 & \ldots & 0 \\ h_k & \ldots & h_1 & h_0 & 0 & \ldots & \ldots & 0 \end{pmatrix}$$

Коды Боуза-Чоудхури-Хоквингема (БЧХ-коды)

Пусть α — примитивный элемент поля $\mathbb{F}_{p^m}$, $g(x)$ — порождающий многочлен циклического кода C длины n такой, что существуют целые числа $b \geq 0$ и $\delta > 1$ такие, что выполняется:

$$g(\alpha^b) = g(\alpha^{b+1}) = \ldots = g(\alpha^{b+\delta-2}) = 0, \tag{1}$$

где δ — *конструктивное расстояние* кода.

Определение 19. *БЧХ-кодом* над $\mathbb{F}_q$ длины $n = p^m - 1$ с конструктивным расстоянием $\delta > 1$ называется циклический код с порождающим многочленом наименьшей степени, для которого выполняется (1).

Теорема 6. *Код БЧХ над $\mathbb{F}_q$ длины $n = p^m - 1$ с порождающим многочленом*

$$g(x) = \textit{НОК}\{M^{(b)}(x), M^{(b+1)}(x), \ldots, M^{(b+\delta-2)}(x)\}$$

для некоторого $b \geq 0$ имеет параметры

$$[n = p^m - 1, k \geq n - (\delta - 1)m, d \geq \delta]$$

Пусть $k \in \mathbb{Z}$. Тогда $M^{(k)}$ — минимальный многочлен элемента α^k, где α - примитивный элемент поля.

$$H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \ldots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \ldots & \alpha^{(n-1)(b+1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \ldots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}$$

Коды Рида-Соломона

Коды Рида-Соломона [11] — это коды БЧХ над полем $\mathbb{F}_q, q = p^m$, длина n которых равна $q - 1$, кодовое расстояние $d = n - k + 1$, а порождающий многочлен имеет вид

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \ldots (x - \alpha^{b+\delta-2})$$

Коды Рида-Соломона можно определить альтернативным способом:

$$\text{RS}_{n,k}(\alpha) = \{f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n) \mid f \in \mathbb{F}_q[x]; \deg(f) < k\}$$

$$G = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \ldots & \ldots & \ldots & \ldots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

Обобщённые коды Рида-Соломона

Пусть $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^n$, где $\alpha_i \neq \alpha_j$ при $i \neq j$, $\beta = (\beta_1, \beta_2, \ldots, \beta_n) \in (\mathbb{F}_q^n)^*$. Тогда обобщённый код Рида-Соломона [8] можно определить следующим образом:

$$GRS_{n,k}(\alpha, \beta) = \{\beta_1 f(\alpha_1), \beta_2 f(\alpha_2), \ldots, \beta_n f(\alpha_n) \mid f \in \mathbb{F}_q[x]; \deg(f) < k\}$$

$$G = \begin{pmatrix} \beta_1 & \beta_2 & \ldots & \beta_n \\ \beta_1\alpha_1 & \beta_2\alpha_2 & \ldots & \beta_n\alpha_n \\ \ldots & \ldots & \ldots & \ldots \\ \beta_1\alpha_1^{k-1} & \beta_2\alpha_2^{k-1} & \ldots & \beta_n\alpha_n^{k-1} \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \ldots & \ldots & \ldots & \ldots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \ldots & \alpha_n^{n-k-1} \end{pmatrix} \cdot \begin{pmatrix} z_1 & 0 & \ldots & 0 \\ 0 & z_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & z_n \end{pmatrix}$$

Известны точные параметры обобщённых кодов Рида-Соломона.

Предложение 1. *Пусть C — $GRS_{n,k}$-код. Тогда $n = q - 1$, $d(C) = n - k - 1$*

Предложение 2. *GRS код имеет эффективный алгоритм декодирования, если количество исправляемых им ошибок $t \leqslant \lfloor \frac{d(GRS_{n,k})-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$*

Вообще говоря наличие эффективного алгоритма декодирования является одним из основных характеристик для практического применения кодов, исправляющих ошибки, в том числе и в криптографии.

Теорема 7. $GRS_{n,k}(\alpha, \beta)^\perp = GRS_{n,n-k}(\alpha, \gamma)$, *где* $\gamma_i = \beta_i^{-1} \prod\limits_{j=1, j\neq i}^{n} (\alpha_i - \alpha_j)^{-1}$

Теорема 8. $GRS_{n,k}(\alpha, \beta) = GRS_{n,k}(a \cdot \alpha + b, c \cdot \beta)$, *где* $a, c \in \mathbb{F}_q^*$ *и* $b = (b_1, \ldots, b_n) \in \mathbb{F}_q^n$

Теорема 8 является основополагающей для обоснования атаки Сидельникова-Шестакова на GRS-коды, которая описана в главе 4.

Криптосистемы, основанные на кодах, исправляющих ошибки

Криптоанализ асимметричной криптосистемы сводится к решению некоторой вычислительно трудной задачи. С появлением достаточно мощного квантового компьютера, некоторые вычислительно трудные задачи смогут быть решены гораздо быстрее, что делает криптосистемы на их основе уязвимыми. Постквантовая криптография занимается разработкой криптосистем, устойчивых к атакам квантового компьютера. Существует пять наиболее популярных подходов к реализации постквантовых криптосистем: криптосистемы на решётках, криптосистемы на кодах, изогении (эллиптические кривые), хэш-функции и уравнения от многих переменных [5]. Эти объекты хороши тем, что не существует ни квантовых, ни классических алгоритмов, для решения некоторых проблем, связанных с ними. В настоящей работе речь пойдёт о криптосистемах на основе кодов, исправляющих ошибки. Декодирование произвольного кода без информации о его структуре — искомая вычислительно трудная задача. Будут рассмотрены две криптосистемы, которые являются кандидатами на стандарт постквантовой криптографии, обе используют алгебраическую теорию кодирования, речь о которой шла ранее.

Криптосистема Мак-Элиса

Криптосистема была предложена Робертом Мак-Элисом в 1978 году [9]. В оригинальной статье предлагается использовать для шифрования коды Гоппа (В.Д. Гоппа, советский и российский математик). Автор отмечает более высокую скорость шифрования, чем у теоретико-числовых криптосистем, но, в то же время, больший вес ключа.

1. Генерация ключа:

   - Выбирается случайный линейный $[n, k]$-код C, исправляющий t ошибок. Вычисляется порождающая матрица G.
   - Генерируется случайная обратимая $k \times k$ матрица S
   - Генерируется случайная перестановочная $n \times n$ матрица P
   - Вычисляется $G_{pub} = SGP$

   Открытый ключ: (G_{pub}, t)

   Закрытый ключ: (S, G, P)

2. Шифрование:

   - $(m_1, m_2, \ldots, m_n)$ - открытый текст
   - Генерируется $z = (z_1, z_2, \ldots, z_n)$, имеющий вес t
   - $c = mG_{pub} + z$ - шифротекст

3. Расшифрование:

   - Вычисляется P^{-1}
   - $\tilde{c} = cP^{-1}$
   - К $\tilde{c}$ применяется алгоритм декодирования. Т.е. $\tilde{c} \to mSG$
   - Вычисляется $m = mS(GG^{-1})S^{-1}$

Криптосистема Нидеррайтера

Криптосистема была разработана Харальдом Нидеррайтером в 1986 году [10]. В отличие от криптосистемы Мак-Элиса, данная криптосистема использует не порождающую, а проверочную матрицу кода. Система позволяет создавать цифровые подписи, кроме того, представленные в данной главе криптосистемы являются устойчивыми к атаке с использованием квантового алгоритма Шора [13]. Несмотря на то, что криптосистема была взломана неквантовыми методами, некоторые её модификации остаются стойкими, это делает её кандидатом для постквантовой криптографии.

1. Генерация ключа:

   - Выбирается случайный линейный $[n, k]$-код C, исправляющий t ошибок. Вычисляется проверочная матрица H.
   - Генерируется случайная обратимая $(n-k) \times (n-k)$ матрица S, $\det(S) \neq 0$
   - Генерируется случайная перестановочная $n \times n$ матрица P
   - Вычисляется $H_{pub} = SHP$

Открытый ключ: (H_{pub}, t)

Закрытый ключ: (S, H, P)

2. Шифрование:

   - $(m_1, m_2, \ldots, m_n)$ — открытый текст
   - $c = m \cdot H_{pub}^T$ — шифротекст

3. Расшифрование:

   - $s = c \cdot (S^T)^{-1} = mP^T H^T S^T (S^T)^{-1} = (mP^T)H^T$
   - К s применяется алгоритм декодирования. Т.е. $s \to mP^T$
   - Вычисляется $m = mP^T (P^T)^{-1}$

Атака Сидельникова-Шестакова на криптосистему Нидеррайтера

Рассмотрим криптосистему Нидеррайтера, где в качестве $[n, k]$-кода C выбран $GRS_{n,k}(\alpha, \beta)$-код. Вообще говоря, в статье [2] показано, что криптостойкость системы Нидеррайдера эквивалентна системе Мак-Элиса ввиду того, что проверочная матрица $GRS_{n,k}$-кода является порождающей матрицей $GRS_{n,n-k}$-кода. Следовательно данная атака [1] применима и к криптосистеме Мак-Элиса.

Как было сказано в предыдущей главе, открытым ключом криптосистемы Нидеррайтера является матрица $H_{pub} = SHT$, а закрытым — матрицы (S, H, T). Идея атаки заключается в использовании жёсткой структуры GRS-кодов для нахождения матриц $S, \tilde{H}$:

$$H_{pub} = S\tilde{H}, \;\; \tilde{H} = HP \tag{2}$$

где

$$H_{pub} = \begin{pmatrix} f_1(\alpha_1) & f_1(\alpha_2) & \ldots & f_1(\alpha_n) \\ f_2(\alpha_1) & f_2(\alpha_2) & \ldots & f_2(\alpha_n) \\ \ldots & \ldots & \ldots & \ldots \\ f_{n-k}(\alpha_1) & f_{n-k}(\alpha_2) & \ldots & f_{n-k}(\alpha_n) \end{pmatrix} \cdot \begin{pmatrix} z_1 & 0 & \ldots & 0 \\ 0 & z_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & z_n \end{pmatrix},$$

где $\alpha_i \in F_q \cup \infty$ — поле, к которому добавлен элемент ∞, обладающий естественными свойствами. При $\alpha_j = \infty$ соответствующий столбец имеет вид $(0, \ldots, 0, z_j)^T$.

Лемма 1. *Любое дробно-линейное отображение $\varphi : F_q \to F_q$ осуществляет перестановку координат q-вектора $a = (a_1, a_2, \ldots, a_q)$: $(\varphi(a_1), \varphi(a_2), \ldots, \varphi(a_q)) \longmapsto (a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(q)}), \sigma \in S_q$*

Исходя из леммы 1 и теоремы 8 можно сделать вывод, что для взлома криптосистемы достаточно найти какое-то частное решение матричного уравнения (2), так как для любого дробно-линейного отображения ϕ существуют такие $(z'_1, z'_2, \ldots, z'_n)$ и матрица S_ϕ, что $(S \cdot S_\phi^{-1}, \phi(\alpha_1), \ldots, \phi(\alpha_n); z'_1, \ldots, z'_n)$ тоже являются решением системы (2), если $(S, \alpha_1, \alpha_2, \ldots, \alpha_n; z_1, \ldots, z_n)$ - решение системы.

Таким образом, положим $\alpha_1 = 1$, $\alpha_2 = 0$, $\alpha_3 = \infty$, где $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}_q \cup \infty$. Следовательно существует решение уравнения (2) вида $(S, 1, 0, \infty, \alpha_4, \ldots, \alpha_n; z'_1, \ldots, z'_n)$

Сначала восстановим вектор α. Для этого рассмотрим отдельные столбцы $\{1, n-k+1, \ldots, 2(n-k-1)\}$ матрицы H_{pub}. Получившуюся матрицу обозначим за H'_{pub}

Найдём ненулевой вектор $\vec{c}_1$ из левого ядра матрицы H'_{pub}, решив систему уравнений для $j = 1, n-k+1, \ldots, 2(n-k-1)$.

$$z_j \sum_{i=1}^{n-k} c_{1,i} f_i(\alpha_j) = 0$$

Положим $F_1(x) = \sum\limits_{i=1}^{n-k} c_{1,i} f_i(x)$. Так как все числа z_j отличны от нуля, то $\alpha_1, \alpha_{n-k+1}, \ldots, \alpha_{2(n-k-1)}$ являются его корнями. Следовательно $F_1(x) = a_1(x-\alpha_1)(x-\alpha_{n-k+1})\ldots(x-\alpha_{2(n-k-1)})$.

Теперь найдём ненулевой вектор $\vec{c}_2$ из левого ядра матрицы, состоящей из столбцов $\{2, n-k+1, \ldots, 2(n-k-1)\}$ матрицы H_{pub}.

$$z_j \sum_{i=1}^{n-k} c_{2,i} f_i(\alpha_j) = 0$$

Положим $F_2(x) = \sum\limits_{i=1}^{n-k} c_{2,i} f_i(x) = a_2(x-\alpha_2)(x-\alpha_{n-k+1})\ldots(x-\alpha_{2(n-k-1)})$.

Заметим, что при $3 \le j \le n-k$ и при $j \ge 2(n-k-1)+1$, α_j не являются корнями многочленов $F_1(x), F_2(x)$. Следовательно можем рассмотреть отношение:

$$\frac{z_j F_1(\alpha_j)}{z_j F_2(\alpha_j)} = \frac{\cancel{z_j} a_1(\alpha_j - \alpha_1)}{\cancel{z_j} a_2(\alpha_j - \alpha_2)},\ 3 \le j \le n-k,\ j \ge 2(n-k-1)+1 \tag{3}$$

$$\alpha_j = \frac{\frac{a_1}{a_2}}{\frac{a_1}{a_2} - \frac{z_j F_1(\alpha_j)}{z_j F_2(\alpha_j)}}$$

Для вычисления значения $\frac{z_j F_1(\alpha_j)}{z_j F_2(\alpha_j)}$ все данные имеются, $\alpha_1 = 1$, $\alpha_2 = 0$ по заданию. Осталось вычислить значение $\frac{a_1}{a_2}$. Для этого рассмотрим случай $j = 3$ в отношении (3). Напомним, что $\alpha_3 = \infty$.

$$\frac{z_3 F_1(\alpha_3)}{z_3 F_2(\alpha_3)} = \frac{\cancel{z_3} a_1(\alpha_3 - \alpha_1)}{\cancel{z_3} a_2(\alpha_3 - \alpha_2)} = \frac{a_1\alpha_3 - a_1\alpha_1}{a_2\alpha_3 - a_2\alpha_2} = \frac{a_1 \frac{a_1\alpha_1}{\alpha_3}}{a_2 \frac{a_2\alpha_2}{\alpha_3}} = \frac{a_1}{a_2}$$

Таким образом, мы можем вычислить α_j для j из интервалов, указанных в (3).

Аналогично найдём ненулевые векторы $\vec{c}_3$, $\vec{c}_4$ из левых ядер матриц, состоящих из столбцов $\{1, 3, \ldots, n-k\}$ и $\{2, 3, \ldots, n-k\}$, соответственно.

Положим $F_3(x) = \sum\limits_{i=1}^{n-k} c_{3,i} f_i(x)$ и $F_4(x) = \sum\limits_{i=1}^{n-k} c_{4,i} f_i(x)$, и рассмотрим отношение

$$\frac{z_j F_3(\alpha_j)}{z_j F_4(\alpha_j)} = \frac{\cancel{z_j} a_3(\alpha_j - \alpha_1)}{\cancel{z_j} a_4(\alpha_j - \alpha_2)},\ n-k+1 \le j \le 2(n-k-1)$$

$$\frac{z_n F_3(\alpha_n)(\alpha_n - \alpha_2)}{z_n F_4(\alpha_n)(\alpha_n - \alpha_1)} = \frac{a_3}{a_4}$$

$$\alpha_j = \frac{\frac{a_3}{a_4}}{\frac{a_3}{a_4} - \frac{z_j F_3(\alpha_j)}{z_j F_4(\alpha_j)}},\ n-k+1 \le j \le 2(n-k-1)$$

Таким образом, мы нашли вектор $\vec{\alpha}$. Для удобства дальнейших вычислений выберем $a \in \mathbb{F}_q$ отличное от всех α_j и заменим каждое α_j на $1/(a-\alpha_j)$. Полученный набор тоже является решением уравнения (2).

Теперь приступим к нахождению матрицы S и элементов z_j из матрицы $H_{pub} = S \cdot V_{(\alpha_1,\ldots,\alpha_n)} \cdot D, D = diag(z_1, \ldots z_n)$. Заметим, что если каждый элемент матрицы S умножить на $a \in F_q, a \neq 0$, а каждый элемент матрицы D умножить на a^{-1}, то их произведение не изменится, поэтому можно положить $z_1 = 1$. Дальше будем рассматривать матрицу H'_{pub}, полученную из H_{pub} отбрасыванием $k-1$ последних столбцов. Найдём вектор $c = (c_1, \ldots, c_{n-k+1}) : H'_{pub} \cdot c^T = 0$, для этого решим систему из $n-k$ линейных уравнений с $n-k+1$ неизвестными. Система имеет нетривиальное решение, потому что, иначе, исходная матрица H_{pub} имела бы $n-k$ линейно зависимых столбцов, что невозможно, так как H_{pub} есть произведение обратимой матрицы S на матрицу Вандермонда V и на диагональную матрицу D:

$det(H_{pub}) = det(S) \cdot det(V_{(\alpha_1,\ldots,\alpha_n)}) \cdot det(D) \neq 0, D = diag(z_1, \ldots z_n)$.

Теперь можно положить $C' = diag(c_1, \ldots, c_{n-k+1})$, $D' = diag(z_1, \ldots, z_{n-k+1})$, $z = (z_1, z_2, \ldots, z_{n-k+1})$ и переписать уравнение в виде:

$$\begin{aligned} H'_{pub}c^T &= SV_{(\alpha_1,\ldots,\alpha_{n-k+1})}D'c^T \\ &= SV_{(\alpha_1,\ldots,\alpha_{n-k+1})} \cdot C'z^T \\ &= S^{-1}SV_{(\alpha_1,\ldots,\alpha_{n-k+1})}C'z^T \\ &= V_{(\alpha_1,\ldots,\alpha_{n-k+1})}C'z^T \\ &= 0. \end{aligned} \tag{4}$$

Получаем систему уравнений относительно n-k неизвестных $z_i, i \in [2, n-k+1]$ с матрицей коэффициентов $V_{(\alpha_1,\ldots,\alpha_{n-k+1})} \cdot C'$, где все c_i и α_{ij} нам уже известны. Данная система имеет единственное решение: $det(V_{(\alpha_1,\ldots,\alpha_{n-k+1})} \cdot C') \neq 0$. Решая систему, находим элементы $z_2, \ldots, z_{n-k+1}$.

Пусть H''_{pub} — матрица, состоящая из $n-k$ первых столбцов матрицы H_{pub}, а $\tilde{H}''$ - $(n-k) \times (n-k)$ матрица, состоящая из $n-k$ первых столбцов матрицы $\tilde{H}$. Решив систему (4), мы нашли элементы $z_1, \ldots, z_{n-k+1}$. Следовательно, элементы матрицы $\tilde{H}''$ полностью известны.

$$S = H''_{pub} \cdot (\tilde{H}'')^{-1}$$

Далее вычисляем матрицу S^{-1} и находим оставшиеся z_j из

$$\tilde{H} = S^{-1}H_{pub}$$

Уравнение $H_{pub} = S\tilde{H}$ решено, секретная матрица $S^{-1} \cdot H_{pub} = \tilde{H}$ получена.

Система Berger-Loidreau: подкоды обобщенных кодов Рида-Соломона

Описание схемы

Одна из модификаций обобщённых кодов Рида-Соломона была предложена Thierry P. Berger и Pierre Loidreau в [4]. Было предложено для маскировки структуры обобщенного кода Рида-Соломона $GRS_{n,k}$ с параметрами $[n, k, d]$ использовать подкод C кода $GRS_{n,k}$ с параметрами $[n, k-l, d']$, где l мало и $d' \geq d$. Очевидно, что алгоритм декодирования $GRS_{n,k}$ можно использовать для C, чтобы исправить до $t = \lfloor (d-1)/2 \rfloor$ ошибок.

Простейший способ построить такой подкод — добавить l строк в проверочную матрицу $GRS_{n,k}$. Введем необходимые обозначения. Пусть M_d — порождающая матрица $GRS^{\perp}_{n,k}$, т.е. проверочная матрица $GRS_{n,k}$ размера $(d-1) \times n$. $l \times n$-матрица A, такая что строки линейно независимы от

строк H. Невырожденная матрица S размера $(d-1+l)\times(d-1+l)$. Тогда проверочная матрица подкода C будет иметь вид:

$$H_{pub} = S\left(\frac{M_d}{A}\right).$$

Анализ устойчивости

Также был проведен анализ этой схемы по поводу ее сопротивления атаке Сидельникова-Шестакова. Пусть $M = (I|B)$ — порождающая матрица $GRS_{n,k}(\alpha,\beta)$ в систематической форме. Очевидно, что с точностью до перестановки столбцов существует порождающая матрица подкода C вида: $\left(\frac{I_{k,k}|B}{\mathbf{0}|I_{l,l}|T}\right)$, где $B = (b_{i,j})$ при $i = 1,\ldots,k$ и $j = k+1,\ldots,n$; $T = (t_{i,j})$ при $i = 1,\ldots,l$ и $j = k+l+1,\ldots,n$. Поскольку известна H_{pub}, то можно вычислить порождающую матрицу C в систематической форме: $\left(I_{k+l,k+l}|R\right)$, где $R = (r_{i,j})$ при $i = 1,\ldots,k+l$ и $j = k+l+1,\ldots,n$.

Следовательно,

- При $i = 1,\ldots,k$ и $j = k+l+1,\ldots,n$ $r_{i,j} = b_{i,j} - \sum_{s=1}^{l} t_{s,j}b_{i,k+s}$.
- При $i = k+1,\ldots,k+l$ и $j = k+1,\ldots,n$ $r_{i,j} = t_{i-k,j}$.

Заметим, что матрица R известна, а значит известна и T.

Чтобы развить атаку Сидельникова-Шестакова, нужно восстановить непосредственно $b_{i,j}$ из $k\times(n-k-l)$ уравнений. Очевидно, что эта система имеет q^{kl} решений и только одно из них верно. Для обычных значений q,k,l это невозможно.

Усовершенствованная атака на криптосистему Berger-Loidreau

Позднее на эту систему была разработана атака, которую впоследствии усовершенствовал в своей работе C. Wieschebrink [14]. Рассмотрим усовершенствованную атаку более подробно.

Пусть $M = SG$ — открытая матрица криптосистемы Berger-Loidreau, которая является порождающей матрицей $(k-l)$-мерного подкода $GRS_{n,k}(\alpha,\beta)$. Представляется алгоритм восстановления секретных параметров α,β из M, который выполним даже для больших l.

Пусть $r_1,\ldots,r_m$ — строки матрицы M, а $f_1,\ldots,f_m (m = k-l)$ — связанные с этими строками полиномы. Для двух векторов-строк $a,b\in\mathbb{F}^n$ определим покомпонентное произведение $a*b := (a_1b_1,a_2b_2,\ldots,a_nb_n)$. Для данной атаки различается два случая. Сначала рассматривается случай $2k-1\leqslant n-2$. Тогда атака работает следующим образом. Вычисляется r_i*r_j для всех $i,j\in 1,\ldots,m, i\leqslant j$. Очевидно, что r_i*r_j имеет вид $r_i*r_j = (\beta_1^2 f_i(\alpha_1)f_j(\alpha_1),\ldots,\beta_n^2 f_i(\alpha_n)f_j(\alpha_n))$, а так как степень $f_if_j\leqslant 2k-2$, код C, порожденный r_i*r_j, является подкодом $GRS_{n,2k-1}(\alpha,\beta')$, где $\beta' = (\beta_1^2,\ldots,\beta_n^2)$. Если $C = GRS_{n,2k-1}(\alpha,\beta')$, то атака Сидельникова-Шестакова может быть применена к порождающей матрице C, возвращающей β',α. Если характеристика поля равна 2, вектор можно вычислить из β' напрямую (путем применения обратного оператора Фробениуса), в противном случае β можно восстановить из M.

В противном случае, если $C\neq GRS_{n,2k-1}(\alpha,\beta')$, считается, что эта атака не удалась. Поскольку время выполнения первоначальной атаки на эту систему в значительной степени зависит от $l = k-m$, по крайней мере, можно применить ее к порождающей матрице C, если $0<2k-1-\dim(C)<l$. Однако, для не слишком большого значения l вероятность того, что $C = GRS_{n,2k-1}(\alpha,\beta')$, кажется очень высокой.

Для типичных примеров криптосистемы Berger-Loidreau может применяться случай $2k-1>n-2$. Первоначальная атака здесь вообще не работает, так как алгоритм Сидельникова-Шестакова

не может быть применен или код, сгенерированный $r_i * r_j$, может быть равен $\mathbb{F}^n$. Однако идея покомпонентного умножения кодовых слов может быть применена к укороченному коду $\langle M \rangle$.

Определение 20. Пусть $C \subset \mathbb{F}^n$ — линейный код длины n и размерности k, и пусть $d \in \mathbb{N}^{\leqslant k}$. *Сокращенный код* $S_d(C)$ состоит из всех кодовых слов $(s_1, \ldots, s_{n-d}) \in \mathbb{F}^{n-d}$ таких, что $(\underbrace{0, \ldots, 0}_{d \text{ раз}}, s_1, \ldots, s_{n-d}) \in C$. Учитывая порождающую матрицу $G_C = [1_k | E]$ кода C в ступенчатой форме (где E обозначает $k \times (n-k)$-матрицу), базис $S_d(C)$ может быть легко получен путем извлечения $n-d$ крайних правых компонент последних $k-d$ строк G_C.

Пусть теперь M снова будет открытой $(m \times n)$-матрицей криптосистемы Berger-Loidreau, а G_{S_d} будет порождающей матрицей $S_d(\langle M \rangle)$. По строке $s = (s_1, \ldots, s_{n-d})$ матрицы G_{S_d} имеем $(0, \ldots, 0, s_1, \ldots, s_{n-d}) \in GRS_{n,k}(\alpha, \beta)$, так что s можем записать как $s = (\beta_{d+1} f(\alpha_{d+1}), \ldots, \beta_n f(\alpha_n))$, где $f(\beta) \in \mathbb{F}[\beta]$ имеет вид $f(\beta) = g(\beta) \prod_{j=1}^{d} (\beta - \alpha_j)$ со степенью $g(x) \leqslant k-d-1$. Пусть $z := (\beta_{d+i} \prod_{j=1}^{d} (\alpha_{d+i} - \alpha_j))_{i=1,\ldots,n-d}$ и $\alpha' := (\alpha_{d+1,\ldots,\alpha_n})$ очевидно, что $S_d(\langle M \rangle) = \langle G_{S_d} \rangle \subset GRS_{n-d,k-d}(\alpha', z)$. Если можно выбрать таким образом, что $d \leqslant m-1$ и $2(k-d)-1 \leqslant n-d-2$, алгоритм можно применить к G_{S_d}, который в случае успеха дает не более $q(q-1)^2$ кандидатов на α' (не более $q-1$ кандидатов, если предположить, что, например, $\alpha'_{n-1} = 1$, $\alpha'_n = 0$). Пусть T —- множество этих решений. Когда T известно, оставшиеся $\alpha_1, \ldots, \alpha_d$ можно вычислить аналогичными методами. Ниже приводится альтернативный подход.

Пусть $m^{(1)}, \ldots, m^{(n)}$ — векторы-столбцы матрицы M, а $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ перестановка. Обозначим через M_π матрицу, полученную из M перестановкой столбцов в соответствии с π, т.е. $M_\pi = (m^{(\pi(1))}, \ldots, m^{(\pi(n))})$. Аналогично для $y := (y_1, \ldots, y_n) \in \mathbb{F}^n$ определим $y_\pi := (y_{\pi(1)}, \ldots, y_{\pi(n)})$. Очевидно, что

$$\langle M_\pi \rangle \subset GRS_{n,k}(\alpha_\pi, \beta_\pi).$$

Для простоты полагаем, что $2d \leqslant n-3$ (для типичных случаев d можно выбрать таким образом, однако метод легко обобщается на общий случай). Пусть теперь $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ задано формулой

$$\pi(i) = \begin{cases} i+d, 1 \leqslant i \leqslant d \\ i-d, d+1 \leqslant i \leqslant 2d \\ i, 2d+1 \leqslant i \leqslant n \end{cases}.$$

Применяется алгоритм к M_π, т.е. вычисляется порождающая матрица G'_{S_d} из $S_d(\langle M_\pi \rangle)$ и умножается каждая пара строк G'_{S_d} покомпонентно. Можно найти множество T' кандидатов в $(\alpha_{\pi(d+1)}, \ldots, \alpha_{\pi(n)})$. Решение для α можно восстановить, найдя $\sigma \in T$ и $\tau \in T'$, где $\sigma_{d+1} = 0$, $\sigma_{d+2} = 1$ и $\sigma_i = \tau_i$ для $i = d+1, \ldots, n-d$ и определив

$$\begin{aligned} &\alpha_i = \tau_i \text{ для } 1 \leqslant i \leqslant d, \\ &\alpha_i = \sigma_{i-d} \text{ для } d+1 \leqslant i \leqslant 2d, \\ &\alpha_i = \sigma_{i-d} = \tau_{i-d} \text{ для } 2d+1 \leqslant i \leqslant n. \end{aligned}$$

Система BBCRS: маскирующая матрица

Для того чтобы можно было избежать атаки Сидельникова-Шестакова матрица P не должна быть перестановочной. В 2016 году Baldi, Bianchi, Chiaraluce, Rosenthal и Schipani [3] была предложена схема, использующая данные идеи. Ниже приводится описание данной схемы.

Ограничимся построением схемы в версии McEliece, т.к. она эквивалента схеме в версии Niederreiter.

Пусть $\mathbb{F}_q$ конечное поле, $1 \leqslant k < n \leqslant q$ целые числа. Обозначим за G порождающую матрицу размера k × n линейного кода C над полем $\mathbb{F}_q$. Выберем любую обратимую матрицу S размера $k \times k$. Теперь вместо перестановочной матрицы P введём матрицу $Q = R + L$, где R и L определяются следующим образом:

1) L — разреженная матрица размера $n \times n$ со строкой и столбцом веса m.

2) Выберем матрицы размера $z \times n$ $(z \leq n)$ $a_1, a_2, \cdots, a_\omega; b_1, b_2, \cdots, b_\omega$, тогда матрица R запишется в следующем виде.

$$R = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_\omega \end{pmatrix}^\top \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_\omega \end{pmatrix}.$$

Опишем алгоритмы зашифрования и расшифрования. Вычисляем $G_{pub} = S^{-1}GQ^{-1}$, тогда **закрытым** ключом будут матрицы (S, G, Q) и **открытым** $(G_{pub}, t_{pub} = \lfloor t/m \rfloor)$. Возьмем сообщение $x \in \mathbb{F}_q^k$, которое хотим зашифровать и вектор ошибок $e \in \mathbb{F}_q^n$ с весом $w(e) \leqslant t_{pub}$. Тогда шифр вычисляется как $y = xG_{pub} + e$, умножим справа на матрицу Q, получим $yQ = xS^{-1}G + eQ$. Рассмотрим 2 случая с $\omega = 2$:

1) $a_1 = a$, $a_2 = \mathbf{0}$, где $\mathbf{0}$ — матрица, у которой все элементы нулевые

$$R = \begin{pmatrix} a \\ \mathbf{0} \end{pmatrix}^\top \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

2) $b_1 = b$, $b_2 = \mathbf{1} + b$, где $\mathbf{1}$ — матрица, у которой все элементы равны 1.

$$R = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^\top \cdot \begin{pmatrix} b \\ \mathbf{1} + b \end{pmatrix}.$$

При этом должно выполняться равенство $\sum_{i=1}^{\omega} a_i e^T = 0$ (это требование на a_i будет открытым, но в [3] показано, как можно избежать этой потенциальной уязвимости), следовательно, eR можно записать так:

$$eR = \begin{cases} ea^T b_1 = \mathbf{0}, & \text{при } a_1 = a, a_2 = \mathbf{0}, \\ e(a_1^T b + a_2^T \mathbf{1} + a_2^T b) = ea_2^T \mathbf{1}, & \text{при } b_1 = b, b_2 = \mathbf{1} + b. \end{cases}$$

Тогда в первом случае получим, что $eQ = eL$, во втором $ea_2^T = \gamma \in \mathbb{F}_q^z$ и при $z = 1$ можно подобрать не более чем за q вариантов. Осталось заметить, что вектор $xS^{-1}G$ есть кодовое слово кода C и, поскольку, $w(eL) \leqslant t$ применяя декодирующий алгоритм получаем xS^{-1} и, умножая на S, получаем сообщение x.

На эту схему в работе Couvreur, Gaborit, Gauthier-Umana, Otmani и Tillich была предложена атака [6], работающая для некоторых значений параметров z и m: $z = 1, m \leqslant 1 + k/n + O(1/\sqrt{n})$.

ЛИТЕРАТУРА

[1] Сидельников В. М., Шестаков С. О. *О системе шифрования, построенной на основе обобщенных кодов Рида–Соломона* // Дискретная математика. 1992. Т. 4, вып. 3. С. 57–63.

[2] Сидельников В. М. *Открытое шифрование на основе двоичных кодов Рида–Маллера* // Дискретная математика. 1994. Т. 4, вып. 3. С. 191—207.

[3] Baldi M., Bianchi M., Chiaraluce F., Rosenthal J., Schipani D. *Enhanced Public Key Security for the McEliece Cryptosystem.* // Journal of Cryptology. 2016. V. 29. P. 1–27.

[4] Berger T. P., Loidreau P. *How to Mask the Structure of Codes for a Cryptographic Use* // Des Codes Crypt. 2005. V. 35, P. 63–79.

[5] Bernstein D. J. *Introduction to post-quantum cryptography* // Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009. P. 1–14.

[6] Couvreur A. Otmani A., Tillich J. P., Gauthier–Umana V. *A polynomial-time attack on the BBCRS scheme* // IACR International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2015. P. 175–193.

[7] Grover L. K. *A fast quantum mechanical algorithm for database search* // Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996. P. 212–219.

[8] MacWilliams F. J., Sloane N. J. A. *The theory of error correcting codes.* Elsevier. 1977. V. 16.

[9] McEliece R. J. *A public-key cryptosystem based on algebraic coding theory.* DSN progress report 42:44. 1978. P. 114–116.

[10] Niederreiter H. *Knapsack-type cryptosystems and algebraic coding theory.* Probl. control and inform. theory. 1986. V. 15. P. 19–34.

[11] Reed I. S., Solomon G. *Polynomial codes over certain finite fields* // Journal of the society for industrial and applied mathematics. 1960. V. 8. №. 2. P. 300–304.

[12] Rivest R. L., Shamir A., Adleman L. *A method for obtaining digital signatures and public-key cryptosystems* // Communications of the ACM. 1978. V. 21. no. 2. P. 120–126.

[13] Shor P. W. *Algorithms for quantum computation: discrete logarithms and factoring* // Proceedings 35th annual symposium on foundations of computer science. IEEE Computer Society, 1994. P. 124–134.

[14] Wieschebrink C. *Cryptanalysis of the Niederreiter Public Key Scheme Based on GRS Subcodes.* In: Sendrier N. (eds) Post-Quantum Cryptography. PQCrypto 2010. P. 61–72. (Lecture Notes in Computer Science, vol 6061)

Кураторы исследования –

к.ф.-м.н., ассистент кафедры теоретической кибернетики ММФ НГУ, н.с. ИМ СО РАН, Куценко Александр Владимирович;

лаборант ИМ СО РАН, студент ММФ НГУ, Бахарев Александр Олегович;

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

БЛОКЧЕЙН-ТЕХНОЛОГИИ

Реализация банковского смарт-контракта

В. Д. Боязитов
Томский Государственный Университет
E-mail: colpakov.vadik@yandex.ru

Аннотация

В данной работе рассматривается разработка смарт-контракта выполняющего простейшие банковские функции.
Ключевые слова: *Структура данных, Безопасность, Блокчейн, Ethereum.*

Актуальность

DeFi — это парадигма в основе которой лежит сервис, предоставляющий финансовые услуги и соответствующий принципам:

1.Открытый исходный код;

2.Доступность для каждого пользователя;

3.Финансовая прозрачность.

Цель DeFi — сделать международную банковскую систему открытой и свободной.

Децентрализация призвана гарантировать отсутствие единого слабого места и предотвратить полный выход системы из строя. Платформа Ethereum - удобный вариант для создания и запуска DeFi-приложений. Вы можете представить DeFi в виде финансовой экосистемы, открытой для всех, создавать собственные эффективные финансовые инструменты и услуги по своему усмотрению. Они децентрализованы и не имеют единой точки отказа.

Смарт-контракт - это программа, которая автоматически запускается в блокчейне при соблюдении определенных условий. Смарт-контракты позволяют разработчикам получать доступ к более сложным функциям. Поскольку все используемые протоколы имеют открытый исходный код, есть возможность создавать свои собственные финансовые продукты на их основе.

Цели и задачи проекта

1.Познакомиться с платформой Ethereum.

2.Написать смарт-контракт на языке Solidity.

3.Реализовать функции работы с депозитами и вкладами.

4.Провести тестирование разработанной системы.

Проектирование модели

При старте реализации проекта следует начать с определения логики его работы. Мы должны понимать, чего хотят конечные пользователи и с чем они будут взаимодействовать. Благодаря этому, мы экономим наше будущее время и деньги, если проект направлен на коммерческие цели. В ходе выполнения проекта могут меняться любые требования. После построения блок-схемы, следует внимательно изучить каждый элемент на безопасность и возможность обновления. Далее следует проделать аналогичные действия с полной блок-схемой. Следующим шагом является выбор структур данных, где будет храниться информация, кто будет иметь к ней доступ. Так

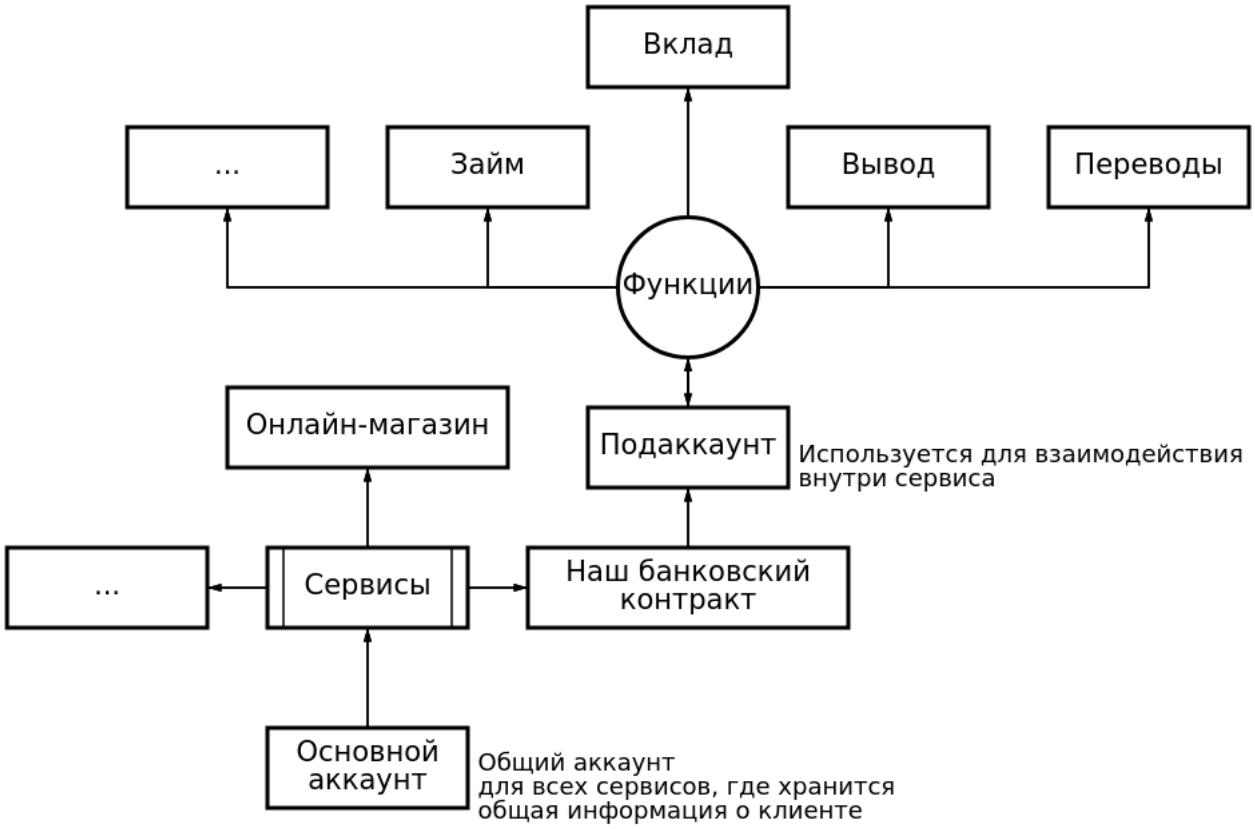

Рис. 1: Блок-схема для поставленной задачи

как информации слишком много, следует использовать алгоритм "Разделяй и властвуй". Следует начинать обход с корня блок-схемы. Соединяя блок-схему и множество структур данных выделяются объекты, глобальные переменные, коды ошибок. Проделав большую часть работы и получив блок-схему и структуры данных, можно приступать к программной реализации. Обычно на данном этапе могут появляться идеи, затруднения и ошибки. Вновь корректируется блок-схема и ее структуры данных. Пройдя данный этап, код переходит в стадию тестирования и анализа безопасности. На данных этапах могут возникать проблемы или идеи. Вновь корректируется код, структуры данных и блок-схема. После прохождения всех этапов выпускается альфа-версия приложения, которая отправляется пользователям на тестирование. После чего учитывается их мнение о продукте и предложения, вносятся соответствующие доработки.

Реализация логики приложения для поставленной задачи:

Создадим множество основных аккаунтов. Эти аккаунты смогут взаимодействовать с другими сервисами: социальными сетями, онлайн-магазинами, банком и т.д. У каждого сервиса будет свой набор подаккаунтов, которые смогут взаимодействовать внутри сервиса и обмениваться данными между основными аккаунтами. При создании основного аккаунта создается уникальный ID, который определяет пользователя в системе. В нашем случае при создании основного аккаунта вы сразу получаете подаккаунт для банка. Под банком подразумевается просто набор функций. Теперь ваш основной аккаунт и подаккаунт для банка связаны одним ID. Если добавляется новый

сервис, то взяв ID основного аккаунта, мы создадим подаккаунт для сервиса с таким же ID. Аккаунты хранятся в массиве, где ID это индекс аккаунта в массиве. Благодаря данной структуре данных, мы очень быстро можем обращаться к нужному аккаунту.

Solidity

Определим термины, которые будут использоваться в дальнейшем.

Блокчейн – это полностью распределенная одноранговая программная сеть, которая использует криптографию для защищенного хостинга приложений, хранения данных и легкой передачи цифровых финансовых инструментов, являющихся эквивалентом реально существующих денег. С помощью криптографических протоколов в Ethereum создается единая защищенная вычислительная среда из тысяч узлов, которая работает без централизованного управления и без единого владельца. Смарт-контракты в Ethereum пишутся на языке программирования Solidity. Полнота по Тьюрингу и возможность Ethereum сохранять состояния являются отличительными особеностями платформы. **Solidity** – это язык программирования, предназначенный для написания программ, называемых смарт-контрактами, которые могут запускаться виртуальной машиной Ethereum. Solidity является высокоуровневым контрактно-ориентированным языком, схожим с JavaScript и языками C. Он позволяет разрабатывать контракты и компилировать их в байт-код EVM. В настоящее время это флагманский язык для Ethereum. В языке Solidity определены следующие типы данных:

1. int и uint - знаковый и без знаковый тип. Размерность по умолчанию 256 бит.
2. bool - логический тип.
3. address - тип, содержащий 20-байтовое значение, которое позволяет хранить адрес аккаунта Ethereum (40 шестнадцатеричных символов или 160 бит).
4. string, struct и другие стандартные типы.

Основной функционал

На практике были реализованы все поставленные задачи и добавлены новые функции. В контракт включены следующие функции: переводы внутри банка, пополнение основного аккаунта, перевод с основного аккаунта на аккаунт сервиса, вклад с ограничением на вывод по времени и без, VIP-статус. Также добавлена функция быстрого займа (для арбитражной торговли), в котором небольшая часть денег при создании аккаунта отправляется в благотворительный фонд.

Начисление процентов

Расчет начисляемых процентов происзодит следующим образом. Пусть мы запустили таймер и он ведет отсчет с 0 секунд. Мы предлагаем следующие условия: 50 процентов за каждые 30 секунд и возможность осуществить вывод в любой момент времени. Пусть спустя 30 секунд от начала отсчета приходит 100р, это время обозначим за t1. Через 30 секунд после вложения, вкладчик забирает свои деньги и процент, это время обозначим за t2. Определение итогового процента:

1.(t2 - t1) / 30. Где 30 - период за который начисляется процент.
(60 - 30) / 30 = 1. Это коэффициент который будем умножать на наш процент. Если коэффициент меньше единицы, то вывод средст произошел раньше, чем прошло 30 секунд. Например положили в 30, а вывели в 45. (45 - 30)/ 30 = 0.5, умножаем 0.5 на 50 процентов и получаем 25 процентов. Если коэффициент больше 1, то аналогично.

2. Готовая формула (Разница времени) / (Оговоренный период времени)

Таким образом пользователь может забирать депозит с процентами в любое удобное время, доступ к депозиту имеет только он. Это касается накопительного счета "Short депозит". Так же реализован вклад "Long депозит". Разница в том, что в первом случае у нас меньше процент и вывод средств доступен в любой момент времени, а во втором – вклад зафиксирован и не может быть получен раньше обговоренного срока.
Замер времени происходит с помощью встроенных механизмов Ethereum, который позволляет получить значение текущего времени блока в формате Unix timestamp.

Результат

Все поставленные цели и задачи выполнены успешно. В результате создан финансовый инструмент под платформу Ethereum выполняющий следующие банковские функции: создание счета, пополнение и вывод, внесение вклада, быстрый займ.
В дальнейшем планируется осуществить следующие шаги по модернизации и оптимизации проекта:
1. Реализация интерфейса для пользователей. Интеграция Web3.
2. Оптимизация кода для меньшего потребления газа при проведении транзакций.
3. Добавление нового функционала.
4. Реализация других сервисов.

ИСТОЧНИКИ

[1] https://ethereum.org/en/whitepaper/.

[2] https://ethereum.org/en/developers/docs/.

[3] https://www.youtube.com/c/YuliyaBedrosova.

[4] https://docs.soliditylang.org/en/latest/.

[5] https://hardhat.org/getting-started.

[6] https://www.youtube.com/c/IlyaBodrovKrukowski.

[7] https://docs.soliditylang.org/en/latest/.

[8] https://vc.ru/crypto/149649-vse-chto-nuzhno-znat-o-defi.

[9] https://vc.ru/finance/276929-perevod-decentralizovannye-finansy-defi.

Кураторы исследования –
Е. Мельничук Ассистент БФУ им. Канта;
Д. Кондырев Ассистент кафедры компьютерных систем ФИТ НГУ;

Разработка смарт-контракта, позволяющего создавать, покупать и продавать NFT

Г. А. Жуков[1]
[1]Новосибирский государственный университет
E-mail: g.zhukov@g.nsu.ru

Аннотация

Сейчас большую популярность приобрели NFT. NFT (non-fungible token) - это сертификат, доказывающий, что вы обладаете уникальным объектом в цифровом пространстве. Этим объектом может быть фотография, музыкальная запись и многое другое. NFT используется во многих областях: цифровое искусство, игры (игровые предметы, например, скины, персонажи), бизнес, недвижимость (для подтверждения собственности). Так же это отличный способ перенести уникальные предметы из реального мира в блокчейн. Каждый из этих токенов неповторим, неразделим и существует в единственном числе. Кроме того, в блокчейне надежно хранится вся необходимая информация о нем. С появлением NFT, появились и стандарты для них. Один из самых известных стандартов - это ERC-721. Он предоставляет такие функции, как перенос токенов с одной учетной записи на другую, ,получение текущего баланса токенов учетной записи, получение владельца определенного токена, а также общее количество токенов, доступных в сети. С полным списком функций можно познакомиться в [6].

В рамаках данного проекта был реализован смарт-контракт, позволяющий каждому пользователю создавать свои собственные NFT токены (удовлетворяющие стандарту ERC-721) и продавать их.

Ключевые слова: *блокчейн, смарт-контракт, язык программирования Solidity, NFT.*

Введение

Технология блокчейн — это усовершенствованный механизм базы данных, который позволяет организовать открытый обмен информацией. База данных блокчейна хранит данные в блоках, связанных между собой в цепочку. Данные являются хронологически последовательными, поскольку нельзя удалять или изменять цепочку без достижения консенсуса между участниками сети. Основные свойства блокчейна:

- Децентрализация: Децентрализация в блокчейне означает передачу контроля и принятия решений от централизованного субъекта (отдельного лица, организации или их группы) к распределенной сети.
- Неизменность: Неизменность означает, что данные не могут быть изменены. Ни один участник не может вмешаться в транзакцию после ее внесения в реестр. Если запись содержит ошибку, то для ее исправления необходимо добавить новую транзакцию. В сети будут отображены обе транзакции.
- Консенсус: Система блокчейн устанавливает набор правил, с помощью которых участники одобряют транзакции. Новые транзакции можно регистрировать только с согласия большинства участников сети.

Благодаря этим свойствам технология блокчейн и стала настолько популярной.
Одной из реализаций блокчейн технологии является платформа Ethereum. Ethereum — это управляемая сообществом технология, обеспечивающая работу криптовалюты «эфир» (ETH) и тысяч

децентрализованных приложений. Эта платформа предоставляет возможность использовать смарт-контракты. Смарт-контракт - это компьютерная программа, хранящаяся в блокчейне, которая позволяет нам преобразовывать традиционные контракты в их цифровые аналоги. Появление смарт-контрактов открыло широкие возможности.
В настоящее время популярным языком для написания смарт-контрактов является язык программирования Solidity, а популярной средой для разработки - Remix IDE.

Разработка смарт-контракта

Разработка смарт-контракта включала в себя 2 этапа:

1. Изучение технологии блокчейн, языка Solidity, сред разработки Remix IDE и Hardhat, NFT.
2. Разработка концепции смарт-контракта и последующая её реализация.

Основная идея задуманного смарт-контракта - это предоставить пользователям возможность создавать свои NFT и продавать их. Поэтому важной частью смарт-контракта является структура User, которая включает в себя следующие ключевые поля:

- string name (Название NFT);
- string symbol (Символ NFT);
- string description (Описание NFT);
- address payable UserAddress(адрес, который будет использоваться для зачисления эфиров, в случае продажи NFT);
- uint price (цена NFT);
- string metadataURI (ссылка на IPFS хранилище, по которой хранятся данные об NFT);

Основные функциональные возможности реализованного смарт-контракта можно посмотреть на UML диаграмме (см. Рис. 1).
И заключительная часть это тестирование написанного смарт-контракта.
В дальнейшем можно провести аудит смарт-контракта, т.е протестировать его на возможные уязвимости, так же можно попробовать реализовать идею смарт-контракта, не в рамках одного контракта (как сейчас), а с использованием нескольких контрактов, что, возможно, сможет повысить эффективность и функциональные возможности.

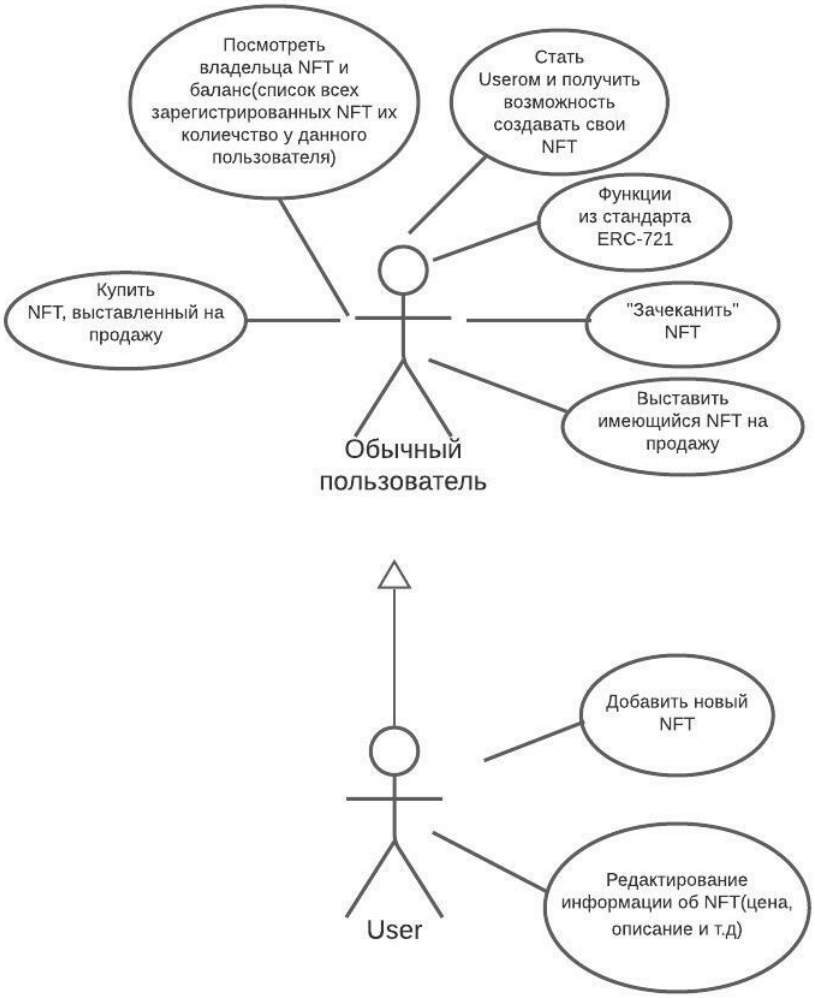

Рис. 1: Функциональные возможности смарт-контракта

ЛИТЕРАТУРА

[1] Платформа Ethereum:https://ethereum.org/en/

[2] Язык Solidity (документация): https://docs.soliditylang.org/en/v0.8.15/

[3] Среда разработки Remix IDE (документация): https://remix-ide.readthedocs.io/en/latest/

[4] Информация о блокчейне: https://aws.amazon.com/ru/what-is/blockchain

[5] Информация об NFT: https://sovcombank.ru/blog

[6] ERC-721: https://ethereum.org/ru/developers/docs/standards/tokens/erc-721/

Кураторы исследования –
Кондырев Д.О;
Мельничук Е. М.;

Реализация смарт-контракта, выпускающего NFT

И. В. Исаков[1], Р.А. Маслов[1], В. В. Скудина[2], Н.Д. Атутова[2]
[1]Новосибирский государственный технический университет
[2]Новосибирский государственный университет
E-mail: einBesserwisser@yandex.ru, tr4in33@yandex.ru, v.skudina@g.nsu.ru, n.atutova@g.nsu.ru

Аннотация

В 2021 году NFT-токены резко набрали популярность и стали известны всему миру. Тем не менее, они до сих пор вызывают много вопросов, так как большинству неизвестен механизм работы смарт-контрактов и блокчейна в целом. В рамках данного проекта был исследован механизм работы децентрализованных блокчейн-систем, в частности, платформы Ethereum, устройство выпуска NFT и написания смарт-контрактов, язык Solidity для реализации. Был реализован смарт-контракт для выпуска NFT, токенами являются части qr-кода, который ведет на сайт группы крипто-школы в социальной сети "Вконтакте".

Ключевые слова: *блокчейн, распределенные системы, Ethereum, NFT, Solidity.*

Блокчейн

Блокчейн — это разновидность децентрализованных систем, которая занимается сбором данных, их хранением и управлением. Формируется как цепочка блоков, каждый из которых содержит информацию в виде набора транзакций о времени и ссылку на предыдущий блок. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную хеш-сумму и хеш-сумму предыдущего блока. Копии цепочек блоков обычно хранятся на множестве узлов.

Ethereum

Основная работа велась с блокчейном на платформе Ethereum [5]. 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.

Смарт контракты

Смарт контракты - компьютерные протоколы, которые позволяют проводить сделки, контролируют их исполнение и хранятся на блокчейн-платформе. Их выполнение не зависит от третьих лиц, а код нельзя произвольно изменить. Блокчейн Bitcoin использует неполный по Тьюрингу язык программирования Script. Bitcoin поддерживает простые смарт-контракты с мультиподписью (для выполнения действия нужны цифровые подписи нескольких участников), удержанием средств на установленное время и так далее. Блокчейн Ethereum работает со смарт-контрактами на Тьюринг-полных языкых программирования, например, Solidity. Solidity позволяет создавать более сложные алгоритмы, но на их основе смарт-контракты гораздо сложнее проверить на наличие уязвимостей.

NFT

NFT означает non-fungible token или невзаимозаменяемый токен. Токен — это своего рода ценная бумага в цифровом мире [13]. Покупка токена означает, что покупатель завладел неким активом. Обычные токены взаимозаменяемы, аналогично денежным купюрам, а невзаимозаменяемые обладают уникальными чертами. Таким образом, NFT - это уникальные цифровые сертификаты, которые подтверждают оригинальность объекта и дают на него эксклюзивные права. Записи хранятся в блокчейне.

Идея проекта

В качестве уникальнных объектов NFT возникла идея использовать кусочки qr-кода. Сам qr-код был разделен на 9 частей, как показано на рисунке 1, следовательно, в нашей сети было создано 9 уникальных токенов. Данные токены можно размещать на всех доступных платформах продажи

Рис. 1: Части QR-кода

NFT. Покупая их, владельцы будут иметь возможность получить доступ к ссылке группы криптошколы в социальной сети "Вконтакте при правильном сложении кода. Данная идея совершенствует игру-головоломку "Пазл превращая её в квест.

Механизм работы контракта

Были реализованы следующие функции:

1. **Minting**
   Функция создания нового токена.

2. **Transfer**
   Функция смены владельца контракта.

3. **tokenURI**
   Функция, которая возвращает уникальный адрес метаданных для конкретного токена в IPFS.

Используемые технологии

Смарт-контракт был написан на **Solidity** версии 0.8.9. Для разработки, развертывания и проверки смарт-контрактов была использована среда разработки Ethereum - **hardhat**.
Токены были реализованы по стандарту **ERC-721** - первый стандарт, определяющий создание невзаимозаменяемых цифровых активов [6]. Стандарт реализуется смарт-контрактами библиотеки **OpenZeppelin**, наследование от которых позволяет создавать совместимые смарт-контракты и упрощает создание и работу с NFT [7].
На сервисе **MetaMask** был создан кошелек с тестовым эфиром. В качестве тестовой сети для развертывания контракта была выбрана **Goerli**, так как с ней нет никаких сложностей в добавлении MetaMask кошелька и получении тестового эфира. Тестовые сети позволяют развертывать и тестировать смарт-контракты не тратя на операции реальные токены.
Так же была использована **Alchemy** - платформа для блокчейн-разработчиков, предоставляющая API, которая позволяет взаимодействовать с цепочкой Ethereum без необходимости запуска собственных узлов.
Изображения и информация о них (в виде JSON-файлов) были размещены на **IPFS** - децентрализованной сети обмена файлами [8]. IPFS ссылка для доступа к файлу непосредственно связана с его содержимым и включает криптографический хэш содержимого. Адрес файла невозможно произвольно переименовать, он может измениться только после изменения содержимого. Аналогично невозможно внести изменение в файл без изменения адреса (старый вариант останется на прежнем адресе, а новый будет доступен через другой адрес, так как хэш от содержимого файла изменится). Таким образом, гарантируется, что файлы не будут заменены после покупки/создания токена.
Для взаимодействия с функциями, прописанными в контракте, в том числе "чеканки"монет, был написан скрипт на **Python**.
Для пользовательского взаимодействия с контрактом был создан **Telegram**-бот, развернутый через webhook и туннель на **Ngrok**. Реализованы следующие функции:

1. Создание токена командой **/mintnft**, далее необходимо передать параметр - адрес токена;
2. Передача прав на контракт командой **/setnewowner**, далее необходимо передать параметр - адрес нового владельца.

Таким образом владелец может выпускать NFT с помощью бота.

ЛИТЕРАТУРА

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

[2] Tutorial "How to write and deploy an NFT"[Электронный ресурс] // URL: https://ethereum.org/en/developers/tutorials/how-to-write-and-deploy-an-nft/#make-api-key

[3] Standart ERC-721 [Электронный ресурс] // URL: https://eips.ethereum.org/EIPS/eip-721

[4] Документация библиотеки OpenZeppelin [Электронный ресурс] // URL: https://docs.openzeppelin.com/contracts/4.x/

[5] Использование IPFS для NFT [Электронный ресурс] // URL: https://docs.ipfs.io/how-to/mint-nfts-with-ipfs/#a-short-introduction-to-nfts

Кураторы исследования –

Е. Мельничук;

Д. Кондырев.

Реализация смарт-контракта лотереи на языке Solidity с подтверждением беспристрастности

А. И. Сацута
БФУ им. И. Канта
E-mail: tolya.satzuta@gmail.com

Аннотация

В рамках данного проекта был исследован принцип работы децентрализованных блокчейн систем, а также устройство смарт-контрактов и язык Solidity для их написания. В результате проекта был создан смарт-контракт лотерея и оракул, генерирующий номер выигрышного билета.

Ключевые слова: *DeFi, Decentralized Finance, Ethereum, Smart-Contract.*

На данный момент с ростом популярности децентрализованных финансов, дающих обладателям крипто-кошельков, в первую очередь, возможность распоряжаться своими активами без посредников, напрямую с другими пользователями, растёт спрос и на различные способы этого взаимодействия. В том числе на различные развлечения внутри системы блокчейн, примером которых могут являться лотереи.

Для любой лотереи в первую очередь важна случайность генерации выигрышных билетов. Из-за детерминированного характера блокчейна эта задача трудна, так как на результат лотереи может повлиять злоумышленник имеющий большие вычислительные мощности, которые дают ему возможность предсказать это случайное число, или майнер, имеющий возможность манипулировать данными в своих интересах. Для решения этой проблемы, а так же проблемы связи блокчейна с внешним миром, предлагается использовать оракулы, которые, однако, создают проблему централизации. Следовательно, в случае компрометации или сбоя используемого оракула смарт-контракт будет получать искажённые значения.

Определение 1. Оракулы — это сервисы, которые связывают блокчейны с внешним миром: децентрализованными приложениями, другими блокчейнами, торговыми площадками, облачными провайдерами, IoT-устройствами, платежными и корпоративными системами.

Оракулы берут на себя многие ключевые функции, одни из которых [5]:

1. Слушать — отслеживать систему блокчейн на предмет наличия входящих запросов на внешние данные от пользователей или смарт-контрактов.

2. Извлекать — собирать данные от одной или множества внешних систем, к примеру API с хостингом на внешних серверах.

3. Вычислять — производить безопасные оффчейн вычисления для смарт-контрактов, например вычисление среднего значения от множественных источников.

4. Транслировать — подписывать и транслировать транзакцию в блокчейне для отправки данных и любого сопроводительного подтверждения on-chain, для последующего использования смарт-контрактом.

Поставленные задачи для реализации проекта:

1. Познакомиться с внутренним устройством блокчейн-платформы Ethereum[13].

2. Разобрать механизмы работы смарт-контрактов и один из языков их реализации – Solidity.

3. Изучить возможности модуля Web3.py.

4. Реализовать смарт-контракт и оракул, определяющий победителя, протестировать их работоспособность.

Главная часть данного проекта – смарт-контракт, в котором были реализованы две основные функции:

- функция, дающая клиенту возможность купить билеты;
- функция, принимающая на вход номер выигрышного билета и реализующая выплату средств владельцу этого билета.

Для реализации смарт-контракта использовалась библиотека для языка Solidity – OpenZeppelin[6]. Из неё была импортирована базовая реализация контроля доступа, чтобы быть уверенным, что функция, отвечающая за определение адреса владельца выигрышного билета, доступна только владельцу контракта.

Вторая часть проекта – оракул, передающий смарт-контракту число, определяющее победителя. Он слушает блокчейн на предмет наличия вызываемого смарт-контрактом события. При его возникновении оракул генерирует случайное число и отправляет транзакцию с вызовом функции от этого числа, выбирающей победивший билет.

Однако, возникает такая проблема – для использования смарт-контракта недостаточно лишь генерации случайного числа. Пользователь должен быть уверен в честности лотереи, необходимо подтвердить, что оно было выбрано непредвзято.

Эта проблема была решена следующим образом: заранее генерируется случайное число от нуля до единицы и вычисляется его хеш (в данном случае с помощью SHA256). После этого программа отправляет транзакцию с данным хешем в открытый доступ, чтобы еще до покупки билетов пользователь мог его увидеть. Когда время, отведённое на проведение лотереи, закончится, оракул отследит событие о завершении лотереи и извлечёт из него общее число билетов, участвующих в игре. После этого он умножит количество билетов на сгенерированное случайное число, результатом этого произведения будет являться номер победившего билета. Кроме того, будет вычислено новое случайное число и его хеш. Таким образом, номер, случайное число и хеш нового случайного числа отправятся смарт-контракту и будут доступны всем пользователям. Посчитав хеш полученного случайного числа они смогут убедиться в честности его выбора, так как заранее переданный хеш будет совпадать с вычисленным хешем.

В итоге был реализован смарт-контракт лотереи с подтверждением беспристрастного выбора победителя. Беспристрастность гарантируется благодаря тому, что другой номер с таким же хешем можно получить только с помощью нахождения коллизии 1-го рода. Кроме того, нахождение данной коллизии не гарантирует, что нужный билет победит. Более того, номер выигрышного билета зависит также и от количества билетов в игре.

Дальнейшие планы по улучшению проекта:

- Повышение отказоустойчивости оракула.
- Оптимизация функций в смарт-контракте.

ЛИТЕРАТУРА

[1] https://blog.chain.link/what-is-the-blockchain-oracle-problem/.

[2] https://ethereum.org/en/developers/docs/.

[3] https://docs.openzeppelin.com/.

Кураторы исследования –
Кондырев Дмитрий Олегович;
Мельничук Евгений Михайлович;

Интеграция алгоритмов приватности в код смарт-контрактов

Е. И. Соколова[1], Е. С. Гричин[2]
[1]Томский государственный университет
[2]Новосибирский государственный университет
E-mail: alskv13@gmail.com, egorgrichin@gmail.com

Аннотация

В настоящее время блокчейн-технологии приобрели большую популярность. В связи с этим, происходит быстрое развитие платформ и инструментов для реализации данной технологии. Поэтому в данной работе проводилось исследование принципов и устройства платформы Ethereum; смарт-контрактов, как инструмента работы с Ethereum; и языка Solidity, позволяющего непосредственно создавать смарт-контракты. Также, для изучения решений проблемы конфиденциальности, рассматривался протокол доказательства с нулевым разглашением zk-SNARKs, инструмент и язык ZoKrates для реализации протокола. Отдельно была изучены библиотека ZoKrates и её устройство. Практическим результатом работы стала реализация функций на языке ZoKrates, генерация верификаторов и интеграция их в смарт контракты, а также разработка модели интеграции алгоритмов анонимности в практических целях (а именно, модель проведения выборов/голосования на основе блокчейна).

Ключевые слова: *блокчейн, децентрализованные системы, криптовалюта, Ethereum, смарт-контракты, Solidity, ZKP, доказательство с нулевым разглашением, zk-SNARKs, ZoKrates.*

Задачами нашего проекта являются:

1. изучение блокчейн-платформы Ethereum, её особенностей и недостатков;
2. сравнительный анализ Ethereum с другими платформами;
3. обзор смарт-контрактов и языка Solidity;
4. изучение общего принципа протоколов ZKP и криптографического протокола обеспечения анонимности zk-SNARK;
5. изучение инструментария ZoKrates;
6. написание функций ZoKrates, дополняющих примеры стандартных схем из библиотеки инструмента;
7. реализация смарт-контракта с верификацией с помощью написанной функции;
8. тестирование проекта;
9. теоритическая разработка модели использования алгоритмов приватности для дальнейшей работы.

Платформа Ethereum

В децентрализованных системах широко используется технология блокчейн: технология хранения информации в выстроенной по определенным правилам, непрерывной, связанной и последовательной цепочке блоков. Криптовалюта и платформа для создания онлайн-сервисов Ethereum реализует эту технологию.

Цель Ethereum состоит в том, чтобы создать альтернативный протокол для децентрализованных приложений, частично решающий проблемы более ранней криптовалюты Bitcoin. Ethereum, в отличии от предшественника, является блокчейном со встроенным полным по Тьюрингу языком программирования. Язык позволяет любому писать смарт-контракты и децентрализованные приложения, в которых можно создавать свои собственные правила для формата транзакций и функций перехода состояний.

Формально, смарт-контракты — это предварительно описанные определенные условия, только при выполнении которых может быть совершена транзакция. Протокол смарт-контрактов является одной из главных особенностей Ethereum, ведь по сравнению с Bitcoin, где отсутствует полнота по Тьюрингу и, соответственно, доступны для реализации лишь простые логические операции, смарт-контракты добавляют пользователям гибкость в написании алгоритмов и расширяют за счёт этого область использования Ethereum.

Доказательство с нулевым разглашением

Несмотря на ряд преимуществ перед Bitcoin, в ряде ситуаций Ethereum может показаться неподходящей технологией. Ведь в отличии от анонимных криптовалют, таких как Monero и Zcash, Ethereum не имеет встроенных технологий обеспечения конфиденциальности. И в некоторых сценариях необходимо доказать, что пользователь относится к некой группе, которой разрешено то или иное действие, транзакция, не раскрывая при этом определенной информации.

Группа протоколов для реализации такого доказательства называется Zero-Knowledge Proof Protocol (доказательство с нулевым разглашением). Цель доказательств с нулевым разглашением состоит в том, чтобы доказывающий смог убедить проверяющего, что утверждение верно, не раскрывая никакой информации, кроме достоверности самого утверждения.

Рассмотрим протокол zk-SNARK. Аббревиатура zk-SNARK расшифровывается как «Краткий неинтерактивный аргумент знания с нулевым разглашением» и относится к конструкции доказательства, при которой можно доказать владение определенной информацией, например, секретным ключом, без раскрытия этой информации и без какого-либо взаимодействия между доказывающим и проверяющим.

Схема zk-SNARK:

$Generator : G(C, \lambda) = (pk, vk)$
$Prover : P(input\ pk, input\ x, w) = prf$
$Verifier : V(vk, x, prf) == (\exists w : C(x, w))$

Обозначения:
C - программа
λ - секретный параметр
pk - сгенерированный доказательный ключ
vk - сгенерированный ключ проверки
x - публичное значение
w - секретное значение

Алгоритмы G, P, V определенны следующим образом:

Генератор ключей G принимает секретный параметр λ и программу C, и генерирует два публичных ключа: доказательный ключ pk, и ключ проверки vk. Эти ключи являются общедоступными параметрами, которые необходимо создать только один раз для этой программы C.

Доказывающий алгоритм P принимает на вход ключ pk, публичное значение x и секретное значение w. Алгоритм генерирует доказательство prf того, что доказывающий знает секретное значение w и что секретное значение удовлетворяет программе C.

Проверяющий алгоритм V вычисляет функцию $V(vk, x, prf)$ результатом которой будет $true$, если доказательство является верным, и $false$ в противном случае. Таким образом, эта функция возвращает $true$, если проверяющий знает секретное значение w, удовлетворяющее

$C(x, w) == true$.

В Ethereum zk-SNARK реализуется в предварительно скомпилированных контрактах. Алгоритм генерации ключей создает проверочный и доказательный ключи вне блокчейна, далее любой доказывающий может использовать сгенерированный ключ для создания доказательства. Далее, алгоритм проверки, обычно называемой верификацией, может выполняться внутри смарт-контракта, принимая на вход доказательство, ключ проверки и публичные значения. На выходе алгоритм верификации либо подтверждает, либо опровергает знание пользователя, что позволяет далее разрешать или запрещать те или иные действия исходя из логики основного смарт-контракта.

Инструмент ZoKrates

Вычисления в zk-SNARK являются весьма трудными, поэтому требуют оптимальной реализации, ведь Ethereum выполняет вычисления на всех узлах сети, что приводит к высоким затратам. Данную проблему позволяет решать ZoKrates – набор инструментов для автоматической генерации верификаторов zk-SNARK. ZoKrates реализует особый высокоуровневый язык, помогающий связывать с блокчейном Ethereum программы, которые так же с его помощью создаются вне сети. Таким образом, инструмент позволяет автоматизировать и упростить разработчику реализацию алгоритмов обеспечения анонимности, тем самым расширяя возможности использования Ethereum.

Стандартная схема использования описанного инструментария выглядит следующим образом:

1. С помощью файла с расширением *.zok, содержащего программу ZoKrates, генерируется смарт-контракт verifier.sol, реализующий zk-SNARK.
2. Функция verifyTX() вызывается основным смарт-контрактом из verifier.sol.
3. В зависимости от результата, в основном контракте разрешаются или запрещаются пользователю некоторые действия.

Результаты и дальнейшая работа

В результате выполнения проекта, была изучена технология блокчейн, особенности разных криптовалют, в частности Ethereum. Изучен язык написания смарт-контрактов Solidity, среда разработки IDE Remix, а также дополнительные инструменты для работы с криптовалютами, такие как: расширение для браузера Metamask, позволяющее контролировать кошельки и проводить транзакции, тестовые блокчейн-сети Ropsten и Goerli, библиотека web3, позволяющая осуществлять работу с блокчейном с помощью высокоуровневых языков программирования.

Много времени было уделено разбору zk-SNARK и инструмента ZoKrates. Разобрались с языком ZoKrates и библиотеками данного инструмента, научились реализовывать написание, сборку и компиляцию проектов.

Практическим итогом работы стала система смарт-контрактов, состоящая из тестового смарт-контракта *test.sol*, вызывающего верификатор и позволяющего выполнять транзакции только в случае возврата значения *true* из функции верификации. И непосредственно код верификаторов на ZoKrates и сгенерированные файлы *verifier.sol*. Наши файлы ZoKrates дополняют библиотеку готовых схем для языка и включают в себя:

1. Доказательство принадлежности заданному промежутку, т.е. $a \leq v \leq b$.
2. Доказательство включения дерева Меркла для заданной глубины дерева (определяется константой DEPTH) с использованием хеша sha256 из стандартной библиотеки ZoKrates.

Определение. Дерево Меркла - полное двоичное дерево, в листовые вершины которого помещены хеши от блоков данных, а внутренние вершины содержат хеши от сложения значений в дочерних

вершинах. Корневой узел дерева содержит хеш от всего набора данных. Дерево Меркла применяется для эффективного хранения транзакций в блокчейне. Оно позволяет получить «отпечаток» всех транзакций в блоке, а также эффективно верифицировать транзакции.

3. Комбинации математических операций.
4. Функции для работы с массивами.

В дальнейшей работе можно создать более сложные схемы на языке ZoKrates, реализовать дополнительные функции для исходного кода библиотеки. Заметим, что пользу практического применения алгоритмов приватности проще всего показать на реальной модели. Поэтому мы предлагаем создать модель, в которой можно наглядно продемонстрировать пользу протокола zk-SNARK. На данный момент нами была создана схема для проведения голосования на основе блокчейна, где необходимо сохранять тайну голосования. В будущем необходимо доработать наш алгоритм, проанализировать его на безопасность и эффективность, реализовать и интегрировать алгоритм доказательства с нулевым разглашением.

ЛИТЕРАТУРА

[1] Vitalik Buterin. Ethereum Whitepaper, 2022. https://ethereum.org/en/whitepaper/

[2] Vitalik Buterin. Ethereum development documentation, 2022. https://ethereum.org/en/developers/docs/

[3] Christian Lundkvist. Introduction to zk-SNARKs with examples, 2017. https://media.consensys.net/introduction-to-zksnarks-with-examples-3283b554fc3b

[4] Ariel Gabizon. Explaining SNARKs, 2017. https://electriccoin.co/blog/snark-explain/

[5] Edi Sinovcic. ZoKrates — zkSNARKs On Ethereum (made easy), 2019. https://medium.com/coinmonks/zokrates-zksnarks-on-ethereum-made-easy-8022300f8ba6

[6] Eric Hu. A Brief Dive Into zk-SNARKs and the ZoKrates Toolbox on the Ethereum Blockchain, 2019. https://medium.com/cornellblockchain/a-brief-dive-into-zk-snarks-and-the-zokrates-toolbox-on-the-ethereum-blockchain-cb7bd7f00fdc

[7] ZoKrates: source code. https://github.com/Zokrates/ZoKrates

Куратор исследования –
Дмитрий Олегович Кондырев - ассистент кафедры компьютерных систем ФИТ НГУ.

Генерация NFT в реальном времени с подтверждением непредвзятости

А. А. Чубань
Балтийский Федеральный Университет имени И. Канта
E-mail: ArtemChuban@yandex.ru

Аннотация

В данной работе разработан метод генерации NFT в реальном времени. Токен генерериуется до чеканки и добавляется в распределенное файловое хранилище. Для подтверждения непредвзятости во время генерации используется хеш токена с солью, которые видны всем пользователям сразу после генерации и до чеканки.

Ключевые слова: *Non-Fungible Token, NFT, Ethereum, Smart-Contract*

В настоящее время с ростом популярности понятия невзаимозаменяемого токена пропорционально растет и спрос на коллекции таких токенов, которые содержат в себе определенное количество заранее созданных или сгенерированных предметов, что приводит либо к невозможности масштабирования, либо к нечестному их расширению, так как возможность увеличения коллекции должна быть заложена в контракт сразу.

Это приводит к мысли о генерации новых токенов прямо во время их появления в блокчейне. В силу детерминированности блокчейна, полноценная генерация случайных чисел прямо внутри контракта невозможна, так как майнер может манипулировать некоторыми изменяемыми данными блока ради собственной выгоды. Следовательно, генерация токенов должна происходить вне блокчейна, а затем записываться при помощи транзакции от оракула.

Определение 1. Оракул - сервис, который связывает блокчейн с внешним миром.

Для отслеживания действий на контракте оракул должен просматривать все транзакции новых блоков, чтобы найти необходимые данные. Для облегчения данного поиска контракт может добавлять вызов событий в транзакцию. Оракулу необходимо будет просматривать лишь транзакции с данным видом события. В нашем случае нам интересно событие появления совершенно нового токена - *mint*.

Определение 2. Minting (чеканка) - процесс создания цифрового токена и последующая запись его в блокчейн.

Получив данное событие оракул должен сгенерировать новый уникальный токен, сохранить его в общедоступном месте и добавить ссылку в блокчейн.

Принцип случайного создания новых уникальных токенов состоит в пробной генерации токена до тех пор, пока он не будет уникален.

Для сохранения токенов лучшим вариантом является использование распределенной файловой системы с протоколом связи IPFS [1].

Определение 3. IPFS (InterPlanetary File System - межпланетная файловая система) - одноранговый гипермедийный протокол связи, индексирующий файлы по их хешу - CID, узлы которого формируют распределенную файловую систему.

Вдобавок случайность генерации должна быть подтверждена, иначе пользователи могут усомниться в честности распределения токена. Доказать непредвзятость ко всем пользователям можно следующим образом:

1. Оракул генерирует токен с солью до его чеканки
2. Сохраняет токен в IPFS
3. Размещает его хеш в общий доступ
4. При чеканке оракул добавляет CID токена в блокчейне

Если пользователь хочет проверить случайность генерации, то ему достаточно найти хеш от уже известного токена и сопоставить с тем, который был известен до чеканки. При их совпадении кто угодно мог получить данный токен, а значит фактор предвзятости полностью исключается. Соль токена необходима для предотвращения перебора хешей всех возможных токенов и, следовательно, предугадывании следующего токена.

Для добавления полученного CID в токен, оракул должен совершить транзакцию, вызвав функцию обновления информации о токене, доступ к которой есть только у оракула.

Дальнейшее улучшение метода вытекает из следующих проблем:

1. Метод генерации уникального токена нетривиален и текущая его реализация недетерминирована.
2. Место размещения хешей токенов для их проверки должно быть отказоустойчивым и подтвержденным, а значит хеши можно размещать внутри каждого блока в блокчейне.

Предложенный метод был реализован с помощью смарт-контракта, использующего в основе своей работы реализацию интерфейса IERC-721 из открытой библиотеки смарт-контрактов OpenZeppelin [2]. IERC-721 реализует функционал по получению, хранению и передаче токенов, а также возможность выбрать доверенное лицо, которое может совершать с токеном такие же действия, как и его владелец. Функционал был расширен возможностью прямой продажи токена от одного пользователя к другому и возможностью выставить токен на продажу любому желающему за указанную владельцем цену.

Сервис-оракул для реализации подтверждения непредвзятости генерации был написан на языке программирования Python с использованием модуля Flask для динамической генерации веб-страницы с хешами и ссылками на уже выпущенные токены и хешем будущего токена. Запущенная программа просматривает все транзакции в блокчейне Ethereum [3] на тестовой сети Goerli [4] в поисках события о выдаче нового токена. Каждый раз, когда токен передается от одного пользователя к другому, вызывается специальное событие, которое оповещает о трансфере. Событие создания нового токена является таким же событием передачи, однако в поле пользователя отправителя находится специальный (нулевой) адрес, который и обозначает чеканку нового токена. Получив данное событие с нулевым пользователем в графе отправителя, скрипт производит транзакцию вызова функции смарт-контракта, которая обновляет ссылку на токен, хеш которого соответствует хешу токена до его выпуска. Информация о токене сохраяntеся в формате JSON через API сервиса для работы с IPFS Pinata [5]. После этого случайным образом генерируется будущий токен, хеш которого добавляется на веб-страницу.

По итогу проекта был разработан смарт-контракт, реализующий продажу токенов между пользователями без использования торговых площадок, а также сервис-оракул генерирующий токены до их чеканки с предоставлением хеша будущего токена на веб-странице. Проект имеет большие перспективы, так как схожих смарт-контрактов в блокчейне Ethereum на данный момент небольшое количество и есть возможность занять данную нишу.

ЛИТЕРАТУРА

[1] https://docs.ipfs.io/

[2] https://docs.openzeppelin.com/

[3] https://ethereum.org/en/developers/docs/

[4] https://goerli.net/

[5] https://www.pinata.cloud/

Кураторы исследования –
Мельничук Евгений Михайлович
Кондырев Дмитрий Олегович

Распределенный сбор средств с использованием системы на основе технологии блокчейн

Ю. А. Шаманов
Балтийский Федеральный Университет имени И. Канта
E-mail: yura.shamanov2000@bk.ru

Аннотация

В работе был исследован механизм инвестирования денежных средств в некий проект используя посредника, хранящего инвестируемые средства. В случае, если денежные средства не были собраны в назначеный заранее срок, то они возращаются инвесторам, иначе средства перейдут организаторам проекта, а не третьим лицам. Был разработан смарт-контракт[1], для блокчейн-системы[2] Ethereum[3], позволяющий обходиться без третьего лица для хранения средств и реализующий автоматический возврат, если условия не выполнены.

Ключевые слова: *smart-contract (смарт-контракт), Ethereum, транзакция, блокчейн.*

Определение 1. Смарт-контракт (англ. smart contract) — компьютерный алгоритм, предназначенный для формирования, управления и предоставления информации о владении чем-либо. Чаще всего речь идёт о применении технологии блокчейна. В более узком смысле под смарт-контрактом понимается набор функций и данных (текущее состояние), находящихся по определённому адресу в блокчейне.

В наше время спектр действий, производимых с деньгами, довольно широкий: деньги можно одалживать, депонировать, инвестировать, давать в рост и так далее. Смарт-контракты в Ethereum [6] позволяют все это реализовать и дают базу для новой экономики, свободной от государств и банков.

Цель Ethereum'а — объединить и усовершенствовать концепции криптовалют, создать протокол, который позволит разработчикам спроектировать распределенные приложения на основе инфраструктуры, предоставляющей масштабируемость, полноту по Тьюрингу и простоту разработки.

Определение 2. Децентрализованные финансовые сервисы, или децентрализованные финансы (англ. Decentralized Finance, DeFi) — общее название для аналогов традиционных финансовых инструментов, реализованных в децентрализованной архитектуре.

Эти сервисы общедоступны, являются проектами с открытым исходным кодом и чаще всего основаны на смарт-контрактах. Основная задумка DeFi – создание независимой и прозрачной финансовой экосистемы, которая не подвержена влиянию регуляторов и человеческого фактора. Проще говоря, с помощью DeFi финансы становятся доступны кому угодно: пользователи проводят транзакции и решают финансовые вопросы напрямую друг с другом, а не через посредников в лице банков, судов брокерских организаций и пр.

Одна из областей применения смарт-контрактов – системы, в которых важно быть уверенным в том, куда переводятся денежные средства и кому достаются инвестиции.

Определение 3. Kickstarter[4] — сайт для привлечения денежных средств на реализацию творческих, научных и производственных проектов по схеме добровольных пожертвований.

Представим ситуацию, что компания собирает по $10 на Kickstarter на крупное исследование.
Процесс выглядит следующим образом:

1. участники переводят по $10,

2. деньги блокируются на платформе,

3. если удалось собрать 100% от заданной суммы, то Kickstarter переводит деньги создателям проекта,

4. если за изначально определенный срок собрать деньги не удалось, то деньги отправляются обратно участникам.

Третьей стороной в этом случае выступает краудфаундинговая платформа Kickstarter и мы должны ей доверять, что отправленные $10 либо передадут организаторам иследования, либо вернут нам обратно.

Этот договор можно реализовать через смарт-контракт.

Разработчик пишет программу на специальном языке Solidity[5][6][7][8] – смарт-контракт, где прописаны условия. Когда они будут выполнены, совершится транзакция: деньги либо уйдут компании, либо вернутся контрибьюторам.

Такой механизм оказывается надежнее:

1. деньги не передаются третьей стороне — они просто блокируются в блокчейне,

2. любой участник сможет просмотреть программный код и убедиться, что он работает именно так, как заявлено в условии.

В ходе работы над проектом был предложен и разработан данный механизм с использованием смарт-контрактов на языке Solidity, работающих с сетью блокчена Ethereum. С помощью библиотеки OpenZeppelin [7] и дугих инструментов, удалось сделать смарт-контракт безопасным для переводов и хранения средств на счету смарт-контракта, а также оптимизировать его работу. Был создан набор тестов и проведена проверка работоспособности смарт-контракта с использованием среды разработки HardHat[9]. Успешно разработан и реализован графический интерфейс. По ходу разработки возникали сложности с безопасностью переводов денежных средств и обхода проверки роли. Получен опыт создания смарт-контрактов на языке Solidity и обеспечения безопасности переводов вутри сети блокчейна Ethereum.

ЛИТЕРАТУРА

[1] https://en.wikipedia.org/wiki/Smart_contract

[2] https://www.oracle.com/cis/blockchain/what-is-blockchain/

[3] https://ethereum.org/en/

[4] https://www.kickstarter.com/

[5] https://habr.com/ru/hub/solidity/

[6] https://ethereum.org/en/developers/docs/

[7] https://docs.openzeppelin.com/

[8] https://docs.soliditylang.org/en/latest/

[9] https://hardhat.org/getting-started

[10] https://www.investopedia.com/terms/p/proof-work.asp

[11] https://en.wikipedia.org/wiki/Proof_of_work

[12] https://www.investopedia.com/terms/c/crypto-token.asp

[13] https://aws.amazon.com/ru/what-is/blockchain/

Кураторы исследования:
Мельничук Евгений Михайлович
Конадырев Дмитрий Олегович

Список участников

1. **ECHIMAM Chineze** – School of Radio Engineering and Computer Studies (FRCT), Department of Intelligent Information Systems and Technologies, Moscow Institute of Physics and Technology(National Research University)

2. **АБРАМОВА Наталья Александровна** – Информационная безопасность, Сибирский государственный университет науки и технологии имени академика М.Ф. Решетнева

3. **АТУТОВА Наталья Дмитриевна** – Механико-математический факультет, Новосибирский государственный университет

4. **БАРХАТКИН Роман Игоревич** – Механико-математический факультет, Новосибирский государственный университет

5. **БАХАРЕВ Александр Олегович** – Механико-математический факультет, Новосибирский государственный университет

6. **БЕЛЯНИН Николай Сергеевич** – Кафедра безопасности информационных технологий, Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

7. **БОБРЫШЕВ Святослав Александрович** – Кафедра безопасности информационных технологий, Институт компьютерных технологий и информационной безопасности, Южный федеральный университет

8. **БОНИЧ Татьяна Андреевна** – Факультет информационных технологий, Новосибирский государственный университет

9. **БОЯЗИТОВ Вадим Дмитриевич** – Компьютерная безопасность, Томский государственный университет

10. **БОЯНДИН Лев Константинович** – Специализированный учебно-научный центр Новосибирского государственного университета

11. **БЫКОВ Денис Александрович** – Механико-математический факультет, Новосибирский государственный университет

12. **ВОРОБЬЁВ Денис Кириллович** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

13. **ГРИЧИН Егор Сергеевич** – Новосибирский государственный технический университет

14. **ДОРОНИН Артемий Евгеньевич** – Факультет информационных технологий, Новосибирский государственный университет

15. **ЕРОХИНА Анна Александровна** – Физический факультет, Московский государственный университет им. М.В. Ломоносова

16. **ЖУКОВ Георгий Алексеевич** – Механико-математический факультет, Новосибирский государственный университет

17. **ЗЮБИНА Дарья Александровна** – Факультет информационных технологий, Новосибирский государственный университет

18. **ИДРИСОВА Валерия Александровна** – Институт математики им. С.Л. Соболева СО РАН

19. **ИСАКОВ Игорь Владимирович** – Факультет автоматики и вычислительной техники, Новосибирский государственный технический университет

20. **ИЩУКОВА Евгения Александровна** – Кафедра безопасности информационных технологий, Южный федеральный университет

21. **КАЛГИН Константин Викторович** – Факультет информационных технологий, Новосибирский государственный университет

22. **КОЛОМЕЕЦ Николай Александрович** – Кафедра теоретической кибернетики, Институт математики им. С.Л. Соболева СО РАН

23. **КОНДЫРЕВ Дмитрий Олегович** – Факультет информационных технологий, Новосибирский государственный университет

24. **КУНИНЕЦ Артем Андреевич** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

25. **КУЦЕНКО Александр Владимирович** – Механико-математический факультет, Новосибирский государственный университет

26. **ЛЕВИН Владимир Александрович** – Механико-математический факультет, Новосибирский государственный университет

27. **МАКСИМЛЮК Юлия Павловна** – Механико-математический факультет, Новосибирский государственный университет

28. **МАЛЫГИНА Екатерина Сергеевна** – Балтийский федеральный университет им. И. Канта

29. **МАСЛОВ Роман Алексеевич** – Факультет автоматики и вычислительной техники, Новосибирский государственный технический университет

30. **МЕЛЬНИЧУК Евгений Михайлович** – Балтийский федеральный университет им. И. Канта

31. **МОКРОУСОВ Антон Сергеевич** – Факультет информационных технологий, Новосибирский государственный университет

32. **ОДУД Илья Михайлович** – Физический факультет, Новосибирский государственный университет

33. **ПАНФЕРОВ Матвей Андреевич** – Институт математики им. С.Л. Соболева СО РАН

34. **ПАРФЕНОВ Денис Романович** – Факультет информационных технологий, Новосибирский государственный университет

35. **ПИВНЕВА Эвелина Андреевна** – Механико-математический факультет, Новосибирский государственный университет

36. **САЦУТА Анатолий Игоревич** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

37. **СЕРГЕЕВА Юлия Алексеевна** – Гуманитарный институт, Новосибирский государственный университет

38. **СКУДИНА Виктория Викторовна** – Механико-математический факультет, Новосибирский государственный университет

39. **СОКОЛОВА Елена Игоревна** – Институт прикладной математики и компьютерных наук, Томский государственный университет

40. **СУТОРМИН Иван Александрович** – Механико-математический факультет, Новосибирский государственный университет

41. **ТОКАРЕВА Наталья Николаевна** – Кафедра теоретической кибернетики, Институт математики им. С.Л.Соболева СО РАН

42. **ФРАНЦУЗОВА Анастасия Павловна** – Гуманитарный институт, Новосибирский государственный университет

43. **ХИЛЬЧУК Ирина Сергеевна** – Механико-математический факультет, Новосибирский государственный университет

44. **ЧЕРКАШИН Виталий Юрьевич** – Механико-математический факультет, Новосибирский государственный университет

45. **ЧУБАНЬ Артем Андреевич** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

46. **ШАМАНОВ Юрий Альбертович** – Институт физико-математических наук и информационных технологий, Балтийский федеральный университет им. И. Канта

47. **ШАПОРЕНКО Александр Сергеевич** – Механико-математический факультет, Новосибирский государственный университет

48. **ШАШКИНА Елизавета Анатольевна** – Специализированный учебно-научный центр Новосибирского государственного университета