МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Лелюк Евгений Андреевич

СИНТЕЗ ПОСТКВАНТОВОЙ СХЕМЫ ИНКАПСУЛЯЦИИ СЕАНСОВОГО КЛЮЧА

Специальность 2.3.6 – «Методы и системы защиты информации, информационная безопасность»

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель:
кандидат технических наук,
Косолапов Юрий Владимирович

Ростов-на-Дону – 2026

Содержание

Стр.

Стр.

Введение

Актуальность темы исследования. Стойкость применяемых в настоящее время на практике асимметричных криптосистем основана на сложности задач факторизации целых чисел или дискретного логарифмирования в конечной группе. Однако показано [1], что эти задачи могут быть решены за полиномиальное время на квантовом компьютере. Актуальной задачей криптографии в настоящее время является разработка криптосистем, стойких к атакам с использованием квантовых вычислений. Об этом свидетельствует проведение таких конкурсов, как NIST PQC (США), KpqC Competition (Южная Корея), а также работа рабочей группы ТК26 (Российская Федерация) по синтезу новой схемы инкапсуляции сеансового ключа, которая была бы стойкой против нарушителя, имеющего доступ к квантовому компьютеру достаточной мощности. Криптографические системы, в основе которых лежит применение помехоустойчивых кодов (далее – кодовые криптосистемы), рассматриваются как одна из альтернатив используемым в настоящее время асимметричным криптографическим системам [2].

Степень разработанности темы. Первой кодовой считается криптосистема, предложенная Робертом Мак–Элисом в 1978 году [3], в основе которой лежит использование кодов Гоппы. Для удобства далее эта система называется Original McEliece. На этой системе основан протокол инкапсуляции сеансового ключа NTS-KEM [4], участвовавший в конкурсе NIST PQC. Впоследствии этот проект был объединен с проектом Classic McEliece [5], входящим в число финалистов NIST PQC. Система Classic McEliece также основана на кодах Гоппы. Ее отличие от Original McEliece в том, что в Classic McEliece используется схема шифрования Нидеррайтера [6]. Стойкость обеих систем, в частности, основана на сложности задачи декодирования случайного кода. Для этой задачи на текущий момент не найдено эффективного решения в модели квантовых вычислений.

Системы Original McEliece и Classic McEliece обладают своими достоинствами и недостатками. Первая, например, позволяет без дополнительных преобразований реализовать рандомизированное шифрование [7], а вторая обладает меньшим размером открытого ключа при сопоставимой стойкости. Тем не менее размер открытого ключа является недостатком систем на кодах Гоппы. Попытки использовать коды Рида–Соломона [6], коды Рида–Маллера [8], алгебро–геометрические коды [9], коды с низкой плотностью проверок на четность [10] для уменьшения ключа не увенчались успехом, поскольку были найдены эффективные атаки на ключ (структурные атаки) для соответствующих криптосистем [11-15]. Отметим, что коды Гоппы принадлежат классу альтернантных кодов. На текущий момент для некоторых классов кодов Гоппы также найдены структурные атаки на соответствующие криптосистемы [16; 17]. Кроме того, была найдена эффективная атака для одного класса подпространственных подкодов кодов Рида–Соломона, которые также являются альтернантными [18]. Эти результаты не исключают появления в будущем эффективных структурных атак на кодовые криптосистемы и на других классах кодов Гоппы. Поэтому, несмотря на имеющиеся стойкие схемы, актуальна задача поиска других эффективно декодируемых помехоустойчивых кодов, обеспечивающих высокую стойкость кодовых криптосистем типа Мак–Элиса при небольшом размере открытого ключа.

Одним из способов получения новых кодов является применение кодовых конструкций. Однако отметим, что применение таких кодовых конструкций на основе известных кодов (базовых кодов), как соединение кодов [19], прямая сумма кодов, переход от расширений полей к базовым полям [20], также не позволили повысить стойкость [21; 22]. Тем не менее кодовые конструкции являются перспективными, поскольку позволяют на основе известных кодов строить новые эффективно декодируемые коды. Важным примером кодовой конструкции является тензорное произведение кодов, так как она находит широкое применение в системах защиты данных от помех [23-25]. В общем случае новые коды принадлежат классу, отличному от класса базовых кодов, т.е. имеют иную

структуру (алгебраическую и/или комбинаторную), поэтому структурные атаки на криптосистемы на основе базовых кодов неприменимы непосредственным образом к криптосистемам на новых кодах. Отметим, что многие структурные атаки основаны на использовании произведения Шура–Адамара [13; 21; 22], поэтому важной считается оценка стойкости криптосистем на основе новых конструкций к такого рода атакам [26].

Целью диссертационного исследования является повышение диверсификации эффективных постквантовых схем инкапсуляции сеансового ключа.

Научная задача, решение которой содержится в работе – разработка, теоретическое обоснование и исследование постквантовых кодовых схем шифрования, основанных на кодовых конструкциях, которые позволяют реализовать механизм инкапсуляции сеансового ключа, стойкий к атакам с использованием квантового компьютера.

Для достижения поставленной цели в диссертации решаются следующие **задачи**:

1. Исследовать основанный на замене помехоустойчивого кода в схеме Мак–Элиса способ построения асимметричных кодовых криптосистем для инкапсуляции сеансового ключа, способы анализа стойкости этих систем к атакам на ключ и шифрограмму, а также свойства и характеристики кодов, влияющие на эту стойкость.
2. Исследовать способы построения новых кодов, основанные на комбинировании известных кодов. Выбрать перспективную для использования в схеме Мак–Элиса кодовую конструкцию. Описать криптографические свойства и характеристики выбранных кодов, в том числе свойства произведения Шура–Адамара.
3. Разработать алгоритмы гарантированного и вероятностного декодирования выбранных кодов с целью применения в схеме Мак–Элиса.
4. Разработать асимметричную криптосистему типа Мак–Элиса на выбранных кодах и провести анализ стойкости этой системы к атакам

на ключ и шифрограмму. Описать параметры построенной криптосистемы, обеспечивающие ее эффективность.

Объект исследования. Объектом исследования являются постквантовые схемы инкапсуляции сеансового ключа для защиты конфиденциальности данных.

Предмет исследования. Предметом исследования являются методы построения эффективных кодовых криптосистем типа Мак–Элиса.

Методология и методы исследования. В диссертационной работе для теоретического исследования были использованы методы дискретной математики, теории кодирования, теории вероятностей, комбинаторного анализа, теории графов, анализа стойкости криптографических схем. Для экспериментального исследования использовались методы процедурного и объектно–ориентированного программирования.

Основные положения, выносимые на защиту:

1. Алгоритм шифрования и основанные на гарантированном и вероятностном декодировании алгоритмы расшифрования криптосистемы типа Мак–Элиса на конструкции D-кодов. Эти алгоритмы позволяют применять построенную криптосистему для решения задачи организации защищенного канала передачи данных с использованием схемы инкапсуляции сеансового ключа.
2. Алгоритм определения подмножеств сильных и слабых ключей криптосистемы Мак–Элиса на конструкции D-кодов на основе кодов Рида–Маллера, основанный на свойствах произведения Шура–Адамара этих кодов.
3. Алгоритм комбинированной атаки для слабых ключей криптосистемы типа Мак–Элиса на основе D-кодов, использующий структурную атаку для атаки на шифрограмму. Построенный алгоритм позволяет повысить качество анализа комплексной стойкости кодовых криптосистем.
4. Параметры эффективных стойких криптосистем типа Мак–Элиса на D-кодах для применения в схеме инкапсуляции сеансового ключа.

Найденные параметры, в частности, позволяют использовать предложенную схему в прикладных задачах, обеспечивая сопоставимые стойкость и размер открытого ключа с аналогичными схемами на кодах Гоппы в случае эфемерного использования сеансовых ключей.

Научная новизна диссертационной работы заключается в следующем:

1. Разработаны и программно реализованы алгоритмы шифрования и расшифрования криптосистемы типа Мак–Элиса на основе D-кодов. В частности, разработаны и реализованы алгоритмическая модель декодирования с гарантированным исправлением ошибок для D-кодов, **отличающаяся** применением мажоритарного подхода к декодированию, и алгоритмы вероятностного декодирования D-кодов на основе кодов Рида–Маллера, **отличающиеся** декодированием ошибок в количестве, превышающем половину кодового расстояния, и позволяющие за счет этого сократить размер открытого ключа. Построенные алгоритмы обеспечивают эффективное расшифрование в криптосистеме типа Мак–Элиса.
2. Разработан и программно реализован алгоритм определения множества сильных и слабых ключей криптосистемы на D-кодах на основе кодов Рида–Маллера, **отличающийся** использованием найденных криптографических свойств разложимости степеней Шура–Адамара D-кодов на основе кодов Рида–Маллера в прямую сумму кодов Рида–Маллера, и позволяющий эффективно находить параметры стойких систем на D-кодах.
3. Разработан алгоритм комбинированной атаки для слабых ключей криптосистемы типа Мак–Элиса на основе D-кодов, **отличающийся** применением структурной атаки с частичным восстановлением секретного ключа для увеличения вероятности успеха атаки на шифрограмму. Теоретически показано и экспериментально подтверждено, что разработанный алгоритм позволяет для слабых ключей криптосистемы

значительно упростить атаку на шифрограмму относительно классической атаки декодированием по информационным совокупностям.

4. На основе разработанных подходов к декодированию D-кодов на кодах Рида–Маллера, исследованных криптографических свойств этих кодов и результатов анализа стойкости построена новая криптосистема типа Мак–Элиса, **отличающаяся** применением конструкции D-кодов.

Теоретическая значимость. Теоретические результаты, полученные в данном исследовании, в частности, свойства произведения Шура–Адамара D-кодов и результаты анализа стойкости криптосистемы на этих кодах могут использоваться как при дальнейшем изучении криптосистем на D-кодах, например, для уточнения множеств сильных и слабых ключей построенной криптосистемы, так и при разработке новых кодовых криптосистем на основе подкодов прямой суммы кодов.

Практическая ценность. Построенная асимметричная кодовая криптосистема типа Мак–Элиса на D-кодах на основе кодов Рида–Маллера обладает либо большей стойкостью при сопоставимом размере ключа, либо меньшим размером ключа при сопоставимой стойкости, либо большей стойкостью при меньшем размере ключа по сравнению с системой Original McEliece. Это позволяет применять предложенную систему в схемах обеспечения конфиденциальности данных, например, для инкапсуляции сеансового ключа симметричной криптосистемы или для повышения защищенности данных, циркулирующих в информационных системах, за счет использования рандомизированного шифрования. Разработанный теоретико–графовый подход к декодированию D-кодов и, в частности, тензорного произведения кодов может применяться в задаче защиты от помех в каналах связи.

Соответствие диссертации паспорту научной специальности. Диссертация соответствует паспорту научной специальности 2.3.6. – «Методы и системы защиты информации, информационная безопасность» и охватывает следующие области исследования, входящие в эту специальность: «Исследования в области безопасности криптографических алгоритмов, криптографи-

ческих примитивов, криптографических протоколов. Защита инфраструктуры обеспечения применения криптографических методов» (п. 19).

Достоверность полученных результатов подтверждается корректностью математических выкладок и доказательств теорем, а также экспериментальными исследованиями.

Внедрение результатов работы. Результаты диссертационного исследования, подтвержденные соответствующими актами, используются в:

1. Деятельности Автономной Некоммерческой Организации «Национальный Технологический Центр Цифровой Криптографии» (г. Москва) при выполнении научно-исследовательской работы «Формирование методики и автоматизированных инструментов выбора постквантовых механизмов, основанных на помехоустойчивом кодировании используемых при обеспечении информационной безопасности сетевого взаимодействия», шифр «Кульминация».
2. Деятельности ФГАНУ "Научно-исследовательский институт "Специализированные вычислительные устройства защиты и автоматика" (г. Ростов-на-Дону) при решении задачи выработки общего сеансового ключа для организации защищенного канала передачи информации.
3. Учебно-исследовательском процессе на кафедре алгебры и дискретной математики Института математики, механики и компьютерных наук им. И.И. Воровича ЮФУ.

Апробация работы. Основные результаты работы были представлены на следующих конференциях и научных семинарах: семинар «Математические методы защиты информации» института математики, механики и компьютерных наук им. И. И. Воровича, Ростов-на-Дону, 2017 г.; III Всероссийский научный форум «Наука будущего – наука молодых», Нижний Новгород, 2017 г.; Научная конференция «Современные информационные технологии: тенденции и перспективы развития» (СИТО), Ростов-на-Дону, 2018, 2019 гг.; XX Международная конференция «Сибирская научная школа–семинар "Компьютерная безопасность и криптография" имени Геннадия Петровича

Агибалова» (SibeCRYPT), Новосибирск, 2021 г.; XII Международный симпозиум «Современные тенденции в криптографии» (CTCrypt), Волгоград, 2023 г.; Всероссийский научный семинар «Кибербезопасность: теория и практика», НИЯУ МИФИ, Москва, 2024 г.

Публикации. Основные положения диссертации опубликованы в 9 научных печатных работах, в том числе: 3 – в ведущих рецензируемых научных журналах, входящих в перечень ВАК (категории К1, RSCI), 2 – в научных рецензируемых журналах, индексируемых в базе Scopus (Q3, что соответствует категории ВАК К1), 4 – в материалах конференций и других изданиях. Получено свидетельство о государственной регистрации программы для ЭВМ.

Личный вклад заключается в выполнении основного объема теоретических и экспериментальных исследований, изложенных в диссертационной работе, включая исследование предметной области, формулировку и доказательство лемм и теорем, разработку методов и алгоритмов, входящих в число основных результатов работы, разработку методов и программных систем для проведения экспериментальных исследований, проведение экспериментов, анализ результатов теоретического и экспериментального исследования, оформление результатов в виде публикаций и научных докладов. Все выносимые на защиту результаты получены автором лично.

Объем и структура работы. Диссертация написана на русском языке, состоит из введения, трех глав, заключения, списка используемой литературы из 90 наименований и двух приложений. Полный объем диссертации составляет 159 страниц (в том числе приложений 7 стр.), включая 10 рисунков и 14 таблиц.

Во введении обосновывается актуальность исследований, проводимых в рамках данной диссертационной работы, приводится обзор научной литературы по изучаемой проблеме, формулируется цель, ставятся задачи работы, излагается научная новизна и практическая значимость представляемой работы, декларируются положения, выносимые на защиту, область исследования

и апробация полученных результатов, а также приведено краткое содержание каждой из глав.

В первой главе рассматриваются вопросы использования асимметричных криптосистем в протоколе инкапсуляции сеансового ключа, а также вопросы построения новых кодовых криптосистем для таких протоколов. Приводятся основные понятия теории кодирования, используемые в кодовой криптографии. Также в рамках задачи построения эффективных декодеров для применения в кодовой криптосистеме рассматривается концепция мажоритарного декодирования, на основе которой в следующей главе строится один из декодеров D-кодов: с гарантированным исправлением ошибок.

Вторая глава посвящена исследованию D-кодов. Именно, приводится определение D-кодов и исследуются их криптографические свойства. В частности, находятся условия разложимости степеней Шура–Адамара D-кодов на основе кодов Рида–Маллера в прямую сумму неразложимых кодов. Для эффективного расшифрования в главе строятся алгоритмические модели гарантированного и вероятностных декодеров для D-кодов. В качестве гарантированного строится мажоритарный декодер, а в качестве вероятностных – декодеры, использующие блочную структуру кодового слова D-кода. Для вероятностных декодеров приводятся результаты экспериментов, демонстрирующих их эффективность.

Третья глава посвящена синтезу схемы инкапсуляции сеансового ключа (KEM) на основе системы типа Мак–Элиса на D-кодах на основе кодов Рида–Маллера, а также исследованию ее стойкости. Именно, выделяются множества сильных и слабых ключей криптосистемы Мак–Элиса на D-кодах на основе кодов Рида–Маллера. Для слабых ключей криптосистемы строится комбинированная атака, позволяющая с помощью структурной атаки значительно повысить эффективность атаки на шифрограмму, и приводится оценка ее эффективности. Для сильных ключей криптосистемы подбираются параметры для возможности практического применения и проводится сравнение с ори-

гинальной системой на кодах Гоппы при использовании разных подходов к декодированию D-кодов.

В заключении изложен основной научный результат диссертации, а также сформулированы теоретические и практические результаты, полученные в результате диссертационной работы.

Глава 1. Схема инкапсуляции сеансового ключа на кодовой криптосистеме

1.1 Схема асимметричного шифрования и механизм инкапсуляции сеансового ключа

При описании криптографических схем будем придерживаться следующих обозначений: случайный и равновероятный выбор элемента a из множества A будем обозначать $a \leftarrow A$, генерацию элемента a с помощью алгоритма Alg – соответственно $a \leftarrow \mathsf{Alg}$, а обращение к алгоритму Alg, не принимающему параметры, будем обозначать $\mathsf{Alg}(\cdot)$.

Схема двухключевой асимметричной криптосистемы шифрования (PKE, public key encryption) для заданного множества сообщений $\mathcal{M}$ может быть описана тройкой $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$, где

1. $\mathsf{Gen}^{\mathsf{PKE}}$ – полиномиальный вероятностный алгоритм генерации ключей pk и sk $(pk, sk) \leftarrow \mathsf{Gen}^{\mathsf{PKE}}(\cdot)$: секретного sk и соответствующего ему публичного pk;
2. Enc – полиномиальный вероятностный алгоритм шифрования, преобразующий произвольное сообщение $\mathbf{m} \in \mathcal{M}$ в шифртекст $\mathbf{c}$: $\mathbf{c} \leftarrow \mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$; здесь $\mathbf{r}$ – значение случайной величины, которое может вырабатываться при шифровании;
3. Dec – полиномиальный алгоритм расшифрования, такой, что $\mathsf{Dec}(sk, \mathbf{c})$ возвращает $\mathbf{m}$, если $\mathbf{c}$ представляет шифртекст, для которого известен полиномиальный алгоритм нахождения $\mathbf{m}$ с помощью sk; в противном случае $\mathsf{Dec}(sk, \mathbf{c})$ возвращает сообщение об ошибке $\perp \notin \mathcal{M}$.

Под OW–стойкостью (**O**ne–**W**ay–стойкостью) схемы PKE обычно понимают стойкость к атакам дешифрования шифртекста, полученного при шифровании случайно и равновероятно выбранного открытого текста. При

этом уровень стойкости $\lambda_{\mathsf{OW}} \in \mathbb{N}$ может быть определен как отрицательный двоичный логарифм вероятности успеха наилучшей из таких атак.

Асимметричные шифросистемы обычно не используются для защиты непосредственно пользовательских данных, но при этом являются основой многих криптографических протоколов, в частности, механизмов инкапсуляции сеансового ключа (KEM, **K**ey **E**ncapsulation **M**echanism). Одной из возможных целей протокола KEM является передача от отправителя к получателю сеансового ключа $\mathbf{K}(\mathbf{K} \leftarrow \{0,1\}^m)$ симметричного шифра (где обычно $m \in \{128,192,256\}$), на котором выполняется шифрование пользовательских данных.

Рассмотрим задачу передачи зашифрованного сообщения с помощью механизма инкапсуляции сеансового ключа. Соответствующая схема представлена на рисунке 1.1. В соответствии с этой схемой, отправитель, с помощью открыто-

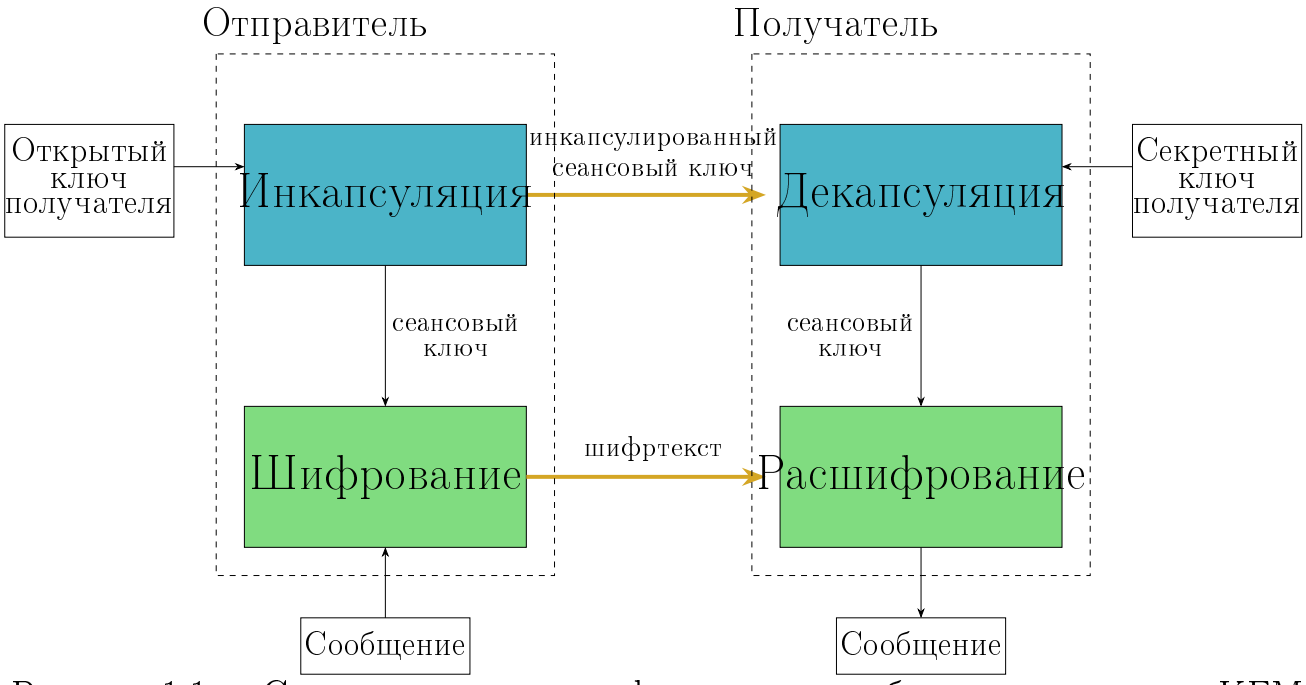

Рисунок 1.1 — Схема передачи зашифрованного сообщения с помощью KEM

го ключа получателя, производит инкапсуляцию общего сеансового ключа $\mathbf{K}$ и передает его получателю. Также отправитель производит шифрование информационного сообщения с помощью общего сеансового ключа. Тогда получатель производит декапсуляцию сеансового ключа и с помощью этого ключа выполняет расшифрование информационного сообщения. В этой схеме шифрование

и расшифрование производится с помощью некоторой симметричной системы, в то время как инкапсуляция и декапсуляция выполняются как раз с помощью асимметричной системы.

Механизм инкапсуляции для фиксированного пространства сеансовых ключей $\mathcal{K}$ и схемы $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ может быть описан тройкой алгоритмов $\mathsf{KEM} = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$, где

1. $\mathsf{Gen}^{\mathsf{KEM}}$ – полиномиальный вероятностный алгоритм генерации публичного pk' и секретного sk' ключей (строится на основе $\mathsf{Gen}^{\mathsf{PKE}}$);
2. Encaps – полиномиальный вероятностный алгоритм получения пары $(\mathbf{K},\mathbf{c})$: $(\mathbf{K},\mathbf{c}) \leftarrow \mathsf{Encaps}(pk')$, где $\mathbf{c}$ называется инкапсуляцией сеансового ключа $\mathbf{K} \in \mathcal{K}$ (строится на основе Enc);
3. Decaps – полиномиальный алгоритм извлечения (декапсуляции) сеансового ключа $\mathbf{K}$ из $\mathbf{c}$ (строится на основе Dec). Алгоритм $\mathsf{Decaps}(pk', sk', \mathbf{c})$ возвращает $\mathbf{K}$, если $\mathbf{c}$ представляет корректную инкапсуляцию ключа $\mathbf{K}$, в противном случае $\mathsf{Decaps}(pk', sk', \mathbf{c})$ может возвращать сообщение об ошибке $\perp \notin \mathcal{K}$ (явный отказ, explicit rejection), либо вектор, отличный от $\mathbf{K}$, но обычно зависящий от $\mathbf{c}$ и sk' (неявный отказ, implicit rejection).

Стойкость $\lambda_{\mathsf{KEM}} \in \mathbb{N}$ механизма KEM может быть определена как отрицательный двоичный логарифм вероятности успеха наилучшей известной атаки на KEM. Обычно для механизмов KEM, основанных на схемах PKE, выполняется неравенство:

$$\lambda_{\mathsf{OW}} \geqslant \lambda_{\mathsf{KEM}}. \tag{1.1}$$

Механизмы KEM условно можно разделить на механизмы с *многоразовым* использованием пары $(pk',sk') \leftarrow \mathsf{Gen}^{\mathsf{KEM}}$ и механизмы с *одноразовым или эфемерным* использованием. Вероятность

$$\delta = \Pr\left(\perp \leftarrow \mathsf{Decaps}(pk', sk', \mathbf{c}) : (pk',sk') \leftarrow \mathsf{Gen}^{\mathsf{KEM}}, (\mathbf{K},\mathbf{c}) \leftarrow \mathsf{Encaps}(pk')\right) \tag{1.2}$$

события, заключающегося в невозможности декапсуляции сеансового ключа, может приводить к различным последствиям для механизмов с одноразовым и многоразовым использованием. Вероятность δ может влиять, как на удобство использования механизма, так и на его стойкость. Неудобства, связанные с ненулевым δ, заключаются в следующем. При многоразовом использовании в случае ошибки декапсуляции отправителю сессионного ключа необходимо повторно использовать алгоритм Encaps, а получателю – алгоритм Decaps; в случае одноразового использования KEM получателю, помимо этого, необходимо генерировать новую пару $(pk',sk') \leftarrow \mathsf{Gen}^{\mathsf{KEM}}$ и передавать pk' отправителю.

Наиболее значительные различия механизмов инкапсуляции с многоразовым и одноразовым использованием проявляются во влиянии значения δ на стойкость этих механизмов. Обычно выделяют два типа стойкости механизма KEM: стойкие к атаке на основе подобранного открытого текста (IND–CPA, **IND**istinguishability under **C**hosen **P**laintext **A**ttack) и стойкие к атаке на основе подобранного шифртекста (IND–CCA1/2, **IND**istinguishability under **C**hosen **C**iphertext **A**ttack).

IND–CCA–модель соответствует многоразовому использованию ключа, так как в рамках этой модели атакующий имеет возможность многократно подбирать шифртекст и наблюдать результаты декапсуляции. Следовательно, при многоразовом использовании пары (pk',sk') не исключается возможность получения атакующим информации об sk' на основании информации об ошибках декапсуляции [27]. Например, для механизма KEM BIKE [28] и его некоторых вариаций большое значение δ позволяет провести атаку с опорой на реакцию, нацеленную на восстановление секретного ключа (см. [29; 30]). Поэтому для многоразового использования ключей от KEM требуется обеспечение IND–CCA–стойкости λ_{KEM}, для достижения которой, в свою очередь, требуется, чтобы $\delta \leqslant 2^{-\lambda_{\mathsf{KEM}}}$. Стоит отметить, что для механизмов KEM с IND–CCA–стойкостью у исследователей пока нет единого мнения в пользу явного отказа, либо в пользу неявного отказа при декапсуляции в алгоритме Decaps [31], хотя в механизмах из конкурса NIST используются неявный отказ.

В случае эфемерной инкапсуляции информация об ошибке декапсуляции представляется малопригодной для атакующего. Поэтому, с одной стороны, для таких механизмов достаточно IND–CPA–стойкости [32], а с другой стороны, в таких механизмах может использоваться явный отказ, вместо неявного, что может сократить размер секретного ключа sk' и время декапсуляции [33].

Таким образом, для многоразового использования ключей вероятность (1.2) должна быть не более $2^{-\lambda_{\mathsf{KEM}}}$, где λ_{KEM} – требуемая IND–CCA–стойкость механизм KEM, а для одноразового использования эта вероятность должна соответствовать требуемому уровню отказоустойчивости информационной системы, который в разных системах может определяться по–разному. Например, в системах связи отказоустойчивость может определяться через вероятность разрыва связи. Другими словами, для одноразовых KEM верхняя граница вероятности (1.2) определяется тем, насколько часто в информационной системе допускается повторное выполнение KEM парой участников информационного взаимодействия. Верхнюю границу можно определить в виде $10^{-\gamma}$, где $\gamma \in \mathbb{N}$. Далее параметр γ будем называть *отказоустойчивостью* информационной системы, в которой используется механизм KEM, или кратко – отказоустойчивостью KEM. В [33] в качестве допустимых значений γ рассматриваются значения из множества $\{5,6,7\}$, при этом отмечается, что $\gamma = 5$ считается «золотым стандартом» отказоустойчивости информационных систем. В [10] в качестве допустимого рассматривается значение $\gamma = 9$.

Условие на вероятность (1.2) для одноразового и многоразового использования ключей можно сформулировать следующим образом:

$$\delta \leqslant \begin{cases} 2^{-\lambda_{\mathsf{KEM}}}, \text{многоразовое использование}, \\ 10^{-\gamma}, \text{одноразовое использование}, \end{cases} \tag{1.3}$$

где λ_{KEM} – требуемая IND–CCA–стойкость механизма KEM, γ – требуемая отказоустойчивость IND–CPA–стойкого механизма KEM.

Одним из наиболее популярных применений схемы KEM является протокол TLS (Transport Layer Security) [34], схема которого изображена на рисунке

1.2. В частности, механизм KEM реализован в рамках процедуры Key exchange этапа TLS Handshake. Протокол TLS используется для приложений, работающих в сети Internet, таких как веб-браузеры, работа с электронной почтой, обмен мгновенными сообщениями, IP-телефония (VoIP) и другие. При этом в основе

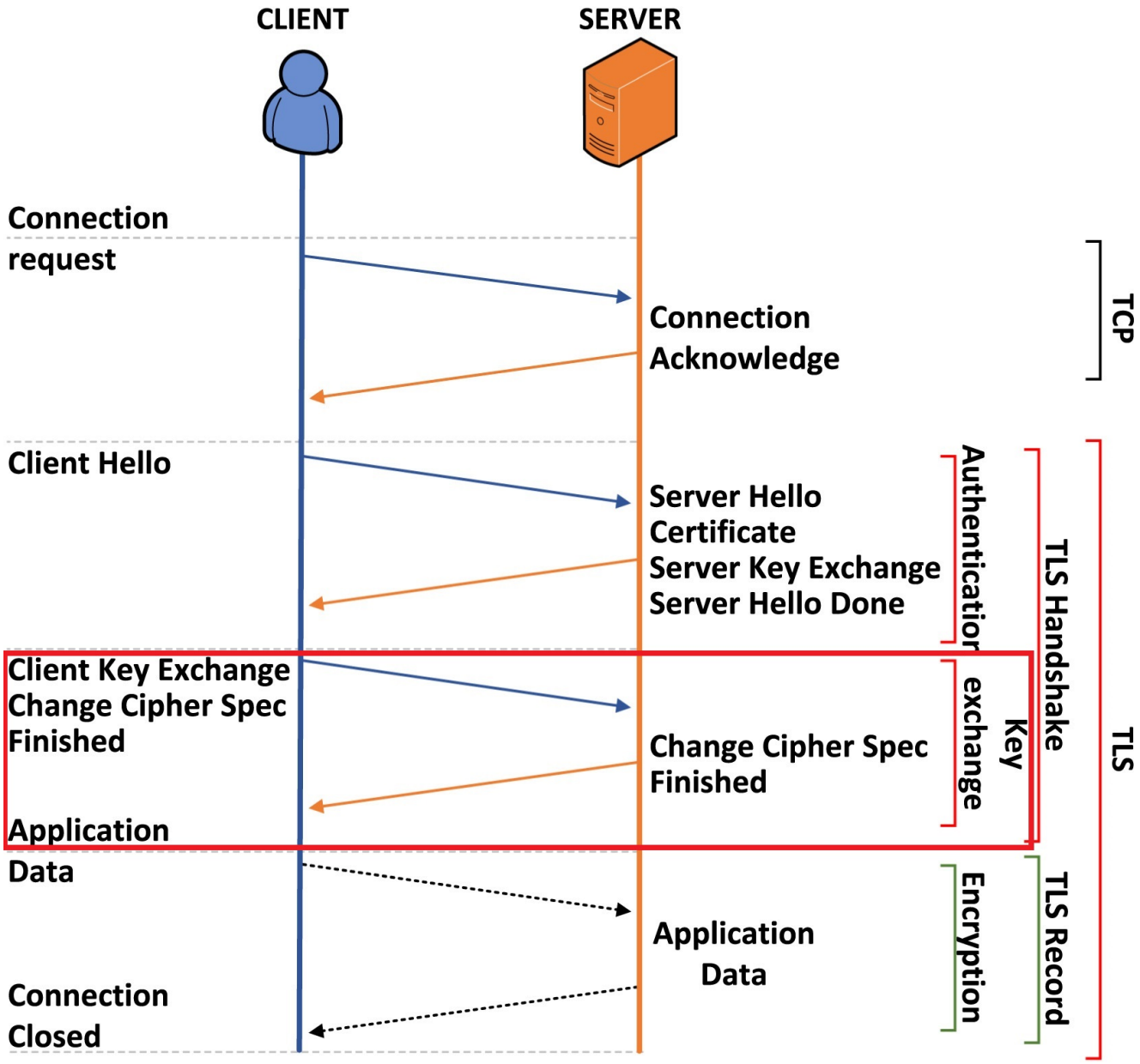

Рисунок 1.2 — Схема протокола TLS

KEM в зависимости от версии TLS используется система RSA или протокол Диффи–Хеллмана, которые основаны на сложности факторизации и дискретного логарифмирования. Как было сказано ранее, эти задачи имеют решение при использовании квантового компьютера [1], поэтому в данной работе предлагается использовать кодовую криптосистему типа Мак–Элиса.

1.2 Линейные коды

Далее определим понятие линейного кода – основы кодовой криптосистемы. Пусть $\mathbb{F}_q^n$ – векторное пространство над полем Галуа $\mathbb{F}_q$. Для вектора $\mathbf{x}(\in \mathbb{F}_q^n)$ множество его ненулевых координат называется *носителем* вектора $\mathbf{x}$ и обозначается $\mathrm{supp}(\mathbf{x})$. *Вес* $\mathrm{wt}(\mathbf{x})$ вектора $\mathbf{x}$ определяется как $|\mathrm{supp}(\mathbf{x})|$. (Здесь и далее символом $|A|$ обозначается мощность множества A.) Записью $[\![a,b]\!]$ будем обозначать диапазон целых чисел от a до b включительно. Конкатенацию двух векторов $\mathbf{x}$ и $\mathbf{y}$ обозначим $\mathbf{x} \parallel \mathbf{y}$. Линейное подпространство C размерности k пространства $\mathbb{F}_q^n$ называется *линейным кодом* [35]. Пусть $\dim(C) = k$ и $n(C) = n$ – размерность и длина кода соответственно, а $d(C) = \min_{\mathbf{c}\in C, \mathbf{c}\neq \mathbf{0}}\{\mathrm{wt}(\mathbf{c})\} = d$ – минимальное кодовое расстояние кода C, тогда код C называют $[n,k,d]_q$–кодом. В ряде случаев будем опускать d и писать $[n,k]_q$–код, если контекст этих случаев не будет требовать явного указания этого параметра. Порождающую матрицу кода C обозначим G_C, то есть $C = \mathcal{L}(G_C)$. Здесь и далее через $\mathcal{L}(U)$ обозначается линейная оболочка множества U. Проверочную матрицу кода C обозначим H_C. Двойственный код к коду C обозначим для удобства $\overline{C}$, то есть $\overline{C} = \mathcal{L}(G_{\overline{C}}) = \mathcal{L}(H_C)$. Коды C и D длины n и размерности k называются *переставновочно эквивалентными*, если в симметрической группе перестановок $\mathcal{P}_n$, действующей на множестве $\{1,...,n\}$, найдется такая перестановка σ, что

$$\sigma(C) = \{(c_{\sigma^{-1}(1)},...,c_{\sigma^{-1}(n)})|(c_1,...,c_n) \in C\} = D.$$

Под *суммой кодов* C и D длины n будем понимать код

$$C + D = \{\mathbf{x} + \mathbf{y}|\mathbf{x} \in C, \mathbf{y} \in D\} = \mathcal{L}(C \cup D) = \mathcal{L}\left(\left[\frac{G_C}{G_D}\right]\right).$$

Также понадобится определение *прямой (внешней) суммы кодов*. Пусть C_i – линейный $[n_i,k_i]_q$–код, $i \in [\![1,m]\!]$. Прямой (внешней) суммой кодов $C_1,...,C_m$ назовем код вида

$$C_1 \oplus C_2 \oplus ... \oplus C_m = \{(\mathbf{c}_1 \parallel \mathbf{c}_2 \parallel ... \parallel \mathbf{c}_m)|\mathbf{c}_i \in C_i, i \in [\![1,m]\!]\}.$$

Код C называется *разложимым*, если он перестановочно эквивалентен прямой сумме двух или более кодов ненулевой длины.

1.2.1 Произведение Шура–Адамара

Важным криптографическим свойством линейных кодов является свойство произведения Шура–Адамара, поскольку оно часто используется при анализе стойкости кодовых криптосистем. Произведение Шура–Адамара в ряде случаев позволяет свести криптоанализ системы на коде неизученной структуры к анализу известной кодовой криптосистемы [13; 21; 22].

Для двух векторов $\mathbf{a} = (a_1,...,a_n)$ и $\mathbf{b} = (b_1,...,b_n)$ из $\mathbb{F}_q^n$ их *покомпонентным произведением* назовем вектор $\mathbf{a} \star \mathbf{b} = (a_1b_1,...,a_nb_n)$. А покомпонентным произведением $k \times n$–матрицы $A = (\mathbf{a}_i)$ и $l \times n$–матрицы $B = (\mathbf{b}_j)$ назовем матрицу $A \star B = (\mathbf{a}_i \star \mathbf{b}_j)$, $i \in [\![1,k]\!]$, $j \in [\![1,l]\!]$. Для кодов C и D из $\mathbb{F}_q^n$ их покомпонентное произведение определяется следующим образом [36]:

$$C \star D = \mathcal{L}(\{\mathbf{x} \star \mathbf{y} | \mathbf{x} \in C, \mathbf{y} \in D\}).$$

Отметим, что это покомпонентное произведение кодов разные авторы могут называть произведением Адамара [26] или произведением Шура [36]. Здесь и далее это произведение будем называть произведением Шура–Адамара. Это же название будем использовать и для покомпонентного произведения векторов и матриц. Известно (см., например, [36]), что $C \star D = \mathcal{L}(G_C \star G_D)$. Произведение $C \star C$ далее обозначается C^2 и называется *квадратом* кода C.

1.2.2 Групповые коды

Пусть $\mathcal{G} = \{g_1 = \hat{1}, \ldots, g_{|\mathcal{G}|}\}$ – конечная группа с зафиксированным линейным порядком $\mathrm{Ord}(\mathcal{G})$ на множестве ее элементов, $\mathbb{F}_q$ – поле Галуа. Для групповой операции будем использовать мультипликативную запись. Рассмотрим групповую алгебру $\mathbb{F}_q\mathcal{G}$, элементами которой являются формальные суммы (функции):

$$\varphi = \sum_{g \in \mathcal{G}} a_g g, a_g \in \mathbb{F}_q \tag{1.4}$$

[37],[38]. Иногда такие формальные суммы будем называть векторами и использовать обозначение $\varphi(g) = a_g$. Операции сложения и умножения в групповой алгебре двух формальных сумм $\sum_{g\in\mathcal{G}} a_g g$ и $\sum_{g\in\mathcal{G}} b_g g$ выполняются по следующим формулам:

$$\left(\sum_{g\in\mathcal{G}} a_g g\right) + \left(\sum_{g\in\mathcal{G}} b_g g\right) := \sum_{g} (a_g + b_g) g,$$

$$\left(\sum_{g\in\mathcal{G}} a_g g\right)\left(\sum_{g\in\mathcal{G}} b_g g\right) := \sum_{g}\left[\sum_{w} a_{gw^{-1}} b_w\right] g, \tag{1.5}$$

где внутренняя сумма в квадратных скобках (1.5) представляет собой сумму над полем $\mathbb{F}_q$, а внешняя сумма правой части (1.5) является формальной суммой.

Нейтральным по умножению элементом $\mathbf{1}$ групповой алгебры $\mathbb{F}_q\mathcal{G}$ назовем функцию, которая принимает значение 1 ($\in \mathbb{F}_q$) на нейтральном элементе $\hat{1}$ группы $\mathcal{G}$ и 0 на всех остальных элементах группы. Нейтральным по сложению элементом $\mathbf{0}$ (нулем) этой групповой алгебры назовем функцию, принимающую нулевое значение на всех элементах группы. Через $\delta_g = 1g$ обозначим функцию Дирака в точке g. Отметим, что элемент групповой алгебры можно представить как линейную комбинацию функций Дирака в каждой точке группы $\mathcal{G}$: $\sum_{g\in\mathcal{G}} a_g \delta_g, a_g \in \mathbb{F}$. Из (1.5) вытекает, что $\delta_x \delta_y = \delta_{xy}$. Заметим, что аналогичные обозначения введены в [39].

В конечномерной групповой алгебре $\mathbb{F}_q\mathcal{G}$ зафиксируем базис $B = B^{\mathbb{F}_q\mathcal{G}} := \{\mathbf{g} = \delta_g\}_{g\in\mathcal{G}}$. Это позволяет рассматривать в $\mathbb{F}_q\mathcal{G}$ метрику Хэмминга. Отметим, что в категории конечномерных линейных пространств алгебра $\mathbb{F}_q\mathcal{G}$ естественно изоморфна пространству $\mathbb{F}_q^{|\mathcal{G}|}$, и соответствующий изоморфизм обозначим $\boldsymbol{\nu}_{\mathcal{G}}$.

В соответствии с [37], с. 39, всякий отличный от нулевого левый идеал C в групповой алгебре $\mathbb{F}_q\mathcal{G}$ называется *групповым* $\mathbb{F}_q\mathcal{G}$*–кодом* длины $n = n(C) = |\mathcal{G}|$. Идеал в групповой алгебре $\mathbb{F}_q\mathcal{G}$ является подпространством пространства функций $\mathbb{F}_q\mathcal{G}$, размерность $k = \dim(C)$ кода C – это размерность этого подпространства. Пусть $B^C = \{\varepsilon_1, \ldots, \varepsilon_{k(C)}\}(\subseteq \mathbb{F}_q\mathcal{G})$ – линейный базис идеала C. Заметим, что порядок $\mathrm{Ord}(\mathcal{G})$ индуцирует порядок на базисе $B^{\mathbb{F}_q\mathcal{G}}$ групповой алгебры $\mathbb{F}_q\mathcal{G}$, что позволяет по строкам выписать порождающую матрицу группового кода C:

$$G_C = \begin{pmatrix} \boldsymbol{\nu}_{\mathcal{G}}(\varepsilon_1) \\ \ldots \\ \boldsymbol{\nu}_{\mathcal{G}}(\varepsilon_{k(C)}) \end{pmatrix} \tag{1.6}$$

Отметим, что группа $\mathcal{G}$ действует слева на групповой алгебре $\mathbb{F}_q\mathcal{G}$ следующим естественным образом (см. [37], с. 32):

$$\mathcal{G} \times \mathbb{F}_q\mathcal{G} \ni (g, \varphi = \sum_{h\in\mathcal{G}} \varphi_h h) \mapsto \varphi g^{-1} := \sum_{h\in\mathcal{G}} \varphi_{hg^{-1}} h \in \mathbb{F}_q\mathcal{G}. \tag{1.7}$$

1.2.3 Тензорное произведение

Далее определим конструкцию тензорного произведения, которая далее будет необходима для определения D-кодов. Для $(k_1 \times n_1)$–матрицы $A = (a_{i,j})$ и $(k_2 \times n_2)$–матрицы B их тензорное произведение $A \otimes B$ определяется как

$(k_1k_2 \times n_1n_2)$–матрица вида

$$\begin{pmatrix} a_{1,1}B & \cdots & a_{1,n_1}B \\ \vdots & \ddots & \vdots \\ a_{k_1,1}B & \cdots & a_{k_1,n_1}B \end{pmatrix}. \tag{1.8}$$

Тензорное произведение $C_1 \otimes C_2$ двух $[n_i,k_i,d_i]_q$–кодов $C_i(\subset \mathbb{F}_q^{n_i})$, где $i \in \{1,2\}$, можно определить как $\mathcal{L}(G_{C_1} \otimes G_{C_2})$. Известно, что $C_1 \otimes C_2$ является $[n_1n_2, k_1k_2, d_1d_2]_q$–кодом ([40], n. 6.2.4.).

Тензорное произведение кодов обладает свойствами левой и правой дистрибутивности относительно суммы кодов. Именно, для линейных кодов $C \subseteq \mathbb{F}_q^m$, $A,B \subseteq \mathbb{F}_q^n$ выполняются равенства:

$$C \otimes A + C \otimes B = C \otimes (A + B),\ A \otimes C + B \otimes C = (A + B) \otimes C.$$

Докажем, например, свойство левой дистрибутивности. Пусть $O = A \cap B$ – пересечение кодов A и B, Δ_A и Δ_B – такие коды, что

$$\Delta_A \cap O = \mathbf{0}, \Delta_B \cap O = \mathbf{0}, \Delta_A + O = A, \Delta_B + O = B.$$

Порождающие матрицы кодов C, A, B, O, Δ_A, Δ_B обозначим соответственно G_C, G_A, G_B, G_O, G_{Δ_A} и G_{Δ_B}. Ясно, что матрицы

$$\begin{bmatrix} G_A \\ G_{\Delta_B} \end{bmatrix}, \begin{bmatrix} G_B \\ G_{\Delta_A} \end{bmatrix}, \begin{bmatrix} G_O \\ G_{\Delta_B} \\ G_{\Delta_A} \end{bmatrix}$$

являются порождающими матрицами кода $A+B$. Пусть $\mathbf{c} \in C \otimes A + C \otimes B$. То есть вектор $\mathbf{c}$ представим в виде $\mathbf{c} = \mathbf{x}(G_C \otimes G_A) + \mathbf{y}(G_C \otimes G_B)$, где $\mathbf{x} \in \mathbb{F}_q^{\dim(C \otimes A)}$, $\mathbf{y} \in \mathbb{F}_q^{\dim(C \otimes B)}$. Всегда можно найти такие матрицы перестановок $P_1 \in \mathcal{P}_{\dim(C \otimes A)}$

и $P_2 \in \mathcal{P}_{\dim(C\otimes B)}$, что справедлива следующая последовательность выкладок:

$$\mathbf{c} = \mathbf{x}(G_C \otimes G_A) + \mathbf{y}(G_C \otimes G_B) =$$

$$= (\mathbf{x} \parallel \mathbf{y}) \begin{bmatrix} G_C \otimes G_A \\ G_C \otimes G_B \end{bmatrix} = (\mathbf{x}P_1 \parallel \mathbf{y}P_2) \begin{bmatrix} G_C \otimes G_O \\ G_C \otimes G_{\Delta_A} \\ G_C \otimes G_O \\ G_C \otimes G_{\Delta_B} \end{bmatrix} = (\tilde{\mathbf{x}} \parallel \tilde{\mathbf{y}}) \begin{bmatrix} G_C \otimes G_O \\ G_C \otimes G_{\Delta_A} \\ G_C \otimes G_O \\ G_C \otimes G_{\Delta_B} \end{bmatrix}$$

$$= (\tilde{\mathbf{x}}_O \parallel \tilde{\mathbf{x}}_A \parallel \tilde{\mathbf{y}}_O \parallel \tilde{\mathbf{y}}_B) \begin{bmatrix} G_C \otimes G_O \\ G_C \otimes G_{\Delta_A} \\ G_C \otimes G_O \\ G_C \otimes G_{\Delta_B} \end{bmatrix} = (\tilde{\mathbf{x}}_O + \tilde{\mathbf{y}}_O \parallel \tilde{\mathbf{x}}_A \parallel \tilde{\mathbf{y}}_B) \begin{bmatrix} G_C \otimes G_O \\ G_C \otimes G_{\Delta_A} \\ G_C \otimes G_{\Delta_B} \end{bmatrix}$$

$$= ((\tilde{\mathbf{x}}_O + \tilde{\mathbf{y}}_O \parallel \tilde{\mathbf{x}}_A)P_1^{-1} \parallel \tilde{\mathbf{y}}_B) \begin{bmatrix} G_C \otimes G_A \\ G_C \otimes G_{\Delta_B} \end{bmatrix}.$$

Так как матрицы

$$\begin{bmatrix} G_C \otimes G_A \\ G_C \otimes G_{\Delta_B} \end{bmatrix}, G_C \otimes \begin{bmatrix} G_A \\ G_{\Delta_B} \end{bmatrix} \tag{1.9}$$

совпадают с точностью до перестановки строк и порождают код $C \otimes (A + B)$, то $\mathbf{c} \in C \otimes (A + B)$ и $C \otimes A + C \otimes B \subseteq C \otimes (A + B)$. Произвольный вектор $\mathbf{c} \in C \otimes (A + B)$ представляет собой линейную комбинацию строк матриц (1.9). Выполняя приведенные выше выкладки в обратном порядке, получим, что $\mathbf{c} \in C \otimes A + C \otimes B$, то есть $C \otimes (A + B) \subseteq C \otimes A + C \otimes B$. Таким образом, $C\otimes(A+B) = C\otimes A+C\otimes B$. Правая дистрибутивность доказывается аналогично.

Из теоремы 5 работы [41] вытекает, что

$$\overline{C_1 \otimes C_2} = \mathbb{F}_q^{n_1} \otimes \overline{C_2} + \overline{C_1} \otimes \mathbb{F}_q^{n_2}. \tag{1.10}$$

Действительно, согласно формуле (7) в [41], порождающая матрица кода $\overline{C_1 \otimes C_2}$ может быть представлена в виде:

$$G_{\overline{C_1\otimes C_2}} = \begin{pmatrix} G_{\overline{C_1}} \otimes G_{\overline{C_2}} \\ A_1 \otimes G_{\overline{C_2}} \\ G_{\overline{C_1}} \otimes A_2 \end{pmatrix},$$

где A_1, A_2 – такие $k_1 \times n_1$– и $k_2 \times n_2$–матрицы, что

$$\mathcal{L}\begin{pmatrix} G_{\overline{C_1}} \\ A_1 \end{pmatrix} = \mathbb{F}_q^{n_1}, \mathcal{L}\begin{pmatrix} G_{\overline{C_2}} \\ A_2 \end{pmatrix} = \mathbb{F}_q^{n_2}.$$

Следовательно,

$$\begin{aligned}
\overline{C_1 \otimes C_2} &= \overline{C_1} \otimes \overline{C_2} + \mathcal{L}(A_1) \otimes \overline{C_2} + \overline{C_1} \otimes \mathcal{L}(A_2) \\
&= \overline{C_1} \otimes \overline{C_2} + \mathcal{L}(A_1) \otimes \overline{C_2} + \overline{C_1} \otimes \mathcal{L}(A_2) + \overline{C_1} \otimes \overline{C_2} \\
&= (\overline{C_1} + \mathcal{L}(A_1)) \otimes \overline{C_2} + \overline{C_1} \otimes (\overline{C_2} + \mathcal{L}(A_2)) \\
&= \mathbb{F}_q^{n_1} \otimes \overline{C_2} + \overline{C_1} \otimes \mathbb{F}_q^{n_2}.
\end{aligned}$$

Известно (см. [42]), что

$$(C_1 \otimes C_2)^s = C_1^s \otimes C_2^s. \tag{1.11}$$

Отметим, что порождающая матрица кода $\mathbb{F}_q^n \otimes C$ имеет блочно-диагональный вид: $G_{\mathbb{F}_q^n \otimes C} = \mathrm{diag}(G_C,...,G_C)$. Поэтому код $\mathbb{F}_q^n \otimes C$ можно представить как прямую (внешнюю) сумму n кодов C:

$$\mathbb{F}_q^n \otimes C = \underbrace{C \oplus ... \oplus C}_{n}. \tag{1.12}$$

1.3 Декодирование линейных кодов

Для использования линейных кодов в криптографии необходимо наличие эффективного алгоритма декодирования, поскольку такие параметры кодовых криптосистем, как скорость расшифрования и стойкость к атакам на шифрограмму, напрямую зависят от выбранного декодера. Для линейных кодов существуют разные подходы к декодированию. Условно их можно разделить на гарантированное и вероятностное декодирование. В первом случае декодеры нацелены на гарантированное исправление любых векторов ошибок, имеющих вес в пределах половины кодового расстояния. Во втором случае декодеры

нацелены на исправление большего количества ошибок, но при этом они обладают ненулевой вероятностью ошибочного декодирования (DFR, decoding failure rate).

Декодером кода C будем называть отображение $\mathsf{Decoder} : \mathbb{F}_q^n \to C \cup \{\perp\}$, где $\perp$ – сообщение об ошибке, возвращаемое декодером при невозможности декодирования входного вектора. Сообщение об ошибке $\perp$ может возвращаться декодером $\mathsf{Decoder}$ в случае, когда, например, входной вектор не может быть однозначно декодирован. Алгоритм, реализующий декодирование, также будем обозначать $\mathsf{Decoder}$. Отметим, что входными параметрами такого алгоритма, кроме декодируемого вектора $\mathbf{z}$, могут быть специфичные для кода и/или декодера параметры, например, условие выхода из алгоритма и т. п. Такие специфичные параметры p_1,p_2,... будем отделять точкой с запятой и писать $\mathsf{Decoder}(\mathbf{z}; p_1,p_2,\dots)$.

1.3.1 Декодирование по информационным совокупностям

Одним из универсальных способов декодирования является декодирование по информационным совокупностям (далее – ДИС или ISD от английского **I**nformation **S**et **D**ecoding). Этот подход к декодированию линейных кодов предложен в работе [43]. Несмотря на то, что известен ряд усовершенствований этого алгоритма (см., например, [44; 45]), в данной работе будет рассматриваться классический вариант.

Приведем описание алгоритма ДИС. Информационной совокупностью линейного $[n,k]_q$–кода C с порождающей матрицей G_C называется такое множество $\boldsymbol{\tau} \subset [\![1,n]\!]$ мощности k, что столбцы любой порождающей матрицы этого кода с номерами из $\boldsymbol{\tau}$ линейно независимы. Обычно при декодировании по информационным совокупностям (см., например, оригинальную работу [43]) случайным образом выполняется поиск такой информационной совокупности $\boldsymbol{\tau}$,

чтобы номера ненулевых координат вектора ошибки не принадлежали $\boldsymbol{\tau}$. Если M – матрица, состоящая из r столбцов, $\boldsymbol{\omega} \subseteq [\![1,r]\!]$, то матрицу, состоящую из столбцов с номерами из $\boldsymbol{\omega}$, будем обозначать $\Pi_{\boldsymbol{\omega}}(M)$ или иногда просто $\boldsymbol{\omega}(M)$.

Пусть $\mathbf{z} = \mathbf{c} + \mathbf{e}$ – зашумленное кодовое слово, где $\mathbf{c} = \mathbf{m}G_C$, $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{c} \in C$, $\mathbf{e} \in \mathbb{F}_q^n$, $\mathrm{wt}(\mathbf{e}) = t$. Пусть также $\boldsymbol{\tau} \subset [\![1,n]\!]$ – случайно выбранная информационная совокупность кода C. Тогда декодирование зашумленного кодового слова $\mathbf{z}$ заключается в вычислении матрицы $G' = \Pi_{\boldsymbol{\tau}}(G)^{-1}$ и последующей попытке вычисления исходного информационного вектора $\mathbf{m}' = \Pi_{\boldsymbol{\tau}}(\mathbf{z})G'$. Далее выполняется проверка правильности декодирования. Именно, вычисляется $w = \mathrm{wt}(\mathbf{z} - \mathbf{m}'G_C)$. Если $\boldsymbol{\tau} \cap \mathrm{supp}(\mathbf{e}) = \varnothing$, то $w = t$ и $\mathbf{m} = \mathbf{m}'$, в противном случае в качестве $\boldsymbol{\tau}$ выбирается другая информационная совокупность и вышеописанные действия повторяются.

Отметим, что на вероятность нахождения информационной совокупности, соответствующей безошибочным координатам зашумленного кодового слова, и, как следствие, на правильное декодирование влияет соотношение длины и размерности кода, а также количество ошибок t. Эта вероятность вычисляется по формуле

$$P_{ISD} = \frac{C_{n-t}^k}{C_n^k}. \tag{1.13}$$

1.3.2 Мажоритарный декодер Мэсси

В данном разделе рассматривается задача мажоритарного декодирования. Эффективность мажоритарного декодирования линейных кодов впервые стала ясна после работы, посвященной декодированию двоичных кодов Рида–Маллера. Первая фундаментальная работа о мажоритарном декодировании линейных кодов принадлежит Мэсси [46]. Мажоритарный метод декодирования основан на вычислении скалярных произведений векторов и процедуре голосования, что и делает его весьма эффективным.

Согласно современным данным, к важнейшим кодам, полностью или большей частью декодируемым с помощью мажоритарного метода, относятся так называемые евклидово–геометрические и проективно–геометрические коды, симплексные коды, а также коды, связанные с разностными множествами. Однако нахождение линейных кодов, допускающих мажоритарное декодирование, до сегодняшнего дня остается очень трудной задачей.

Приведем необходимые сведения о конструкции мажоритарного декодера. Пусть V – некоторое векторное пространство. Множество векторов $\mathcal{M}_{\mathbf{v}} = \{\mathbf{v}^{(1)},...,\mathbf{v}^{(r)}\}(\subset V)$ называется M-ортогональным вектору $\mathbf{v}(\in V)$, если $|\mathrm{supp}_B(\mathbf{v}^{(i)})| > |\mathrm{supp}_B(\mathbf{v})|$ для всех $i \in [\![1,r]\!]$ и

$$\forall\ i \neq j: \quad \mathrm{supp}_B(\mathbf{v}^{(i)}) \cap \mathrm{supp}_B(\mathbf{v}^{(j)}) = \mathrm{supp}_B(\mathbf{v}). \tag{1.14}$$

Пусть $\mathbf{c}(\in C)$ – кодовый вектор, $\mathbf{x} = \mathbf{c} + \mathbf{e}$ – принятый и канала вектор, $\mathrm{wt}(\mathbf{e}) \leqslant \lfloor (d(C)-1)/2 \rfloor$. Рассмотрим разложение $\sum_{\mathbf{b}\in B} e_{\mathbf{b}}\mathbf{b}$ вектора ошибок $\mathbf{e}$ по базису B. Если для $\mathbf{b}$-координаты существует *декодирующее дерево* $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}$, такое, что $\lfloor r_{\mathbf{b}}/2 \rfloor \geqslant \mathrm{wt}(\mathbf{e})$, то значение $e_{\mathbf{b}}$ для $\mathbf{b}$-координаты вектора $\mathbf{e}$ находится однозначно с помощью мажоритарного декодера. Декодирующим деревом $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}} = \mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$ для $\mathbf{b}$-координаты здесь и далее будем называть в соответствии с [39] помеченное дерево с корнем $\mathbf{b}$, обладающее следующими свойствами:

1) множество вершин этого дерева состоит из $L_{\mathbf{b}}+1$ уровня; корень с меткой $\mathbf{b}(\in B)$ находится на уровне 0, а листья – на уровне $L_{\mathbf{b}}$; метки вершин i-ого уровня образуют набор V_i, $i \in [\![0,L_{\mathbf{b}}]\!]$;

2) листья дерева помечены элементами из $\overline{C}$;

3) каждая вершина, не являющаяся листом, имеет не менее $r_{\mathbf{b}}(\in \mathbb{N})$ непосредственно следующих за ней вершин;

4) с меткой $\mathbf{p}$ каждой вершины дерева связывается числовое значение $l(\mathbf{p})(\in \mathbb{F})$ метки, вычисляемое в зависимости от значения принятого из канала вектора $\mathbf{x}$.

5) метки вершин, непосредственно следующих из произвольной вершины $\mathbf{p}$, находящейся на уровне $i(0 \leqslant i < L_{\mathbf{b}})$, образуют в совокупности множество $\mathcal{M}_{\mathbf{p}}$, M-ортогональное $\mathbf{p}$; символом $l[\mathcal{M}_{\mathbf{p}}]$ обозначается набор $(l(\mathbf{q}))_{\mathbf{q}\in\mathcal{M}_{\mathbf{p}}}$.

Для каждого листа $\mathbf{p}$ дерева значение $l(\mathbf{p})$ равно скалярному произведению $(\mathbf{p},\mathbf{x})$ векторов $\mathbf{p}$ и $\mathbf{x}$, а для вершин на уровне $i(0 \leqslant i \leqslant L_{\mathbf{b}} - 1)$ значение $l(\mathbf{p})$ вычисляется с помощью мажоритарного голосования;

Тогда мажоритарное декодирование заключается в следующем. Для вершин $\mathbf{p}$ на последнем уровне дерева вычисляется скалярное произведение $(\mathbf{p},\mathbf{x}) = (\mathbf{p},\mathbf{c}) + (\mathbf{p},\mathbf{e})$. Так как, согласно условию 2 декодирующего дерева, $\mathbf{p} \in \overline{C}$, то $(\mathbf{p},\mathbf{c}) = 0$. Следовательно, результатом скалярного произведения будет сумма соответствующих $\mathbf{p}$ координат вектора ошибок $\mathbf{e}$. На уровне ниже и последующих, значение метки $l(\mathbf{p})$ вершины $\mathbf{p}$ вычисляется с помощью мажоритарного голосования. Среди значений меток вершин, составляющих M-ортогональное множество для вершины $\mathbf{p}$, выбирается значение, которое встречается наибольшее количество раз. Таким образом, вычисляется значение метки для вершины $\mathbf{b}(\in B)$, в которой есть только одна ненулевая координата. Значение метки будет соответствовать значению вектора ошибок в этой координате.

Если для кода C существует такой набор

$$\mathcal{WB}(C) = \{\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}\}_{\mathbf{b}\in B}, \tag{1.15}$$

для которого $\mathrm{dmaj}_B(C) = \min_{\mathbf{b}\in B}\{r_{\mathbf{b}}\} = d(C) - 1$, то код C называют MLD-кодом (Majority Logic Decodable). Заметим, что $\mathrm{dmaj}_B(C) \leqslant d(C) - 1$, иначе в противном случае получили бы, что код может гарантированно исправлять более $\lfloor (d(C) - 1)/2 \rfloor$ ошибок.

Построение набора $\mathcal{WB}(C)$ представляется в общем случае сложной задачей. С одной стороны, сложной представляется задача построения дерева для фиксированной координаты, а с другой стороны, деревья для разных координат строятся независимо. В то же время вторая задача решается просто для

групповых кодов за счет дополнительных алгебраических свойств, если имеется декодирующее дерево хотя бы для одной координаты. Отметим, что если C – групповой код, то $\overline{C}$ – также групповой код [37], то есть левый идеал. В силу этого действие группы $\mathcal{G}$ на $\mathbb{F}_q\mathcal{G}$ по правилу (1.7) индуцирует действие этой группы на коде $\overline{C}$. С другой стороны, группа $\mathcal{G}$ действует транзитивно на элементах из базиса $B^{\mathbb{F}_q\mathcal{G}}$ и не нарушает M-ортогональности (см. [39], раздел 2.2). Это позволяет в случае групповых MLD–кодов построить набор декодирующих деревьев, описанных в данном разделе, для всех координат только по одному такому дереву. В частности, если для базисной функции $\mathbf{1} = \delta_{\hat{1}} = 1\hat{1}$ удалось построить декодирующее дерево $\mathrm{WB}_{\mathbf{1},r_\mathbf{b},L_\mathbf{b}}[C]$, то дерево $\mathrm{WB}_{\mathbf{g},r_\mathbf{b},L_\mathbf{b}}[C]$ для базисной функции $\mathbf{g} = \delta_g = 1g$ может быть построено путем действия элементов $g^{-1}(\in \mathcal{G})$ на метки узлов дерева $\mathrm{WB}_{\mathbf{1},r_\mathbf{b},L_\mathbf{b}}[C]$ по правилу (1.7).

1.4 Коды Рида–Маллера

Широко известным классом линейных кодов является класс бинарных (двоичных) кодов Рида–Маллера [47]. Для их определения и исследования некоторых их свойств рассмотрим $\mathbb{F}_2[x_1,...,x_m]$ – кольцо полиномов от m переменных над полем $\mathbb{F}_2$. Многочлены из $\mathbb{F}_2[x_1,...,x_m]$ будем записывать следующим образом:

$$f(x_1,...,x_m) \;=\; \sum_{\boldsymbol{\alpha}=(\alpha_1,...,\alpha_m)\in\mathbb{F}_2^m} f_{\boldsymbol{\alpha}}\bar{x}^{\boldsymbol{\alpha}},$$

где $\bar{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1}...x_m^{\alpha_m}$ – моном степени $\mathrm{wt}(\boldsymbol{\alpha})$. Для $\boldsymbol{\alpha} = (\alpha_1,...,\alpha_m) \in \mathbb{F}_2^m$ через $f(\boldsymbol{\alpha})$ обозначим значение многочлена $f(x_1,...,x_m)$, вычисленное при $x_i = \alpha_i$, $i \in [\![1,m]\!]$. Степенью многочлена f назовем максимальную степень его мономов. Через $\mathbb{F}_2^{(r)}[x_1,...,x_m]$ обозначим линейное пространство многочленов из $\mathbb{F}_2[x_1,...,x_m]$ степени, не превышающей r. Определим оператор $\mathrm{Eval}_{(m,r)} :\ \mathbb{F}_2^{(r)}[x_1,...,x_m] \to \mathbb{F}_2^n$

следующим образом:

$$\mathrm{Eval}_{(m,r)}(f) = (f(\boldsymbol{\alpha}_1), ..., f(\boldsymbol{\alpha}_{2^m})),$$

где $\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \in \mathbb{F}_2^m$, $\boldsymbol{\alpha}_i \neq \boldsymbol{\alpha}_j$ для $i \neq j$, $n = 2^m$. Двоичный код Рида–Маллера $\mathrm{RM}_2(r,m)$ с параметрами r и m определяется следующим образом:

$$\mathrm{RM}_2(r,m) = \{\mathrm{Eval}_{(m,r)}(f) | f \in \mathbb{F}_2^{(r)}[x_1,...,x_m]\},$$

при этом $\mathrm{RM}_2(r,m) = \mathrm{RM}_2(m,m) = \mathbb{F}_2^n$ для $r \geqslant m$.

Известно, что $\overline{\mathrm{RM}_2(r,m)} = \mathrm{RM}_2(m-r-1,m)$, а в [48] доказано, что

$$\mathrm{RM}_2(r_1,m) \star \mathrm{RM}_2(r_2,m) = \mathrm{RM}_2(r_1+r_2,m). \tag{1.16}$$

Из (1.16) и определения кода Рида–Маллера вытекает, что $\mathrm{RM}_2(r_1,m) \star \mathrm{RM}_2(r_2,m) = \mathbb{F}_2^n$ при $r_1 + r_2 \geqslant m$.

Справедлива следующая лемма [49].

Лемма 1. *Пусть* $\mathrm{RM}_2(r_i,m_i)$ *– бинарный код Рида–Маллера,* $i \in \{1,2\}$. *Тогда*

$$\mathrm{RM}_2(r_1,m_1) \otimes \mathrm{RM}_2(r_2,m_2) \subset \mathrm{RM}_2(r_1+r_2, m_1+m_2).$$

Доказательство. Рассмотрим тензорное произведение пространств $\mathbb{F}_2^{(r_1)}[x_1,...,x_{m_1}]$ и $\mathbb{F}_2^{(r_2)}[y_1,...,y_{m_2}]$, которое, по определению, имеет вид:

$$\mathbb{F}_2^{(r_1)}[x_1,...,x_{m_1}] \otimes \mathbb{F}_2^{(r_2)}[y_1,...,y_{m_2}] = \\ = \{f \cdot g | f \in \mathbb{F}_2^{(r_1)}[x_1,...,x_{m_1}],\, g \in \mathbb{F}_2^{(r_2)}[y_1,...,y_{m_2}]\},$$

где $f \cdot g$ – произведение полиномов в кольце $\mathbb{F}_2[x_1,...,x_{m_1},y_1,...,y_{m_2}]$. Очевидно, что

$$\mathbb{F}_2^{(r_1)}[x_1,...,x_{m_1}] \otimes \mathbb{F}_2^{(r_2)}[y_1,...,y_{m_2}] \subset \mathbb{F}_2^{(r_1+r_2)}[x_1,...,x_{m_1},y_1,...,y_{m_2}].$$

Пусть $f \in \mathbb{F}_2^{(r_1)}[x_1,...,x_{m_1}]$, $g \in \mathbb{F}_2^{(r_2)}[y_1,...,y_{m_2}]$. Покажем, что

$$\mathrm{Eval}_{(m_1+m_2,r_1+r_2)}(f \cdot g) = \mathrm{Eval}_{(m_1,r_1)}(f) \otimes \mathrm{Eval}_{(m_2,r_2)}(g).$$

Так как $(f \cdot g)(\boldsymbol{\alpha},\boldsymbol{\beta}) = f(\boldsymbol{\alpha}) \cdot g(\boldsymbol{\beta})$ для $\boldsymbol{\alpha} \in \mathbb{F}_2^{m_1}$, $\boldsymbol{\beta} \in \mathbb{F}_2^{m_2}$, то, с одной стороны,

$$\begin{aligned}\mathrm{Eval}_{(m_1+m_2,r_1+r_2)}(f \cdot g) &= \big((f \cdot g)(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_j)\big)_{i=1,\ldots,2^{m_1},j=1,\ldots,2^{m_2}} \\ &= \big(f(\boldsymbol{\alpha}_i) \cdot g(\boldsymbol{\beta}_j)\big)_{i=1,\ldots,2^{m_1},j=1,\ldots,2^{m_2}}.\end{aligned}$$

С другой стороны, по определению тензорного произведения (1.8), получаем

$$\mathrm{Eval}_{(m_1,r_1)}(f) \otimes \mathrm{Eval}_{(m_2,r_2)}(g) = \big(f(\boldsymbol{\alpha}_i) \cdot g(\boldsymbol{\beta}_j)\big)_{i=1,\ldots,2^{m_1},j=1,\ldots,2^{m_2}},$$

что доказывает утверждение. □

1.4.1 Коды Рида–Маллера–Бермана

В [50] С.Д. Берманом открыто важное семейство групповых кодов. Рассмотрим эти коды, используя обозначения из [39]. Абелева группа G представима в виде *внутреннего произведения* подгрупп H и K, если они пересекаются только в нейтральном элементе группы G и каждый элемент $g \in G$ представляется единственным образом в виде $g = hk$, $h \in H$, $k \in K$. В этом случае будем писать $G = H \times K$. Пусть $\mathbb{F}_q = \mathbb{F}_2$, $\Delta_{2,n}$ – абелева 2-группа поля $\mathbb{F}_{2^n}$ порядка 2^n, имеющая представление в виде внутреннего произведения n подгрупп, каждая из которых имеет порядок 2:

$$\Delta_{2,n} =< g_1 > \times \ldots \times < g_n >, \tag{1.17}$$

где для $i \in [\![1,n]\!]$, элемент g_i – образующий элемент порядка 2, $g_i^2 = \hat{1}$. Пусть $\{h_1; \ldots; h_n\}$ – минимальная система порождающих элементов группы $\Delta_{2,n}$, то есть каждый элемент $g(\in \Delta_{2,n})$ представим в виде $g = h_1^{a_1} \ldots h_n^{a_n}$, $(a_1,\ldots,a_n \in \{0;1\})$, при этом никакое подмножество этого множества таким свойством не обладает [37]. Приведем конструкцию построенного Берманом семейства кодов в групповой алгебре $\mathbb{F}_2\Delta_{2,n}$, которое изоморфно семейству бинарных кодов Рида–Маллера $\mathrm{RM}_2(r,m)$. Пусть

$$B = \{\mathbf{1}; \delta_{g_1}; \ldots; \delta_{g_n}; \delta_{g_1}\delta_{g_2}; \ldots; \delta_{g_1}\delta_{g_n}; \ldots; \delta_{g_{n-1}}\delta_{g_n}; \ldots; \delta_{g_1}\ldots\delta_{g_n}\}$$

– стандартный базис групповой алгебры $\mathbb{F}_2\Delta_{2,n}$, которая как $\mathbb{F}_2$–линейное пространство изоморфна пространству $\mathbb{F}_2^{2^n}$. Кроме базиса B групповая алгебра $\mathbb{F}_2\Delta_{2,n}$ имеет также базис (см. [37], с. 49)

$$B_0 = \left\{ (\delta_{g_1} - \mathbf{1})^{a_1} ... (\delta_{g_n} - \mathbf{1})^{a_n} | a_1, ..., a_n \in \{0; 1\}, \sum_{s=1}^{n} a_s \geqslant 0 \right\}.$$

Далее предполагается, что в базисе B_0 элементы упорядочены следующим образом:

$$B_0 = \{\mathbf{1}; (\delta_{g_1} - \mathbf{1}); ...; (\delta_{g_n} - \mathbf{1}); (\delta_{g_1} - \mathbf{1})(\delta_{g_2} - \mathbf{1}); ...; (\delta_{g_1} - \mathbf{1})(\delta_{g_n} - \mathbf{1}); ...;$$

$$(\delta_{g_1} - \mathbf{1}) ... (\delta_{g_n} - \mathbf{1})\}.$$

Пусть $t \in \mathbb{N}$. Определим групповой код Бермана $\mathcal{RM}_t$ так: $\mathcal{RM}_t = \mathcal{L}(B_t)$, где

$$B_t := \left\{ (\delta_{g_1} - \mathbf{1})^{a_1} ... (\delta_{g_n} - \mathbf{1})^{a_n} | a_1, ..., a_n \in \{0; 1\}, \sum_{s=1}^{n} a_s \geqslant t \right\}. \tag{1.18}$$

Из этого определения вытекает, что

$$\mathcal{RM}_n = \{\mathbf{0}; (\delta_{g_1} - \mathbf{1}) ... (\delta_{g_n} - \mathbf{1})\}. \tag{1.19}$$

Из вложения $B_0 \supset B_1 \supset B_2 \supset ... \supset B_n \supset \{\mathbf{0}\}$ следует вложение $\mathbb{F}_2\Delta_{2,n} = \mathcal{RM}_0 \supset \mathcal{RM}_1 \supset \mathcal{RM}_2 \supset ... \supset \mathcal{RM}_n \supset \{\mathbf{0}\}$.

В [50] доказано, что для любого t код $\mathcal{RM}_t$ изоморфен бинарному коду Рида–Маллера $\mathrm{RM}_2(n-t,n)$. Так как $\overline{\mathrm{RM}_2(n-t,n)} = \mathrm{RM}_2(t-1,n)$ ([51], с.203), то $\overline{\mathcal{RM}_t} = \mathcal{RM}_{n-t+1}$. Далее эти коды будем называть кодами Рида–Маллера–Бермана или просто кодами Рида–Маллера в контексте групповых кодов.

1.4.2 Декодирование кодов Рида–Маллера

Для кодов Рида–Маллера существуют как гарантированные декодеры, так и вероятностные [47]. Примерами гарантированных декодеров являются

классический декодер Рида [52] и мажоритарный декодер Мэсси [46], [53]. Примерами вероятностных декодеров являются декодер Сидельникова–Першакова [54], его модификации и списочный декодер Думера, а также декодер для низкоплотностных кодов [55], перестановочный декодер [56], рекурсивный декодер Йе–Аббэ [57]. Отметим, что одним из первых вероятностных декодеров является декодер Сидельникова–Першакова, в то время как одним из последних таких декодеров является декодер Йе–Аббе, который, согласно [57], обладает лучшей корректирующей способностью по сравнению с декодером Сидельникова–Першакова. Отметим также, что вероятностные декодеры, как правило, вычислительно сложнее, чем эффективные гарантированные.

Построение декодирующего дерева

Коды Рида–Маллера принадлежат классу MLD–кодов, описанных в разделе 1.3.2. Это значит, что для них может быть построен набор декодирующих деревьев вида (1.15). Приведем алгоритм построения декодирующих деревьев для кодов Рида–Маллера подробно описанный в [39].

Рассмотрим коды $\mathcal{RM}_n$ и $\mathcal{RM}_1 = \overline{\mathcal{RM}}_n$. Определим множество

$$\mathcal{M} = \left\{\delta_g - \mathbf{1} : g \in \Delta_{2,n}, g \neq \hat{1}\right\} \subseteq \mathcal{RM}_1.$$

Так как $\mathcal{RM}_n$ имеет вид (1.19), то множество $\mathcal{M}$ является M-ортогональным относительно базисного элемента $\mathbf{1}(\in B^{\mathbb{F}_2\Delta_{2,n}})$. Это означает, что для базисного элемента $\mathbf{1}$ можно построить двухуровневое декодирующее дерево $\mathrm{WB}_{\mathbf{1},2^n-1,1}$.

В [37] предложен способ построения декодирующего дерева для кода $\mathcal{RM}_t$ при $1 \leqslant t < n$, когда известно декодирующее дерево для кода $\mathcal{RM}_{t+1}$. В работе [39] разработан алгоритм, реализующий этот способ. Для полноты изложения приведем описание этого алгоритма. В [39] вводятся два вспомогательных алгоритма: MakeGenSystem и MakeGroup. Первый алгоритм по

элементу $\mathbf{p}(\in \mathbb{F}_2\Delta_{2,n})$ вида

$$\mathbf{p} = (\delta_{h_1} - \mathbf{1})...(\delta_{h_s} - \mathbf{1}), \tag{1.20}$$

где $S_{\mathbf{p}} = \{h_1; ...; h_s\}$ – подмножество некоторой минимальной системы порождающих элементов, достраивает $S_{\mathbf{p}}$ до другой минимальной системы порождающих элементов $\mathcal{M}_{\mathbf{p}} = \{h_1; ...; h_s; f_1; ...; f_{n-s}\}$ и возвращает $\mathcal{M}_{\mathbf{p}} \backslash \mathcal{S}_{\mathbf{p}}$. Второй алгоритм по подмножеству S группы $\Delta_{2,n}$ строит подгруппу $< S >$. Теперь приведем разработанный в [39] алгоритм MakeRMWBTree построения декодирующего дерева кода $\mathcal{RM}_t$ с корнем $\mathbf{1}$ по декодирующему дереву кода $\mathcal{RM}_{t+1}$ с тем же корнем. В алгоритме для каждой

Алгоритм 1 MakeRMWBTree

Исходные параметры: $t(\in [\![1, n-1]\!])$, дерево $\mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathcal{RM}_{t+1}]$, каждая метка $\mathbf{p}$ которого на уровне $L_{\mathbf{1}}$ имеет вид (1.20), где $s = n - (t+1) + 1$

Результат: дерево $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathcal{RM}_t]$

1: $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathcal{RM}_t] := \mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathcal{RM}_{t+1}]$

2: **для каждого** метки $\mathbf{p}$ на уровне $L_{\mathbf{1}}$ в дереве $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathcal{RM}_t]$ **выполнять**

3: на уровень $L_{\mathbf{1}} + 1$ добавить $2^t - 1$ вершину и соединить их с вершиной, имеющей метку $\mathbf{p}$

4: добавленным вершинам однозначно присвоить метки

$$\mathbf{q}_h := \mathbf{p}(\delta_h - \mathbf{1}),\ h \in \mathrm{MakeGroup}(\mathrm{MakeGenSystem}(\mathbf{p})),\ h \neq \hat{1}; \tag{1.21}$$

5: **вернуть** $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathcal{RM}_t]$

вершины $\mathbf{p}$ строится вспомогательное множество с помощью алгоритмов MakeGenSystem, MakeGroup. Далее формальная сумма $\mathbf{p}$ умножается по правилу 1.5 на каждый из элементов вспомогательного множества. Полученные формальные суммы составляют M-ортогональное множество для вершины $\mathbf{p}$. Подробное обоснование алгоритма приводится в [39]. Данный алгоритм реализован на C++, с реализацией можно ознакомиться в [58] (см. `https://github.com/lelukevgeniy/Dissertation-D-codes/blob/main/Tensor%20ML%20Decoding/TensorDecoding.cpp`).

1.5 Криптосистема типа Мак–Элиса и механизм KEM на её основе

1.5.1 Криптосистема типа Мак–Элиса

Далее опишем схему асимметричного шифрования типа Мак–Элиса и соответствующий ей механизм KEM. Пусть C – линейный $[n,k,d]_q$–код с фиксированной порождающей матрицей G, для которого известен *эффективный* декодер $\mathsf{Decoder}$, G^{-1} – псевдообратная к G матрица, то есть такая матрица, что GG^{-1} – единичная $(k \times k)$–матрица. Пусть также $\mathrm{GL}_k(\mathbb{F}_q)$ – кольцо обратимых $(k \times k)$–матриц, $\mathcal{P}_n$ – кольцо $(n \times n)$–матриц перестановок, $\mathcal{E}_{n,t} = \{\mathbf{y} \in \mathbb{F}_q^n : \mathrm{wt}(\mathbf{y}) = t\}$. Схему $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ типа Мак–Элиса на основе кода C обозначим $\mathrm{McE}(C) = (\mathsf{Gen}^{\mathrm{McE}(C)}, \mathsf{Enc}, \mathsf{Dec})$; соответствующие алгоритмы приведены на рис. 1.3. Публичной матрицей будем называть матрицу $\hat{G}$ следующего вида:

$$\tilde{G} = SGP, \tag{1.22}$$

где $S \leftarrow \mathrm{GL}_k(\mathbb{F}_q)$, $P \leftarrow \mathcal{P}_n$.

Отметим, что здесь и далее рассматривается схема Мак–Элиса, когда код C не является секретным, поэтому матрица G и все другие связанные с ним параметры является частью алгоритмов $\mathsf{Gen}^{\mathrm{McE}(C)}$, Enc, Dec и не указывается в списке аргументов.

Так как в ряде случаев декодер кода C может исправлять векторы ошибок веса больше, чем $\lfloor (d-1)/2 \rfloor$, то для алгоритма $\mathsf{Gen}^{\mathrm{McE}(C)}$ введен один входной параметр: вес t добавляемого вектора ошибок. В алгоритме шифрования Enc вектор $\mathbf{e}$ веса t может выбираться, например, случайно из $\mathcal{E}_{n,t}$, или генерироваться детерминированно, например, в зависимости от $\mathbf{m}$ с помощью инъективного отображения из $\mathbb{F}_q^k$ в $\mathcal{E}_{n,t}$. В первом случае схему $\mathrm{McE}(C)$ будем называть схемой с *вероятностным шифрованием*, а во втором – с *детерминированным шифрованием*.

$\mathsf{Gen}^{\mathrm{McE}(C)}(t)$:

---

1. $S \leftarrow \mathrm{GL}_k(\mathbb{F}_q)$
2. $P \leftarrow \mathcal{P}_n$
3. $pk = (\tilde{G} = SGP, t)$
4. $sk = (S, P)$
5. ***вернуть*** (pk, sk)

$\mathsf{Enc}(pk, \mathbf{m}; \mathbf{e})$:

---

1. $\mathbf{c} = \mathbf{m}\tilde{G} + \mathbf{e}$, $\mathrm{wt}(\mathbf{e}) = t$
2. ***вернуть*** $\mathbf{c}$

$\mathsf{Dec}(sk, \mathbf{c})$:

---

1. $\mathbf{x} \leftarrow \mathsf{Decoder}(\mathbf{c}P^{-1})$
2. *если* $\mathbf{x} \neq \perp$
3. $\quad \mathbf{m}' = \mathbf{x}G^{-1}S^{-1}$
4. $\quad$ ***вернуть*** $\mathbf{m}'$
5. *иначе*
6. $\quad$ ***вернуть*** $\perp$

Рисунок 1.3 — Схема асимметричного шифрования $\mathrm{McE}(C) = (\mathsf{Gen}^{\mathrm{McE}(C)}, \mathsf{Enc}, \mathsf{Dec})$

1.5.2 Атаки на кодовые криптосистемы

При оценке стойкости асимметричных криптосистем принято разделять стойкость к атакам восстановления ключа или структурную стойкость и стойкость к атакам на шифрограмму. При этом для кодовых криптосистем структурные атаки можно разделить на три типа: атаки с полным восстановлением ключа, атаки с частичным восстановлением ключа и атаки–различители. В случае атак с полным восстановлением ключа криптоаналитик получает полную информацию о секретном ключе и может дешифровать любые сообщения. Атаки с частичным восстановлением ключа позволяют получить лишь

некоторую информацию о секретном ключе, которая впоследствии может быть использована для увеличения вероятности атаки на шифрограмму. Атаки–различители позволяют отличить используемый код от случайного. Несмотря на то, что в общем случае атаки–различители не позволяют получить информацию о секретном ключе, многие атаки с полным или частичным восстановлением ключа были получены путем усиления атак–различителей.

Часто в основе структурных атак лежит использование произведения Шура–Адамара [14], [48], [59], [60], [61], [62]. Например, в [59] и [62] используется следующая схема атаки на криптосистему типа Мак–Элиса на основе подкодов кодов Рида–Маллера и Рида–Соломона соответственно. По публичному ключу $\tilde{G}$, основанному на порождающей матрице подкода $D \subset C \subset \mathbb{F}_q^n$, с помощью произведения Шура–Адамара строится матрица $\tilde{G} \star \tilde{G}$. В ряде случаев для указанных кодов с большой вероятностью $D^2 = C^2$. Для кодов Рида–Маллера известно [48], что квадрат кода Рида–Маллера является кодом Рида–Маллера. Аналогичное утверждение справедливо и для кодов Рида–Соломона [60]. Если $C^2 \neq \mathbb{F}_q^n$, то, приводя матрицу $\tilde{G} \star \tilde{G}$ к матрице полного ранга, можно применить к ней известные структурные атаки и таким образом найти подходящий секретный ключ. Другой вариант использования произведения Шура–Адамара для структурной атаки описан в [61]. В этой работе произведение Шура–Адамара используется для поиска случайных столбцов, добавляемых при генерации публичного ключа криптосистемы типа Мак–Элиса. Изначально предполагалось, что добавление таких столбцов усилит стойкость криптосистемы типа Мак–Элиса на кодах Рида–Соломона и кодах Рида–Маллера. В основе атаки из [61] лежит следующее свойство. Для двух $[n,k]_2$-кодов C и D, где C – код Рида–Маллера, а D – случайный, с большой вероятностью выполняется $\dim(C^2) < \dim(D^2)$. Это позволяет отличить случайный код от неслучайного, что как раз используется для поиска случайных столбцов. После нахождения этих столбцов могут применяться известные атаки для вычисления подходящего секретного ключа.

В случае, когда для криптосистемы типа Мак–Элиса неизвестны эффективные атаки на ключ, стойкость системы принято оценивать с помощью вероятности успешного декодирования по информационным совокупностям. Этот подход в настоящее время считается наиболее эффективной атакой на шифрограмму для криптосистем типа Мак–Элиса. В настоящей работе под OW–стойкостью λ_{OW} криптосистемы типа Мак–Элиса на $[n,k,d]_q$–коде здесь и далее понимается отрицательный двоичный логарифм вероятности успеха классического алгоритма ДИС из раздела 1.3.1, или иначе, логарифм сложности этого алгоритма:

$$\lambda_{\mathsf{OW}} = -\log_2\left(\frac{C_{n-t}^k}{C_n^k}\right) = \log_2\left(\frac{C_n^k}{C_{n-t}^k}\right). \tag{1.23}$$

1.5.3 Параметры оригинальной системы Мак–Элиса на кодах Гоппы

Важными характеристиками систем типа Мак–Элиса, помимо OW–стойкости, являются также размер публичного ключа (число байт для хранения матрицы $\tilde{G}$, далее pk_size) и время расшифрования. В таблице 1 приведены характеристики классических асимметричных схем шифрования типа Мак-Элиса на кодах Гоппы: $[n,k]$ – длина и размерность используемого $[n,k,d]_2$–кода Гоппы C; $t \leqslant \lfloor d-1 \rfloor/2$ – вес вектора ошибок, добавляемого в алгоритме Enc на рис. 1.3; λ_{OW} – стойкость, вычисленная в соответствии с (1.23); размер ключа в мегабайтах (МБ) для систем Original McEliece и Classic McEliece; T – время расшифрования (в миллисекундах) информационного сообщения при использовании декодера из [63] (измерение времени выполнения операций во всех экспериментах настоящей работы производилось на компьютере с процессором Intel Core i7-4771 3.50GHz и оперативной памятью 32 ГБ). Стоит отметить, что уровни стойкости криптосистемы Original McEliece, а также параметры $[n,k]$ и t выбраны такими же, как и для криптосистемы Classic

McEliece из работы [64], и соответствуют используемым в конкурсе NIST. При этом размер публичного ключа для Original McEliece определяется размером публичной $(k \times n)$-матрицы, а именно $pk\_size(\mathsf{OriginalMcEliece}) = \frac{kn}{8 \cdot 1024 \cdot 1024}$ МБ, а размер публичного ключа для Classic McEliece определяется размером несистематической части публичной $((n-k) \times n)$-матрицы, представляющей собой $((n-k) \times k)$-матрицу, то есть $pk\_size(\mathsf{ClassicMcEliece}) = \frac{(n-k)k}{8 \cdot 1024 \cdot 1024}$ МБ.

Значения λ_{OW} из таблицы 1 в настоящей работе рассматриваются как *целевые* при разработке схемы шифрования.

Таблица 1 — Характеристики системы Мак–Элиса на кодах Гоппы

Номер уровня стойкости	$[n,k]$	t	λ_{OW}	pk_size, Original McEliece, МБ	pk_size, Classic McEliece, МБ	T
1(mceliece348864)	$[3488, 2720]$	64	142.8	1.13	0.25	20
2(mceliece460896)	$[4608, 3360]$	96	185.92	1.85	0.5	51
3(mceliece6688128)	$[6688, 5024]$	128	262.28	4.01	1	98
4(mceliece6960119)	$[6960, 5413]$	119	265.6	4.49	1	95
5(mceliece8192128)	$[8192, 6528]$	128	302.21	6.38	1.29	120

1.5.4 KEM на основе криптосистемы типа Мак–Элиса

На основе асимметричной криптосистемы $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ типа Мак–Элиса на $[n,k,d]_q$–коде C наиболее стойким механизмом KEM с IND–CCA–стойкостью считается механизм, получаемый с помощью преобразования Фудзисаки–Окамото (FO)[27]. В частности, для всех λ_{OW} из таблицы 1 криптосистема Classic McEliece обеспечивает IND–CCA–стойкость протокола KEM на основании преобразования Фудзисаки–Окамото [64]. Тройка алгоритмов преобразования $\mathsf{FO}^{\not\perp}$ – механизма KEM с неявным отказом (сообщение $\perp$

не возвращается) – показана на рис. 1.4, где $\mathsf{G} : \mathbb{F}_q^k \to \mathcal{E}_{n,t}$ – инъективное отображение, $\mathsf{H} : \mathbb{F}_q^* \to \{0,1\}^m$ – криптографическая хеш–функция.

$\mathsf{Gen}^{\mathsf{KEM}}(t)$:

---

1. $\mathbf{s} \leftarrow \mathbb{F}_q^k$
2. $(pk, sk) \leftarrow \mathsf{Gen}^{\mathrm{McE}(C)}(t)$
3. $pk' = pk$
4. $sk' = (sk, \mathbf{s})$
5. ***вернуть*** (pk', sk')

$\mathsf{Encaps}(pk')$:

---

1. $\mathbf{r} \leftarrow \mathbb{F}_q^k$
2. $\mathbf{c} \leftarrow \mathsf{Enc}(pk', \mathbf{r}; \mathsf{G}(\mathbf{r}))$
3. $\mathbf{K} \leftarrow \mathsf{H}(\mathbf{r} \parallel \mathbf{c})$
4. ***вернуть*** $(\mathbf{K}, \mathbf{c})$

$\mathsf{Decaps}(pk', sk' = (sk, \mathbf{s}), \mathbf{c})$:

---

1. $\mathbf{r}' \leftarrow \mathsf{Dec}(sk, \mathbf{c})$
2. *если* $\mathbf{r}' \neq \perp$
3. $\quad \mathbf{c}' \leftarrow \mathsf{Enc}(pk', \mathsf{r}'; \mathsf{G}(\mathbf{r}'))$
4. $\quad$ *если* $\mathbf{c} = \mathbf{c}'$
5. $\quad\quad$ ***вернуть*** $\mathsf{H}(\mathbf{r}' \parallel \mathbf{c})$
6. ***вернуть*** $\mathsf{H}(\mathbf{s} \parallel \mathbf{c})$

Рисунок 1.4 — Механизм $\mathsf{KEM} = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$, полученный путем применения преобразования $\mathsf{FO}^{\not\perp}$ на основе схемы асимметричного шифрования $\mathrm{McE}(C)$ с детерминированным правилом шифрования

1.6 Выводы

В главе была рассмотрена схема асимметричного шифрования и механизм инкапсуляции сеансового ключа (KEM). Предложено применение кодовой

криптосистемы в схеме KEM. Рассмотрены вопросы построения новых кодовых криптосистем. Для этого приведены основные понятия теории кодирования, используемые в кодовой криптографии, включая произведения Шура–Адамара, понятие группового кода, а также тензорное произведение кодов. В рамках задачи построения эффективных декодеров для применения в кодовой криптосистеме приводится описание концепции мажоритарного декодирования на основе подхода Мэсси. Также рассматривается важный класс кодов Рида–Маллера и некоторые его свойства. В конце главы приводится описание криптосистемы Мак–Элиса и KEM на ее основе, а также приводится классификация атак на кодовые криптосистемы.

Глава 2. D-коды

2.1 Определение и свойства D-кодов

В соответствии с подходом Касами–Лина [65] рассмотрим M–ортогональное семейство линейных кодов

$$\mathcal{S} = \{C(0), C(1), \ldots, C(J)\} \tag{2.1}$$

из пространства $\mathbb{F}_q^n$, для которого выполняется условие:

$\forall j \in [\![1,J]\!]\ \exists i \in [\![0,j-1]\!],\ \exists U_{ij} \subset C(i)|$

s1) $C(0) = \mathbb{F}_q^n, C(J) = \{\bar{0}\}$

s2) $C(i) = \mathcal{L}(U_{ij})$,

s3) $\exists d_{ij} \in \mathbb{N} | \forall \mathbf{u} \in U_{ij}\ (\subset C(i))\ \exists \mathcal{M}_{\mathbf{u}} = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{d_{ij}-1}}\} \subset C(j)$, при этом $d_{ij} \leqslant d(C(i))$.

Следуя [65], с. 600, s3) будем записывать в виде: $C(i) \xleftarrow{d_{ij}} C(j)$.

Приведем альтернативное определение MLD–кода. В соответствии с [65] код $\overline{C(j)}$ называется $l-MLD$–кодом (или MLD–кодом), если существует такая последовательность $j_1 = 0, j_2, \ldots, j_l = j$, что выполняется условие:

$$C(j_k) \xleftarrow{d_{j_k, j_{k+1}}} C(j_{k+1}), \tag{2.2}$$

при этом, если $d(\overline{C(j)}) = \min(d_{j_k, j_{k+1}})$, то мажоритарный декодер полностью реализует минимальное кодовое расстояние кода, т.е. позволяет исправить $\left\lfloor (d(\overline{C(j)}) - 1)/2 \right\rfloor$ ошибок ([39], лемма 1; [65], с. 601). В разделе 2.3.2 это понятие $l - MLD$–кода будет интерпретировано с точки зрения декодирования.

Рассмотрим два семейства кодов

$$\mathcal{S}_1 = \{C_1(0), C_1(1), \ldots, C_1(J_1)\},\ \mathcal{S}_2 = \{C_2(0), C_2(1), \ldots, C_2(J_2)\},$$

из пространств $\mathbb{F}_q^{n_1}$ и $\mathbb{F}_q^{n_2}$ соответственно, удовлетворяющих условиям s1–s3 и, дополнительно, условию

s4) $C_t(0) \supset C_t(1) \supset C_t(2) \supset \cdots \supset C_t(J)$, $\overline{C_t(0)} \subset \overline{C_t(1)} \subset \overline{C_t(2)} \subset \cdots \subset \overline{C_t(J)}$, $t \in \{1,2\}$.

Пусть $D_0 = \{(i,j) | i \in [\![0,J_1]\!], j \in [\![0,J_2]\!]\}$. Для любого $D \subseteq D_0$ определим код длины $n_1 n_2$

$$C(D) = \mathcal{L}\left(\bigcup_{(i,j)\in D} C_1(i) \otimes C_2(j)\right), C_1(i) \in \mathcal{S}_1, C_2(j) \in \mathcal{S}_2. \tag{2.3}$$

Далее будем рассматривать только такие множества $D(\subset D_0)$, которые удовлетворяют условию: если $(i,j) \in D$, $k \geqslant i$, $s \geqslant j$, то $(k,s) \in D$. Набор таких множеств D обозначим через $F(D_0)$. Если $D \in F(D_0)$, то *D-кодом* будем называть код $\overline{C(D)}$, двойственный коду $C(D)$. Отметим, что условия s1–s4 для семейств $\mathcal{S}_1$ и $\mathcal{S}_2$ позволяют для D-кодов построить мажоритарный декодер.

Пример 1. *Рассмотрим бинарные коды Рида–Маллера* $\mathrm{RM}_2(r,m)(\subset \mathbb{F}_2^n)$, *где* $r,m \in \mathbb{N}$, $r \leqslant m$, $n = 2^m$. *Пусть*

$$\mathcal{S}_1 = \{C_1(0) = \mathrm{RM}_2(m_1,m_1), C_1(1) = \mathrm{RM}_2(m_1-1,m_1), \ldots, C_1(m_1) = \mathrm{RM}_2(0,m_1), C_1(m_1+1) = \{\bar{0}\}\},$$

$$\mathcal{S}_2 = \{C_2(0) = \mathrm{RM}_2(m_2,m_2), C_2(1) = \mathrm{RM}_2(m_2-1,m_2), \ldots, C_2(m_2) = \mathrm{RM}_2(0,m_2), C_2(m_2+1) = \{\bar{0}\}\}.$$

Эти семейства удовлетворяют условиям s1–s4. Действительно, условия s1, s4 выполняются в силу определения кода Рида–Маллера, а s2 и s3 фактически проверяются в работе [39], в которой строится набор M–ортогональных систем для кодов Рида–Маллера.

Рассмотрим подмножества D^*, D_b, D_b^* множества D_0, которые определяются по $D \in F(D_0)$:

$$D^* = D_1^* \cup D_2^* \cup D_3^*, \tag{2.4}$$

где

$$D_1^* = \{(i,j)|(i-1,j-1) \in D_0, (i-1,j-1) \notin D\},$$

$$D_2^* = \begin{cases} \{(i,0)|i \in [\![0,\min_{(i,0)\in D}(i)]\!]\}, \text{если } \min_{(i,j)\in D}(j) = 0, \\ \{(i,0)|i \in [\![0,J_1]\!]\}, \text{иначе}, \end{cases}$$

$$D_3^* = \begin{cases} \{(0,j)|j \in [\![0,\min_{(0,j)\in D}(j)]\!]\}, \text{если } \min_{(i,j)\in D}(i) = 0, \\ \{(0,j)|j \in [\![0,J_2]\!]\}, \text{иначе}; \end{cases}$$

$$D_b = \{(i,j)|\ (i,j) \in D \wedge ((i-1,j) \notin D) \wedge ((i,j-1) \notin D)\};$$

$$D_b^* = \{(i,j)|\ (i,j) \in D^* \wedge ((i,j+1) \notin D^*) \wedge ((i+1,j) \notin D^*)\}. \qquad (2.5)$$

Эти подмножества и условие s4 для рассматриваемых семейств, позволяют определить D-код как сумму тензорных произведений кодов, двойственных к кодам из $\mathcal{S}_1$, $\mathcal{S}_2$, а также использовать простую формулу для вычисления размерности D-кода.

Пример 2. *Если* $D = \{(i,j)|\ i+j\ >\ \mu\}$, *то* $D^* = \{(i,j)|\ i+j\ \leqslant\ \mu+2\}$, $D_b = \{(i,j)|\ i+j\ =\ \mu+1\}$, $D_b^* = \{(i,j)|\ i+j\ =\ \mu+2\}$ *(см. [65], с. 604).*

Пример 3. *Если* $D = \{(i,j)|\ i\ \geqslant\ J_1-1\ \vee\ j\ \geqslant\ J_2-1\}$, *то* $D^* = \{(i,j)|\ i\ \leqslant\ J_1-1\ \vee\ j\ \leqslant\ J_2-1\}$, $D_b = \{(J_1-1,\,0),\ (0,\,J_2-1)\}$, $D_b^* = \{(J_1-1,\,J_2-1)\}$ *(см. [65], пример 3).*

Следующие леммы доказаны в [65].

Лемма 2. *Для произвольного* $D \in F(D_0)$*:*

$$\dim(C(D)) = \sum_{(i,j)\in D} k_1(i)k_2(j),$$

где $k_1(i) = \dim(C_1(i)) - \dim(C_1(i+1))$, $k_2(j) = \dim(C_2(j)) - \dim(C_2(j+1))$.

Следствие 1. *Пусть* $\mathcal{S}_1$, $\mathcal{S}_2$ – *семейства кодов Рида–Маллера из примера 1,* $C(D)$ – *код вида* (2.3). *Тогда*

$$\dim(C(D)) = \sum_{(i,j)\in D} C_{m_1}^i C_{m_2}^j.$$

Доказательство. Так как $\mathcal{S}_1$, $\mathcal{S}_2$ – семейства кодов Рида–Маллера, то

$$\begin{aligned} k_t(l) &= \dim(C_t(l)) - \dim(C_t(l+1)) \\ &= \dim(\mathrm{RM}_2(m_t - l, m_t)) - \dim(\mathrm{RM}_2(m_t - l - 1, m_t)) \\ &= \sum_{p=0}^{m_t-l} C_{m_t}^p - \sum_{p=0}^{m_t-l-1} C_{m_t}^p \\ &= C_{m_t}^{m_t-l} + \sum_{p=0}^{m_t-l-1} C_{m_t}^p - \sum_{p=0}^{m_t-l-1} C_{m_t}^p \\ &= C_{m_t}^{m_t-l} = C_{m_t}^l, \end{aligned} \tag{2.6}$$

где $l = i$ для $t = 1$ и $l = j$ для $t = 2$. Поэтому

$$\dim(C(D)) = \sum_{(i,j)\in D} C_{m_1}^i C_{m_2}^j.$$

□

Лемма 3. *Пусть* $\overline{d}_i^t$ – *минимальное кодовое расстояние* $\overline{C_t(i)}$, $t \in \{1,2\}$. *Тогда минимальное кодовое расстояние* d *кода* $\overline{C(D)}$:

$$d = \min_{(i,j)\in D_b} \{\overline{d}_i^1 \overline{d}_j^2\}.$$

Следует отметить, что Э.Л. Блохом и В.В. Зябловым [66] и В.А. Зиновьевым [67] введен класс обобщенных каскадных кодов, который по основным параметрам близок к D-кодам, однако определения класса кодов Блоха–Зяблова–Зиновьева и класса D-кодов существенно отличаются.

Лемма 4. *Для любого* $D \in F(D_0)$ *выполняется:*

$$1)\ C(D) = \mathcal{L}\left(\bigcup_{(i,j)\in D_b} C_1(i) \otimes C_2(j)\right);$$

$$2)\ \overline{C(D)} = \mathcal{L}\left(\bigcup_{(i,j)\in D_b^*} \overline{C_1(i)} \otimes \overline{C_2(j)}\right).$$

Из леммы 4 вытекает, что в примере 3 $\overline{C(D)} = \overline{C_1(J_1-1)} \otimes \overline{C_2(J_2-1)}$ – тензорное произведение кодов.

Лемма 5. *Пусть $\mathcal{S}_1$, $\mathcal{S}_2$ - семейства кодов Рида–Маллера из примера 1, $C(D)$ – код вида* (2.3). *Если $D = \{(i,j)|\ i+j\ >\ \mu\}$, то $C(D)\ =\ \mathrm{RM}_2(m_1+m_2-\mu-1, m_1+m_2)$, $\overline{C(D)}\ =\ \mathrm{RM}_2(\mu, m_1+m_2)$, где $\mu \in [\![0, m_1+m_2-1]\!]$.*

Доказательство. Для заданного D множество $D^* = \{(i,j)|\ i+j\ \leqslant\ \mu+2\}$ по определению (2.4). В соответствии с леммой 4, код $\overline{C(D)}$ представим в виде:

$$\overline{C(D)} = \mathcal{L}\left(\bigcup_{(i,j)\in D_b^*} \overline{\mathrm{RM}_2(m_1-i, m_1)} \otimes \overline{\mathrm{RM}_2(m_2-j, m_2)}\right),$$

где $D_b^* = \{(i,j)|\ i+j\ =\ \mu+2\}$ по определению (2.5).

Из свойства кода, двойственного к коду Рида–Маллера и леммы 1, для любого $(i,j) \in D_b^*$ выполняется

$$\overline{\mathrm{RM}_2(m_1-i, m_1)} \otimes \overline{\mathrm{RM}_2(m_2-j, m_2)} = \mathrm{RM}_2(i-1, m_1) \otimes \mathrm{RM}_2(j-1, m_2)$$

$$\subset \mathrm{RM}_2(i+j-2, m_1+m_2) = \mathrm{RM}_2(\mu, m_1+m_2).$$

Из этого следует, что

$$\overline{C(D)} \subseteq \mathrm{RM}_2(\mu, m_1+m_2). \tag{2.7}$$

Теперь покажем, что

$$\dim(\overline{C(D)}) = \dim(\mathrm{RM}_2(\mu, m_1+m_2)).$$

По лемме 1, формуле (30) из [65], с учетом (2.6) получаем

$$\dim(\overline{C(D)}) = n_1 n_2 - \sum_{(i,j)\in D} k_1(i)k_2(j) = \sum_{(i,j)\notin D} k_1(i)k_2(j)$$

$$= \sum_{i+j\leqslant\mu} C_{m_1}^i C_{m_2}^j = \sum_{\lambda=0}^{\mu}\sum_{i=0}^{\lambda} C_{m_1}^i C_{m_2}^{\lambda-i}.$$

Из тождества Вандермонда следует формула:

$$\sum_{i=0}^{\lambda} C_{m_1}^{i} C_{m_2}^{\lambda-i} = C_{m_1+m_2}^{\lambda}.$$

Следовательно,

$$\dim(\overline{C(D)}) = \sum_{\lambda=0}^{\mu} C_{m_1+m_2}^{\lambda}.$$

С другой стороны

$$\dim(\mathrm{RM}_2(\mu, m_1+m_2)) = \sum_{\lambda=0}^{\mu} C_{m_1+m_2}^{\lambda}.$$

Получаем, что

$$\dim(\overline{C(D)}) = \dim(\mathrm{RM}_2(\mu, m_1+m_2)) \tag{2.8}$$

Из (2.7) и (2.8) следует, что

$$\overline{C(D)} = \mathrm{RM}_2(\mu, m_1+m_2),$$

$$C(D) = \mathrm{RM}_2(m_1+m_2-\mu-1, m_1+m_2).$$

Теорема доказана.

□

2.1.1 Произведение Шура–Адамара D-кодов

В данном разделе исследуются важные криптографические свойства D-кодов, а именно определяются условия разложимости степеней Шура–Адамара D-кодов на основе кодов Рида–Маллера. Эти условия будут использованы в следующей главе для оценки структурной стойкости криптосистемы типа Мак–Элиса на D-кодах. Результаты данного раздела опубликованы в [49], [68].

Рассмотрим семейства кодов Рида–Маллера

$$\mathcal{S}_1 = \{C_1(0) = \mathrm{RM}_2(m_1,m_1), C_1(1) = \mathrm{RM}_2(m_1-1,m_1), ..., C_1(m_1) = \mathrm{RM}_2(0,m_1), C_1(J_1) = \{\bar{0}\}\},$$

$$\mathcal{S}_2 = \{C_2(0) = \mathrm{RM}_2(m_2,m_2), C_2(1) = \mathrm{RM}_2(m_2-1,m_2), ..., C_2(m_2) = \mathrm{RM}_2(0,m_2), C_2(J_2) = \{\bar{0}\}\},$$

где $J_t = m_t + 1$, $t \in \{1,2\}$. Будем считать, что $C_t(i) = C_t(m_t+1)$ для $i \geqslant m_t+1$, $C_t(i) = C_t(0)$ для $i \leqslant 0$, $t = 1,2$. Эти семейства для $t \in \{1,2\}$ удовлетворяют условиям:

$$C_t(0) \supset C_t(1) \supset C_t(2) \supset \cdots \supset C_t(J_t), \tag{2.9}$$

$$\overline{C_t(0)} \subset \overline{C_t(1)} \subset \overline{C_t(2)} \subset \cdots \subset \overline{C_t(J_t)}. \tag{2.10}$$

Поэтому

$$C_1(i_1) \otimes C_2(j_1) \supseteq C_1(i_2) \otimes C_2(j_2), \tag{2.11}$$
$$\overline{C_1(i_1)} \otimes \overline{C_2(j_1)} \subseteq \overline{C_1(i_2)} \otimes \overline{C_2(j_2)},$$

для $i_1 \leqslant i_2, j_1 \leqslant j_2$.

Далее нам понадобится следующая техническая лемма.

Лемма 6. *Пусть* $D_b^* = \{(k_1,\ell_1),(k_2,\ell_2),...,(k_s,\ell_s)\}$ *для некоторого* $D \in F(D_0)$, *где* $s \leqslant \min\{m_1,m_2\}+1$, $k_i \in [\![0,m_1+1]\!]$, $\ell_i \in [\![0,m_2+1]\!]$. *Последовательность* $(k_i)_{i=1}^s$ – *возрастающая тогда и только тогда, когда* $(\ell_i)_{i=1}^s$ – *убывающая последовательность.*

Доказательство. Пусть $(k_i)_{i=1}^s$ – возрастающая последовательность. Предположим, что последовательность $(\ell_i)_{i=1}^s$ при этом не является убывающей. Это означает, что для $j > i$ существуют такие точки $(k_i,\ell_i), (k_j,\ell_j) \in D_b^*$, что $k_j > k_i$ и $\ell_j \geqslant \ell_i$.

По определению множества D_b^* имеем: $(k_i,\ell_i), (k_j,\ell_j) \in D^*$, при этом $(k_i+1,\ell_i) \notin D^*$, где $k_i+1 \leqslant k_j$. По определению множества D^*, если $(k_j,\ell_j) \in D^*$, то $(k_j-1,\ell_j-1) \notin D$, а так как $D \in F(D_0)$, то $(k_p,\ell_p) \notin D$ для любых $k_p \leqslant k_j-1$,

$\ell_p \leqslant \ell_j - 1$. Это означает, что $(k_p, \ell_p) \in D^*$ для любых $k_p \leqslant k_j$, $\ell_p \leqslant \ell_j$ по определению D^*.

Поэтому если $(k_j,\ell_j) \in D^*$, $k_j \geqslant k_i + 1$, $\ell_j \geqslant \ell_i$, то и $(k_i + 1, \ell_i) \in D^*$. Приходим к противоречию, значит $(\ell_i)_{i=1}^s$ – убывающая последовательность. Аналогично утверждение доказывается в обратную сторону. □

Пусть $D_b^* = \{(k_1,\ell_1),(k_2,\ell_2),...,(k_s,\ell_s)\}$, где $s \leqslant \min(|\mathcal{S}_1|,|\mathcal{S}_2|)$, $k_i \in [\![0,m_1 + 1]\!]$, $\ell_i \in [\![0,m_2 + 1]\!]$, $(k_i)_{i=1}^s$ – возрастающая последовательность. Тогда по лемме 6 $(\ell_i)_{i=1}^s$ – убывающая.

Поэтому в соответствии с леммой 4

$$\overline{C(D)} = \sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(\ell_i)}. \tag{2.12}$$

Для $s \geqslant 2$ и $i < s$ во введенных обозначениях получаем

$$\overline{C_1(k_i)} \subset \overline{C_1(k_{i+1})}, \overline{C_2(\ell_{i+1})} \subset \overline{C_2(\ell_i)}. \tag{2.13}$$

Пусть $r_{k_i}^1 = m_1 - k_i$ – порядок кода Рида–Маллера $C_1(k_i) = \mathrm{RM}_2(r_{k_i}^1, m_1)$, $r_{\ell_i}^2 = m_2 - \ell_i$ – порядок кода Рида–Маллера $C_2(\ell_i) = \mathrm{RM}_2(r_{\ell_i}^2,m_2)$, а

$$\overline{r}_{k_i}^1 = m_1 - (m_1 - k_i) - 1 = k_i - 1, \overline{r}_{\ell_i}^2 = m_2 - (m_2 - \ell_i) - 1 = \ell_i - 1 \tag{2.14}$$

– порядки двойственных кодов $\overline{C_1(k_i)}$ и $\overline{C_2(\ell_i)}$ соответственно.

Тогда с учетом (1.16), (2.12) и (2.14) получаем

$$\begin{aligned}
\overline{C(D)}^2 &= \left(\sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(\ell_i)}\right)^2 \\
&= \sum_{i=1}^{s} \overline{C_1(k_i)}^2 \otimes \overline{C_2(\ell_i)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} (\overline{C_1(k_p)} \star \overline{C_1(k_j)}) \otimes (\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)}) \quad (2.15) \\
&= \sum_{i=1}^{s} \mathrm{RM}_2(2\overline{r}^1_{k_i}, m_1) \otimes \mathrm{RM}_2(2\overline{r}^2_{\ell_i}, m_2) \\
&+ \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \mathrm{RM}_2(\overline{r}^1_{k_p} + \overline{r}^1_{k_j}, m_1) \otimes \mathrm{RM}_2(\overline{r}^2_{\ell_p} + \overline{r}^2_{\ell_j}, m_2) \\
&= \sum_{i=1}^{s} \mathrm{RM}_2(2k_i - 2, m_1) \otimes \mathrm{RM}_2(2\ell_i - 2, m_2) \\
&+ \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \mathrm{RM}_2(k_p + k_j - 2, m_1) \otimes \mathrm{RM}_2(\ell_p + \ell_j - 2, m_2) \\
&= \sum_{i=1}^{s} \overline{C_1(2k_i - 1)} \otimes \overline{C_2(2\ell_i - 1)} \quad (2.16) \\
&+ \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_1(k_p + k_j - 1)} \otimes \overline{C_2(\ell_p + \ell_j - 1)}.
\end{aligned}$$

Таким образом, код $\overline{C(D)}^2$ представляет собой сумму тензорных произведений двойственных кодов к кодам из семейств $\mathcal{S}_1$, $\mathcal{S}_2$.

Отметим, что в общем случае для произвольного кода неизвестна формула для нахождения размерности квадрата этого кода. Но в случае кода $\overline{C(D)}$ эту формулу можно получить. Перепишем (2.16) в виде

$$\begin{aligned}
\overline{C(D)}^2 = &\sum_{i=1}^{s} C_1(m_1 - 2k_i) \otimes C_2(m_2 - 2\ell_i) \\
&+ \sum_{p\neq j} C_1(m_1 - (k_p + k_j)) \otimes C_2(m_2 - (\ell_p + \ell_j))
\end{aligned}$$

Пусть $X = \{(m_1 - 2k_i, m_2 - 2\ell_i) | i \in [\![1,s]\!]\} \bigcup \{(m_1 - (k_p + k_j), m_2 - (\ell_p + \ell_j)) | p,j \in [\![1,s]\!], p \neq j\}$ – множество точек из D_0, которое соответствует коду $\overline{C(D)}^2$. Построим множество $D' \subseteq D_0$ следующим образом. Для каждой точки

$(i,j) \in X$ добавим в D' все точки $(k,l) \in D_0$ такие, что $k \geqslant i$ и $l \geqslant j$. Множеству D' соответствует код $C(D')$ вида (2.3), при этом из (2.11), учитывая способ построения D', получаем

$$C(D') = \overline{C(D)}^2.$$

Отметим, что $D' \in F(D_0)$ по построению. Поэтому, согласно 1, размерность кода $\overline{C(D)}^2$ вычисляется по формуле

$$\dim(\overline{C(D)}^2) = \dim(C(D')) = \sum_{(i,j)\in D'} C^i_{m_1} C^j_{m_2}. \tag{2.17}$$

Рассмотрим случаи, при которых удается упростить представление (2.15) и сделать вывод о разложимости кода $\overline{C(D)}^2$ в прямую сумму неразложимых кодов.

Теорема 1. *Пусть $\overline{C(D)}$ код вида* (2.12).

1) Если $\overline{r}^1_{k_1} \geqslant m_1/2$ и $\overline{r}^2_{\ell_1} < m_2/2$, то

$$\overline{C(D)}^2 = \mathbb{F}_2^{n_1} \otimes \overline{C_2(\ell_1)}^2.$$

2) Если $\overline{r}^1_{k_s} < m_1/2$ и $\overline{r}^2_{\ell_s} \geqslant m_2/2$, то

$$\overline{C(D)}^2 = \overline{C_1(k_s)}^2 \otimes \mathbb{F}_2^{n_2}.$$

Доказательство. Докажем первое утверждение. Согласно (1.16), если $\overline{r}^1_{k_1} \geqslant m_1/2$ и $\overline{r}^2_{\ell_1} < m_2/2$, то $\overline{C_1(k_1)}^2 = \mathbb{F}_2^{n_1}$ и $\overline{C_2(\ell_1)}^2 \neq \mathbb{F}_2^{n_2}$. Так как $(k_i)_{i=1}^s$ – возрастающая последовательность, то из леммы 6 и (2.14) последовательность $(\overline{r}^1_{k_i})_{i=1}^s$ – возрастающая, а $(\overline{r}^2_{\ell_i})_{i=1}^s$ – убывающая. Поэтому из (2.15) следует

$$\begin{aligned}
\overline{C(D)}^2 &= \sum_{i=1}^{s} \mathbb{F}_2^{n_1} \otimes \overline{C_2(\ell_i)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \mathbb{F}_2^{n_1} \otimes (\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)}) \\
&= \mathbb{F}_2^{n_1} \otimes \left(\sum_{i=1}^{s} \overline{C_2(\ell_i)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)} \right) \\
&= \mathbb{F}_2^{n_1} \otimes \overline{C_2(\ell_1)}^2,
\end{aligned}$$

где последнее равенство вытекает из вложений $\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)} \subseteq \overline{C_2(\ell_1)}^2$ и $\overline{C_2(\ell_i)}^2 \subseteq \overline{C_2(\ell_1)}^2$, которые следуют из (1.16) и (2.13).

Докажем теперь второе утверждение. Согласно (1.16), если $\overline{r}^1_{k_s} < m_1/2$ и $\overline{r}^2_{\ell_s} \geqslant m_2/2$, то $\overline{C_1(k_s)}^2 \neq \mathbb{F}_2^{n_1}$ и $\overline{C_2(\ell_s)}^2 = \mathbb{F}_2^{n_2}$. Поэтому из (2.15) следует

$$\begin{aligned}\overline{C(D)}^2 &= \sum_{i=1}^{s} \overline{C_1(k_i)}^2 \otimes \mathbb{F}_2^{n_2} + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} (\overline{C_1(k_p)} \star \overline{C_1(k_j)}) \otimes \mathbb{F}_2^{n_2} \\ &= \left(\sum_{i=1}^{s} \overline{C_1(k_i)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_1(k_p)} \star \overline{C_1(k_j)} \right) \otimes \mathbb{F}_2^{n_2} \\ &= \overline{C_1(k_s)}^2 \otimes \mathbb{F}_2^{n_2}.\end{aligned}$$

Утверждение доказано. □

Согласно [69] двоичные коды Рида–Маллера $\mathrm{RM}_2(r,m)$ для $r < m$ являются неразложимыми. Так как $\overline{C_2(\ell_1)}^2$ – код Рида–Маллера, не совпадающий со всем пространством при выполнении условий первого утверждения теоремы 1, то, как следует из (1.12), код $\overline{C(D)}^2$ раскладывается в прямую сумму неразложимых кодов. При выполнении условий второго утверждения теоремы 1 код $\overline{C_1(k_s)}^2$ также является кодом Рида–Маллера, не совпадающим со всем пространством. Согласно [70] (см. формулу (10)) для любых двух матриц A и B найдутся такие перестановочные матрицы P и Q подходящих размеров, что $A \otimes B = P(B \otimes A)Q$. Тогда найдется такая перестановка $\sigma \in \mathcal{P}_{n_1 n_2}$, что $\sigma(\overline{C_1(k_s)}^2 \otimes \mathbb{F}_2^{n_2}) = \mathbb{F}_2^{n_2} \otimes \overline{C_1(k_s)}^2$. Следовательно, в этом случае код $\overline{C(D)}^2$ является перестановочно эквивалентным прямой сумме неразложимых кодов.

Теорема 2. *Пусть $\overline{C(D)}$ код вида* (2.12). *Если выполняется хотя бы одно из условий:*

1) существует такой $i \in \{1,...,s\}$, что $\overline{r}^1_{k_i} \geqslant m_1/2$ и $\overline{r}^2_{\ell_i} \geqslant m_2/2$,

2) существуют такие $p \in \{1,...,s\}$, $j \in \{1,...,s\}$, $p \neq j$, что выполняются неравенства $\overline{r}^1_{k_p} + \overline{r}^1_{k_j} \geqslant m_1$ и $\overline{r}^2_{\ell_p} + \overline{r}^2_{\ell_j} \geqslant m_2$,

то

$$\overline{C(D)}^2 = \mathbb{F}_2^{n_1 n_2}.$$

Доказательство. Пусть выполняется первое условие, тогда из (1.16) следует, что $\overline{C_1(k_i)}^2 = \mathbb{F}_2^{n_1}$ и $\overline{C_2(\ell_i)}^2 = \mathbb{F}_2^{n_2}$. Значит одно из слагаемых в (2.15) будет иметь вид $\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}$. В этом случае $\overline{C(D)}^2 = \mathbb{F}_2^{n_1 n_2}$.

Пусть выполняется второе условие, тогда из (1.16) следует, что $\overline{C_1(k_p)} \star \overline{C_1(k_j)} = \mathbb{F}_2^{n_1}$ и $\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)} = \mathbb{F}_2^{n_2}$. Таким образом, одно из слагаемых в (2.15) будет иметь вид $\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}$, поэтому $\overline{C(D)}^2 = \mathbb{F}_2^{n_1 n_2}$. □

Отметим, что при выполнении условий теоремы 2 код $\overline{C(D)}^2$ совпадает с $\mathbb{F}_2^{n_1 n_2}$ и, поэтому, не эквивалентен прямой сумме нетривиальных кодов Рида–Маллера.

Теорема 3. *Пусть $\overline{C(D)}$ код вида (2.12) и выполняются условия $\overline{r}^1_{k_1} < m_1/2$, $\overline{r}^2_{\ell_1} \geqslant m_2/2$, и $\overline{r}^1_{k_j} \geqslant m_1/2$, $\overline{r}^2_{\ell_j} < m_2/2$ для любых $j \geqslant 2$. Если для любых $p \in \{1,...,s\}$, $j \in \{1,...,s\}$, $p \neq j$ выполняются неравенства $\overline{r}^1_{k_p} + \overline{r}^1_{k_j} \geqslant m_1$ и $\overline{r}^2_{\ell_p} + \overline{r}^2_{\ell_j} < m_2$, то*

$$\overline{C(D)}^2 = \overline{\tilde{C}_1 \otimes \tilde{C}_2},\ \overline{C(D)}^3 = \mathbb{F}_2^{n_1 n_2},$$

где $\tilde{C}_1 = \overline{\overline{C_1(k_1)}^2}$, $\tilde{C}_2 = \overline{\overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p \neq j}}^{s} \overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)}}$.

Доказательство. При выполнении условий теоремы, согласно (1.16), формула (2.15) примет вид

$$\overline{C(D)}^2 = \overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2} + \mathbb{F}_2^{n_1} \otimes \overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p \neq j}}^{s} (\overline{C_1(k_p)} \star \overline{C_1(k_j)}) \otimes (\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)}) \tag{2.18}$$

$$= \overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2} + \mathbb{F}_2^{n_1} \otimes \overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p \neq j}}^{s} \mathbb{F}_2^{n_1} \otimes (\overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)})$$

$$= \overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2} + \mathbb{F}_2^{n_1} \otimes (\overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p \neq j}}^{s} \overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)}).$$

Согласно (1.10) во введенных в условии теоремы обозначениях получаем доказываемое утверждение. При этом,

$$\overline{C(D)}^3 = (\overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2} + \mathbb{F}_2^{n_1} \otimes (\overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)})) \star \overline{C(D)}$$

$$= (\overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2}) \star (\sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(\ell_i)})$$

$$+ (\mathbb{F}_2^{n_1} \otimes (\overline{C_2(\ell_2)}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_2(\ell_p)} \star \overline{C_2(\ell_j)})) \star (\sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(\ell_i)}).$$

Рассмотрим первое слагаемое:

$$(\overline{C_1(k_1)}^2 \otimes \mathbb{F}_2^{n_2}) \star (\sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(\ell_i)}) = (\overline{C_1(k_1)}^2 \star \overline{C_1(k_s)}) \otimes \mathbb{F}_2^{n_2} = \mathbb{F}_2^{n_1 n_2},$$

где последнее равенство вытекает из $\overline{r}^1_{k_p} + \overline{r}^1_{k_j} \geqslant m_1$. Тогда $\overline{C(D)}^3 = \mathbb{F}_2^{n_1 n_2}$. □

Следующая теорема доказывается аналогично.

Теорема 4. *Пусть* $\overline{C(D)}$ *код вида* (2.12) *и выполняются условия* $\overline{r}^1_{k_j} < m_1/2$, $\overline{r}^2_{\ell_j} \geqslant m_2/2$ *для любых* $j < s$ *и* $\overline{r}^1_{k_s} \geqslant m_1/2$, $\overline{r}^2_{\ell_s} < m_2/2$. *Если для любых* $p \in \{1,...,s\}$, $j \in \{1,...,s\}$, $p \neq j$ *выполняются неравенства* $\overline{r}^1_{k_p} + \overline{r}^1_{k_j} < m_1$ *и* $\overline{r}^2_{\ell_p} + \overline{r}^2_{\ell_j} \geqslant m_2$, *то*

$$\overline{C(D)}^2 = \overline{\hat{C}_1 \otimes \hat{C}_2},\ \overline{C(D)}^3 = \mathbb{F}_2^{n_1 n_2},$$

где $\hat{C}_1 = \overline{\overline{C_1(k_{s-1})}^2 + \sum_{\substack{p,j=1 \\ p\neq j}}^{s} \overline{C_1(k_p)} \star \overline{C_1(k_j)}}$, $\hat{C}_2 = \overline{C_2(\ell_s)}^2$.

Отметим, что в теоремах 3 и 4 коды $\tilde{C}_1$, $\tilde{C}_2$, $\hat{C}_1$, $\hat{C}_2$ – коды Рида–Маллера. Поэтому при выполнении условий теорем 3, 4 код $\overline{C(D)}^2$ является неразложимым и не эквивалентен прямой сумме нетривиальных кодов Рида–Маллера.

В случаях, не рассмотренных в теоремах 1, 2, 3, 4, представление (2.15) пока не удается упростить и сделать выводы о разложимости кода $\overline{C(D)}^2$ в прямую сумму кодов Рида–Маллера. Но можно вычислить размерность кода $\overline{C(D)}^2$ с помощью (2.17) и оценить возможность разложения этого кода в прямую сумму одинаковых кодов Рида–Маллера. Если код $\overline{C(D)}^2$ раскладывается в прямую

сумму 2^{m_1} (2^{m_2}) одинаковых кодов Рида–Маллера K длины 2^{m_2} (2^{m_1}), то, согласно (1.12), его размерность равна $2^{m_1}\dim(K)$ ($2^{m_2}\dim(K)$). Поэтому если размерность кода $\overline{C(D)}^2$ не делится на 2^{m_1} (2^{m_2}), то этот код не раскладывается в прямую сумму 2^{m_1} (2^{m_2}) одинаковых кодов Рида–Маллера и не является перестановочно эквивалентным такому коду.

2.2 Групповые D-коды

В качестве M–ортогонального семейства (2.1) можно рассматривать семейство групповых кодов. Пусть $\mathcal{G} = \{g_1 = \hat{1}, \ldots, g_{|\mathcal{G}|}\}$ – конечная группа с зафиксированным линейным порядком на множестве ее элементов. Рассмотрим групповую алгебру $\mathbb{F}_q\mathcal{G}$. Тогда семейство $\mathcal{S} = \{C(0), C(1), \ldots, C(J)\}, C(i) \subset \mathbb{F}_q\mathcal{G}$, удовлетворяющее условиям s1–s4, представляет собой M–ортогональное семейство групповых кодов. Примером такого семейства является семейство кодов Рида–Маллера–Бермана [39]. Как отмечалось в разделе 1.3.2, поскольку групповой код – левый идеал, действие группы $\mathcal{G}$ на $\mathbb{F}_q\mathcal{G}$ по правилу (1.7) индуцирует действие этой группы на групповом коде. С другой стороны, группа $\mathcal{G}$ действует транзитивно на элементах из $B^{\mathbb{F}_q\mathcal{G}}$ и не нарушает M-ортогональности. Это позволяет построить набор декодирующих графов для всех координат только по одному такому графу. Отметим, что в разделе 2.3.2 рассматривается декодирование только одной выбранной координаты по соответствующему этой координате декодирующему графу, и в общем случае построение декодирующего графа для каждой отдельной координаты – независимая задача. Но в случае групповых кодов, имея декодирующий граф для одной координаты, можно построить декодирующие графы для каждой координаты благодаря алгебраической структуре групповых кодов.

Пусть $\mathcal{G}$, $\mathcal{H}$ – конечные группы мощности n_1 и n_2 соответственно, а $\mathbb{F}_q\mathcal{G}$, $\mathbb{F}_q\mathcal{H}$ – их групповые алгебры. Рассмотрим два M–ортогональных семейства

групповых кодов

$$\mathcal{S}_1 = \{C_1(0), C_1(1), \ \ldots, C_1(J)\}, C_1(i) \subset \mathbb{F}_q\mathcal{G}, \tag{2.19}$$

$$\mathcal{S}_2 = \{C_2(0), C_2(1), \ \ldots, C_2(J)\}, C_2(i) \subset \mathbb{F}_q\mathcal{H}.$$

Пусть $C(D) \subset \mathbb{F}_q(\mathcal{G} \times \mathcal{H})$ – код длины $n_1 n_2$, построенный на основе этих семейств (см. (2.3), леммы 2, 3). Тогда двойственный к нему код $\overline{C(D)}$ называется групповым D-кодом. При решении задачи построения мажоритарного декодера будем рассматривать только групповые D-коды.

Отметим, что тензорное произведение групповых MLD–кодов входит в класс групповых D-кодов, и разработанные ниже алгоритмы подходят для их декодирования.

2.3 Декодирование D-кодов

Далее для применения в схеме Мак–Элиса исследуется вопрос эффективного декодирования D-кодов. Как было сказано ранее, подходы к декодированию линейных кодов условно можно разделить на гарантированное и вероятностное декодирование. В данной работе реализованы оба подхода для декодирования D-кодов. Именно, в данном разделе строится алгоритмическая модель гарантированного декодирования D-кодов на основе мажоритарного подхода Мэсси [46], а также два алгоритма вероятностного декодирования D-кодов на основе кодов Рида–Маллера, позволяющих декодировать ошибки в количестве, превышающем половину кодового расстояния.

2.3.1 Конструкция кодера

Для начала определим конструкцию кодера для D-кодов. Рассмотрим два семейства групповых кодов $\mathcal{S}_1$, $\mathcal{S}_2$ (см. (2.19)), которые удовлетворяют условиям s1–s4, и множество $D(\subset D_0)$. Пусть по множеству D построены соответствующие множества D^*, D_b, D_b^* из раздела 2.2. Тогда в соответствии с леммой 4,

$$\overline{C(D)} = \mathcal{L}\left(\bigcup_{(i,j)\in D_b^*} \overline{C_1(i)} \otimes \overline{C_2(j)}\right). \tag{2.20}$$

Пусть $G_{\overline{C_t(i)}}$ – порождающая матрица кода $\overline{C_t(i)}$, двойственного коду $C_t(i)$ $(\in \mathcal{S}_t), t \in \{1,2\}$. Зафиксируем линейный порядок

$$(i_1,j_1),(i_2,j_2),\ldots,(i_{|D_b^*|},j_{|D_b^*|})$$

на множестве D_b^* и построим матрицу

$$\tilde{G} = \begin{pmatrix} G_{\overline{C_1(i_1)}} \otimes G_{\overline{C_2(j_1)}} \\ G_{\overline{C_1(i_2)}} \otimes G_{\overline{C_2(j_2)}} \\ G_{\overline{C_1(i_3)}} \otimes G_{\overline{C_2(j_3)}} \\ \cdots \\ G_{\overline{C_1(i_{|D_b^*|})}} \otimes G_{\overline{C_2(j_{|D_b^*|})}} \end{pmatrix}.$$

Пусть $G_{\overline{C(D)}}$ – такая $(k \times n)$–матрица полного ранга, что $\mathcal{L}(G_{\overline{C(D)}}) = \mathcal{L}(\tilde{G})$. Матрица $G_{\overline{C(D)}}$ может быть получена приведением $\tilde{G}$ к матрице полного ранга. Тогда $G_{\overline{C(D)}}$ – порождающая матрица кода $\overline{C(D)}$, где длина кода $n = n_1 n_2$, а размерность, в соответствии с леммой 2, равна

$$k = n_1 n_2 - \dim(C(D)) \;=\; n_1 n_2 - \sum_{(i,j)\in D} k_1(i) k_2(j).$$

Процедура кодирования информационного сообщения a $(\in \mathbb{F}_q^k)$ заключается в умножении его справа на матрицу $G_{\overline{C(D)}}$.

2.3.2 Гарантированное декодирование

Далее решается вопрос гарантированного декодирования D-кодов. Пусть $\mathbf{c}(\in \overline{C(D)})$ – кодовый вектор, который на выходе из канала принимает вид

$$\mathbf{x} = \mathbf{c} + \mathbf{e}, \;\; \mathbf{e} \in \mathbb{F}_q(\mathcal{G} \times \mathcal{H}). \tag{2.21}$$

Ниже строится алгоритмическая модель, позволяющая находить значение вектора ошибок $\mathbf{e}$, при условии, что $\mathrm{wt}(\mathbf{e}) \leqslant \lfloor (d-1)/2 \rfloor$, где $d = d(\overline{C(D)})$ – минимальное кодовое расстояние кода $\overline{C(D)}$, определяемое в соответствии с леммой 3. При этом задача гарантированного декодирования D-кодов решается в несколько этапов. Сначала строятся алгоритмы декодирования тензорного произведения кодов, основанные на использовании декодирующих деревьев, описанных в разделе 1.3.2. Далее строятся алгоритмы декодирования D-кодов, использующие улучшенную конструкцию декодирующих графов и включающие алгоритм построения по декодирующим графам $G(\mathcal{S}_1, \mathbf{b}^1)$, $G(\mathcal{S}_2, \mathbf{b}^2)$ для двух семейств MLD–кодов декодирующего графа $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ для D-кода, алгоритм декодирования координаты, соответствующей $\mathbf{b}^1 \otimes \mathbf{b}^2$, алгоритм построения набора декодирующих графов для группового D-кода и алгоритм декодирования по этому набору всех координат зашумленного кодового слова. Результаты, полученные в данном разделе, изложены в работах [71-75].

Декодер для тензорного произведения

В данном разделе решается задача мажоритарного декодирования тензорного произведения двух групповых MLD–кодов. Построенная алгоритмическая модель программно реализована на языке C++ и доступна в [58] (см. `https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/`

Tensor%20ML%20Decoding). Вместе с программой содержится текстовый документ с примерами работы. Пусть $\mathbf{c}(\in C_1 \otimes C_2)$ – кодовый вектор вида (2.21), где для вектора ошибок $\mathbf{e}$ выполняется условие

$$\mathrm{wt}(\mathbf{e}) \leqslant \lfloor (d(C_1 \otimes C_2) - 1)/2 \rfloor . \tag{2.22}$$

Прежде всего, сформулируем вспомогательную лемму.

Лемма 7. *Пусть* $\mathbf{c}_1 \in \mathbb{F}_q^{n_1}$, $\mathcal{M}_{\mathbf{c}_1}$ *– M–ортогональное множество для вектора* $\mathbf{c}_1$, $\mathbf{c}_2 \in \mathbb{F}_q^{n_2}$, $\mathcal{M}_{\mathbf{c}_2}$ *– M–ортогональное множество для вектора* $\mathbf{c}_2$, *тогда M-ортогональное множество для вектора* $\mathbf{c}_1 \otimes \mathbf{c}_2$ *состоит из векторов вида:*

1. $\mathbf{c}_1 \otimes \mathbf{w}$,
2. $\mathbf{v} \otimes \mathbf{c}_2$,
3. $\mathbf{c}_1 \otimes \mathbf{w} + \mathbf{v} \otimes \mathbf{c}_2 + \mathbf{v} \otimes \mathbf{w}$,

где $\mathbf{v} \in \mathcal{M}_{\mathbf{c}_1}$, $\mathbf{w} \in \mathcal{M}_{\mathbf{c}_2}$.

Доказательство. Эта лемма вытекает из [37], с.121–122. В [37] конструкция леммы 7 имеет другой вид. Согласно свойству M–ортогональности, вектора $\mathbf{v}$,$\mathbf{w}$ имеют вид:

$$\mathbf{v} = \mathbf{c_1} + \mathbf{v}', \mathbf{w} = \mathbf{c_2} + \mathbf{w}'.$$

Тогда в [37] конструкция 3 имеет вид

$$\mathbf{c}_1 \otimes \mathbf{c}_2 + \mathbf{v}' \otimes \mathbf{w}'. \tag{2.23}$$

Покажем, что конструкция 3 настоящей леммы может быть представлена в виде (2.23):

$$\begin{aligned}
&\mathbf{c}_1 \otimes \mathbf{w} + \mathbf{v} \otimes \mathbf{c}_2 + \mathbf{v} \otimes \mathbf{w} = \\
&\quad = \mathbf{c}_1 \otimes (\mathbf{c}_2 + \mathbf{w}') + (\mathbf{c}_1 + \mathbf{v}') \otimes \mathbf{c}_2 + (\mathbf{c}_1 + \mathbf{v}') \otimes (\mathbf{c}_2 + \mathbf{w}') = \\
&\quad = \mathbf{c}_1 \otimes \mathbf{c}_2 + \mathbf{c}_1 \otimes \mathbf{w}' + \mathbf{c}_1 \otimes \mathbf{c}_2 + \mathbf{v}' \otimes \mathbf{c}_2 + \\
&\quad + \mathbf{c}_1 \otimes \mathbf{c}_2 + \mathbf{v}' \otimes \mathbf{c}_2 + \mathbf{c}_1 \otimes \mathbf{w}' + \mathbf{v}' \otimes \mathbf{w}' = \\
&\quad = \mathbf{c}_1 \otimes \mathbf{c}_2 + \mathbf{v}' \otimes \mathbf{w}'.
\end{aligned}$$

□

Эта лемма позволяет по двум векторам $\mathbf{c}_1$, $\mathbf{c}_2$ и их M–ортогональным множествам построить M–ортогональное множество для вектора $\mathbf{c}_1 \otimes \mathbf{c}_2$.

Для векторов $\mathbf{c}_1 \in C_1(\subseteq \mathbb{F}_q\mathcal{G})$ и $\mathbf{c}_2 \in C_2(\in\subseteq \mathbb{F}_q\mathcal{H})$ рассмотрим соответствующие им M-ортогональные множества $\mathcal{M}_{\mathbf{c}_1}$ и $\mathcal{M}_{\mathbf{c}_2}$.В соответствии с леммой построим алгоритм M_orth, который конструирует для каждого вектора $\mathbf{c}_1 \otimes \mathbf{c}_2 \in C_1 \otimes C_2(\subseteq \mathbb{F}_q(\mathcal{G} \times \mathcal{H}))$ M–ортогональное множество $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$, соответствующее элементам первого и второго типа леммы 7. Заметим, что

$$|\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}| = |\mathcal{M}_{\mathbf{c}_1}| + |\mathcal{M}_{\mathbf{c}_1}|. \tag{2.24}$$

Алгоритм 2 M_orth

Исходные параметры: Векторы $\mathbf{c}_1$, $\mathbf{c}_2$ и соответствующие им $\mathcal{M}$-ортогональные множества $\mathcal{M}_{\mathbf{c}_1}$, $\mathcal{M}_{\mathbf{c}_2}$.
Результат: $\mathcal{M}$-ортогональное множество $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$ для вектора $\mathbf{c}_1 \otimes \mathbf{c}_2$.

1: $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2} := \varnothing$
2: **для каждого** $\mathbf{w} \in \mathcal{M}_{\mathbf{c}_2}$ **выполнять**
3: к $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$ добавить $\mathbf{c}_1 \otimes \mathbf{w}$
4: **для каждого** $\mathbf{v} \in \mathcal{M}_{\mathbf{c}_1}$ **выполнять**
5: к $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$ добавить $\mathbf{v} \otimes \mathbf{c}_2$
6: **вернуть** $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$

Далее построим алгоритм MakeTensorTree, который для корня с меткой $\mathbf{1}_\mathcal{G} \otimes \mathbf{1}_\mathcal{H}$ по декодирующим деревьям $\mathrm{WB}_{\mathbf{1}_\mathcal{G}, r_{\mathbf{1}_\mathcal{G}}, L_{\mathbf{1}_\mathcal{G}}}[C_1]$ и $\mathrm{WB}_{\mathbf{1}_\mathcal{H}, r_{\mathbf{1}_\mathcal{H}}, L_{\mathbf{1}_\mathcal{H}}}[C_2]$ строит *вспомогательное декодирующее дерево*, которое обозначим следующим образом

$$\mathrm{WB}^{\otimes}{}_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}, r_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}}, L_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}}}[C_1 \otimes C_2]. \tag{2.25}$$

В алгоритме MakeTensorTree сначала для корней декодирующих деревьев для кодов C_1, C_2 с помощью алгоритма M_orth строится M–ортогональное множество, составляющее первый уровень дерева (2.25). Далее на каждом уровне k, для каждой вершины $\mathbf{v}^k = (\mathbf{c}_1^k \otimes \mathbf{c}_2^k)$ строится M–ортогональное множество с помощью вершин $\mathbf{c}_1^k, \mathbf{c}_2^k$, соответствующих декодирующих деревьев и их M–ортогональных множеств. Полученное множество добавляется на $k+1$ уровень дерева.

Алгоритм 3 MakeTensorTree

Исходные параметры: $\mathrm{WB}_{\mathbf{1}_\mathcal{G},r_{\mathbf{1}_\mathcal{G}},L_{\mathbf{1}_\mathcal{G}}}[C_1]$, $\mathrm{WB}_{\mathbf{1}_\mathcal{H},r_{\mathbf{1}_\mathcal{H}},L_{\mathbf{1}_\mathcal{H}}}[C_2]$.
Результат: дерево вида (2.25).

1: $V_1^{\otimes} := \mathsf{M\_orth}(\mathbf{1}_\mathcal{G},\mathbf{1}_\mathcal{H},\mathcal{M}_{\mathbf{1}_\mathcal{G}},\mathcal{M}_{\mathbf{1}_\mathcal{H}})$
2: **цикл** $1 \leqslant k \leqslant L_{\mathbf{1}_\mathcal{G}} + L_{\mathbf{1}_\mathcal{H}} - 2$ **выполнять**
3: **для каждого** $\mathbf{v}^k = (\mathbf{c}_1^k \otimes \mathbf{c}_2^k) \in V_k^{\otimes}$ **выполнять**
4: **если** $\mathbf{v}^k \notin \overline{(C_1 \otimes C_2)}$ **то**
5: на уровень $V_{k+1}^{\otimes}$ добавить $|\mathcal{M}_{\mathbf{c}_1^k}| + |\mathcal{M}_{\mathbf{c}_2^k}|$ вершин и соединить их с вершиной, имеющей метку $\mathbf{v}^k$ на уровне $V_k^{\otimes}$;
6: пометить добавленные вершины метками из множества $\mathcal{M}_{\mathbf{v}^k} = \mathsf{M\_orth}(\mathbf{c}_1^k, \mathbf{c}_2^k, \mathcal{M}_{\mathbf{c}_1^k}, \mathcal{M}_{\mathbf{c}_2^k})$.
7: **вернуть** $\mathrm{WB}^{\otimes}_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H},r_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}},L_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}}}[C_1 \otimes C_2]$

Поясним, почему дерево, строящееся в алгоритме MakeTensorTree, имеет $L_{\mathbf{1}_\mathcal{G}} + L_{\mathbf{1}_\mathcal{H}} - 1$ уровней. В этом алгоритме с помощью алгоритма M_orth для каждой вершины $\mathbf{v} = \mathbf{c}_1^i \otimes \mathbf{c}_2^j$ на уровне k дерева (2.25), где $i \in [\![0, L_{\mathbf{1}_\mathcal{G}} - 1]\!]$ – i–й уровень декодирующего дерева для кода C_1, $j \in [\![0, L_{\mathbf{1}_\mathcal{H}} - 1]\!]$ – j–й уровень декодирующего дерева для кода C_2, строится множество векторов на уровне $k+1$ дерева (2.25), состоящее из подмножеств двух типов:

1. $\mathbf{c}_1^i \otimes \mathbf{c}_2^{j+1}$,
2. $\mathbf{c}_1^{i+1} \otimes \mathbf{c}_2^j$.

Таким образом, максимальный уровень дерева (2.25) будет достигаться, например, если сначала поочередно пройти $L_{\mathbf{1}_\mathcal{G}} - 1$ векторов второго типа, каждый из которых находится на следующем уровне дерева, затем пройти $L_{\mathbf{1}_\mathcal{H}}$ векторов первого типа. Тогда глубина дерева (2.25) будет равна $L_{\mathbf{1}_\mathcal{G}} + L_{\mathbf{1}_\mathcal{H}} - 1$.

Если C_1 и C_2 – групповые MLD–коды, то есть $\mathrm{dmaj}(C_1) = d(C_1) - 1$ и $\mathrm{dmaj}(C_2) = d(C_2) - 1$, то в общем случае вспомогательное декодирующее дерево, построенное по алгоритму MakeTensorTree, не позволяет найти с помощью мажоритарного декодера значения ошибок $\mathbf{e}$, вес которых удовлетворяет неравенству (2.22). Действительно, мощность $|\mathcal{M}_{\mathbf{c}_1}| + |\mathcal{M}_{\mathbf{c}_2}|$ построенного алго-

ритмом MakeTensorTree M–ортогонального множества $\mathcal{M}_{\mathbf{c}_1\otimes\mathbf{c}_2}$ меньше

$$d(C_1\otimes C_2)-1=|\mathcal{M}_{\mathbf{c}_1}||\mathcal{M}_{\mathbf{c}_2}|+|\mathcal{M}_{\mathbf{c}_1}|+|\mathcal{M}_{\mathbf{c}_2}|$$

для каждого узла с меткой $\mathbf{c}_1\otimes\mathbf{c}_2$.

Вспомогательное декодирующее дерево (2.25) может быть достроено до полного декодирующего дерева MLD-кода $C_1\otimes C_2$ на основании конструкции 3 леммы 7 путем добавления недостающих вершин, однако представляется удобным работать только со значениями меток недостающих вершин. Алгоритм AddVals для каждого узла $\mathbf{v}=\mathbf{c}_1\otimes\mathbf{c}_2$ по принятому из канала вектору $\mathbf{x}$ и вспомогательному декодирующему дереву (2.25) вычисляет дополнительные $|\mathcal{M}_{\mathbf{c}_1}||\mathcal{M}_{\mathbf{c}_2}|$ значений меток недостающих вершин, но, подчеркнем, к дереву (2.25) при выполнении алгоритма AddVals дополнительные вершины не добавляются.

Алгоритм 4 AddVals

Исходные параметры: $\mathbf{x}$ – вектор вида (2.21), $\mathcal{M}_{\mathbf{v}^k}$ – M–ортогональное множество для $\mathbf{v}^k=\mathbf{c}_1^k\otimes\mathbf{c}_2^k$, $\mathrm{WB}^{\otimes}{}_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H},r_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}},L_{\mathbf{1}_\mathcal{G}\otimes\mathbf{1}_\mathcal{H}}}[C_1\otimes C_2]$ – декодирующее дерево, k – текущий уровень дерева.

Результат: Набор чисел ℓ_k мощности $|\mathcal{M}_{\mathbf{c}_1^k}|\cdot|\mathcal{M}_{\mathbf{c}_2^k}|$.

1: **если** $k=L_{\mathbf{1}_\mathcal{G}}+L_{\mathbf{1}_\mathcal{H}}-2$ **то**
2: **цикл** $1\leqslant i\leqslant|\mathcal{M}_{\mathbf{c}_2^k}|$ **выполнять**
3: **цикл** $|\mathcal{M}_{\mathbf{c}_2^k}|+1\leqslant j\leqslant|\mathcal{M}_{\mathbf{c}_2^k}|+|\mathcal{M}_{\mathbf{c}_1^k}|$ **выполнять**
4: к ℓ_k добавить $l(\mathbf{v}_i=(\mathbf{c}_1^i\otimes\mathbf{c}_2^i))+l(\mathbf{v}_j=(\mathbf{c}_1^j\otimes\mathbf{c}_2^j))+<\underline{\mathbf{c}_1^j\otimes\mathbf{c}_2^i,\mathbf{x}}>$, где $\mathbf{v}_i,\mathbf{v}_j\in\mathcal{M}_{\mathbf{v}^k},\mathbf{c}_1^j\otimes\mathbf{c}_2^i\in\overline{(C_1\otimes C_2)}$
5: **иначе**
6: **цикл** $1\leqslant i\leqslant|\mathcal{M}_{\mathbf{c}_2^k}|$ **выполнять**
7: **цикл** $|\mathcal{M}_{\mathbf{c}_2^k}|+1\leqslant j\leqslant|\mathcal{M}_{\mathbf{c}_2^k}|+|\mathcal{M}_{\mathbf{c}_1^k}|$ **выполнять**
8: к ℓ_k добавить $l(\mathbf{v}_i=(\mathbf{c}_1^i\otimes\mathbf{c}_2^i))+l(\mathbf{v}_j=(\mathbf{c}_1^j\otimes\mathbf{c}_2^j))+l(\mathbf{c}_1^j\otimes\mathbf{c}_2^i)$, где $\mathbf{v}_i,\mathbf{v}_j\in\mathcal{M}_{\mathbf{v}^k},\mathbf{c}_1^j\otimes\mathbf{c}_2^i\in V^{\otimes}{}_{k+2}$
9: **вернуть** ℓ_k

В алгоритме AddVals в зависимости от того, на каком уровне находится вершина дерева (2.25), для которой необходимо вычислить значение метки, вычисляются значения дополнительных меток, необходимых для декодирования.

В случае если вершина находится на предпоследнем уровне дерева, значение дополнительной метки вычисляется как сумма меток для вершин первого типа $\mathbf{c}_1^i \otimes \mathbf{c}_2^i$, второго типа $\mathbf{c}_1^j \otimes \mathbf{c}_2^j$ из леммы 7 и скалярного произведения декодируемого вектора на вектор $\mathbf{c}_1^j \otimes \mathbf{c}_2^i$, составленный из соответствующих компонент векторов первого и второго типа. Ввиду того, что вершины находятся на последнем уровне дерева, вектор $\mathbf{c}_1^j \otimes \mathbf{c}_2^i$ также будет принадлежать коду $\overline{(C_1 \otimes C_2)}$ и значение метки вычислится корректно. В случае, когда заданная вершина находится на уровне $k < L_1 + L_2 - 2$, вместо вычисления скалярного произведения значение метки для вектора $\mathbf{c}_1^j \otimes \mathbf{c}_2^i$ берется на уровне $k + 2$ дерева (2.25).

Приведем, построенный в [39], алгоритм MajorVote, выполняющий процедуру мажоритарного голосования. В алгоритме MajorVote по множеству $\mathcal{A}$

Алгоритм 5 MajorVote

Исходные параметры: $\mathcal{A}$ – последовательность чисел из $\mathbb{F}_q$.

Результат: элемент $v(\in \mathbb{F}_q)$, который в последовательности $\mathcal{A}$ встречается наибольшее число раз.

1: **для каждого** $a \in \mathbb{F}_q$ **выполнять**
2: вычислить величину $\mathrm{count}(a)$, равную числу появления элемента a в последовательности $\mathcal{A}$
3: **если** найдется только один $a'(\in \mathbb{F}_q)$, что $\mathrm{count}(a') \geqslant \lceil |\mathcal{A}|/2 \rceil$ **то**
4: $v := a'$
5: **иначе**
6: $v := 0$
7: **вернуть** v

выбирается элемент, который встречается наибольшее количество раз.

Построим алгоритм декодирования DecodeTensorBit, который по принятому вектору $\mathbf{x}$ находит значение $e_{\mathbf{1}_\mathcal{G} \otimes \mathbf{1}_\mathcal{H}}$ вектора ошибок $\mathbf{e}$ в координате, соответствующей базисной функции $\mathbf{1}_\mathcal{G} \otimes \mathbf{1}_\mathcal{H}$. (Отметим, что в алгоритме DecodeTensorBit используется операция конкатенации наборов чисел, которая обозначается символом $\uplus$.) Таким образом, алгоритм DecodeTensorBit в случае тензорного произведения кодов выполняет функцию упомянутого выше

алгоритма мажоритарного декодирования и правильно находит значение координаты вектора ошибок $\mathbf{e}$, когда вес ошибки удовлетворяет неравенству (2.22).

Алгоритм 6 DecodeTensorBit

Исходные параметры: $\mathbf{x}$ – вектор вида (2.21), $\mathrm{WB}^{\otimes}[C_1 \otimes C_2] = \mathrm{WB}^{\otimes}{}_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}},r_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}},L_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}}}[C_1 \otimes C_2]$ – декодирующее дерево.
Результат: $e_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}}$.

цикл $1 \leqslant k \leqslant L_{\mathbf{1}_{\mathcal{G}}} + L_{\mathbf{1}_{\mathcal{H}}} - 1$ **выполнять**
  для каждого $\mathbf{v}_k \in V^{\otimes}{}_k$ **выполнять**
    если $\mathbf{v}_k \in \overline{(C_1 \otimes C_2)}$ **то**
      $l(\mathbf{v}_k) :=< \mathbf{v}_k,\mathbf{x} >$
    иначе
      $l(\mathbf{v}_k) := \mathsf{MajorVote}(l[\mathcal{M}_{v^k}] \uplus \mathsf{AddVals}(\mathbf{x},\mathcal{M}_{v^k},\mathrm{WB}^{\otimes}[C_1 \otimes C_2],k))$
$e_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}} := \mathsf{MajorVote}(l[\mathcal{M}_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}}] \uplus \mathsf{AddVals}(\mathbf{x},\mathcal{M}_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}},\mathrm{WB}^{\otimes}[C_1 \otimes C_2],0))$
вернуть $e_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}}$

Заметим, что алгоритм MakeTensorTree строит вспомогательное декодирующее дерево для координаты, соответствующей элементу $\widehat{1}_{\mathcal{G}} \times \widehat{1}_{\mathcal{H}} (\in \mathcal{G} \times \mathcal{H})$. В [39] для групповых кодов построен приведенный ниже вспомогательный алгоритм CloneTree, позволяющий по вспомогательному декодирующему дереву с одной меткой у корня построить вспомогательное декодирующее дерево для корня с любой другой меткой.

Алгоритм 7 CloneTree

Исходные параметры: $\mathcal{G}$, $\mathrm{WB}^{\otimes}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$, $\mathbf{b}'$.
Результат: $\mathrm{WB}^{\otimes}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$.

$\mathrm{WB}^{\otimes}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C] := \mathrm{WB}^{\otimes}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$
Найти $g (\in \mathcal{G} \times \mathcal{H})$ такой, что $(g,\mathbf{b}) = \mathbf{b}'$
для каждого метки $\mathbf{p}$ дерева $\mathrm{WB}^{\otimes}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$ **выполнять**
  $\mathbf{p} := (g,\mathbf{p})$ ▷ Действие элементом g на $\mathbf{p}$ по правилу (1.7)
вернуть $\mathrm{WB}^{\otimes}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$

Таким образом, набор вспомогательных декодирующих деревьев

$$\mathcal{WB}^{\otimes}(C_1 \otimes C_2) = \{\mathrm{WB}^{\otimes}_{\delta_{(g,h)},r_{\delta_{(g,h)}},L_{\delta_{(g,h)}}}\}_{\delta_{(g,h)} \in B^{\mathbb{F}q(\mathcal{G}\times\mathcal{H})}}$$

для группового кода $C_1 \otimes C_2$ может быть построен по дереву (2.25). Именно, для любой базисной функции $\delta_{(g,h)} \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}$:

$$\mathrm{WB}^{\otimes}_{\delta_{(g,h)},r_{\delta_{(g,h)}},L_{\delta_{(g,h)}}} = \mathsf{CloneTree}(\mathcal{G}\times\mathcal{H},\mathrm{WB}^{\otimes}{}_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}},r_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}},L_{\mathbf{1}_{\mathcal{G}}\otimes\mathbf{1}_{\mathcal{H}}}},\delta_{(g,h)}).$$

По аналогии с алгоритмом Decoder3 из [39] построен алгоритм DecodeTensorVector декодирования принятого вектора $\mathbf{x}$, в котором каждая координата декодируется с помощью алгоритма DecodeTensorBit.

Алгоритм 8 DecodeTensorVector

Исходные параметры: принятый вектор $\mathbf{x} = \mathbf{c} + \mathbf{e}$, набор вспомогательных декодирующих деревьев $\mathcal{WB}^{\otimes}(C_1 \otimes C_2) = \{\mathrm{WB}^{\otimes}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}\}_{\mathbf{b}\in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}}$.

Результат: вектор $\mathbf{c}'$ – результат декодирования.

1: **для каждого** $\mathbf{b} \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}$ **выполнять**

2: $\quad c'_{\mathbf{b}} := x_{\mathbf{b}} - \mathsf{DecodeTensorBit}(\mathbf{x},\mathbf{b}_i,\mathrm{WB}^{\otimes}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}})$

3: **вернуть** $\mathbf{c}'$

Декодирование M–ортогонального семейства кодов

В данном разделе определяется конструкция декодирующего графа для M–ортогонального семейства кодов и строятся алгоритмы декодирования по этому графу. Похожая конструкция была разработана в [39] и применялась в предыдущем разделе, где использовались декодирующие деревья. Новая конструкция декодирующих графов, которая предъявляется в данном разделе, с одной стороны является более удобной для построения алгоритмов, а с другой – более общей и будет использована далее для декодирования групповых D-кодов.

Ниже разработан процесс мажоритарного декодирования кода $\overline{C(J-1)}$, двойственного коду $C(J-1)$ из семейства $\mathcal{S}$ (см. (2.1)). Для вектора $\mathbf{b}(\in C(0))$ веса 1 и кода $\overline{C(J-1)}$ определим декодирующий граф $G(\mathcal{S},\mathbf{b}) = (V,E)$ как связный ориентированный ациклический граф, удовлетворяющий следующим условиям, при формулировке которых используются обозначения из s1–s3.

g1) В графе существует единственный исток, который помечен парой $(\mathbf{b},(0,k))$, где $k \in [\![1,J]\!]$. Остальные вершины графа $v \in V$ помечены парами $(\mathbf{x},(i,j))$, где $\mathbf{x} \in U_{ij}$, U_{ij} - порождающее множество кода $C(i)$, удовлетворяющее условию s3).

g2) Вершины, у которых нет исходящих дуг, помечены парой $(\mathbf{x}, (J-1, J))$, где $\mathbf{x} \in C(J-1)$; эти вершины являются стоками графа.

g3) Любая помеченная вершина $v_1(\mathbf{x_1}, (i,j))$, не являющаяся стоком графа, является началом $d_{ij} - 1$ ориентированных дуг

$$e = (v_1(\mathbf{x_1}, (i,j)), v_2(\mathbf{x_2}, (j,k))) \in E$$

для некоторого фиксированного $k \in [\![j+1,J]\!]$, где $\mathbf{x_2}$ пробегает множество $\mathcal{M}_{\mathbf{x_1}}$ из условия s3; т.е. вектор $\mathbf{x_2}$, которым помечена вершина v_2, входит в M-ортогональное множество для вектора $\mathbf{x_1}$, которым помечена вершина v_1.

g4) Из любой вершины существует путь к одному из стоков графа.

Легко убедиться, что существование такого графа является следствием условий s1–s3, однако этот граф в общем случае определяется неоднозначно. В частном случае семейства кодов Рида–Маллера построение декодирующего графа аналогично построению декодирующего дерева, алгоритм которого описан в [39] и приводится в разделе 1.4.2.

В случае использования семейства групповых MLD–кодов построение набора декодирующих графов для каждой координаты пространства значительно упрощается. В частности, если для базисной функции $\mathbf{1} = \delta_{\hat{1}} = 1\hat{1}$ удалось построить декодирующий граф $G(\mathcal{S}, \mathbf{1})$, то граф $G(\mathcal{S}, \mathbf{g})$ для базисной функции $\mathbf{g} = \delta_g = 1g$ может быть построен путем действия элементов $g^{-1}(\in \mathcal{G})$ на метки вершин графа $G(\mathcal{S}, \mathbf{1})$ по правилу (1.7).

Опишем принцип построения декодирующего графа G для вектора $\mathbf{b}(\in C(0))$ веса 1 и кода $\overline{C(J-1)}$ из семейства S при выполнении условий s1–s3 для этого семейства. Декодирующий граф строится итерационно. Сначала для кода $C(0)$ определяем код $C(i)$ из семейства, который удовлетворяет условию $C(0) \overset{d_{0i}}{\longleftarrow} C(i)$. Затем в коде $C(i)$ выберем множество векторов $\mathcal{M}_{\mathbf{b}} = \{\mathbf{u} | \mathbf{u} \in$

$C(i)\}, |\mathcal{M}_{\mathbf{b}}| = d_{0i} - 1$, являющееся M–ортогональным для вектора $\mathbf{b}(\in C(0))$. Соединим вершины, соответствующие этим векторам, с помощью дуг следующим образом. Вершина $v_{\mathbf{b}}$, помеченная вектором $\mathbf{b}$, соединяется с каждой вершиной $v_{\mathbf{u}}$, помеченной вектором из M–ортогонального множества, таким образом, что вершина $v_{\mathbf{b}}$ является началом дуги, а каждая вершина $v_{\mathbf{u}}(\mathbf{u} \in \mathcal{M}_{\mathbf{b}})$ является концом соответствующей дуги. На следующей итерации построения графа для кода $C(i)$ определяем код $C(j)$ из семейства, который удовлетворяет условию $C(i) \xleftarrow{d_{ij}} C(j)$, и для каждого вектора $\mathbf{u}(\in \mathcal{M}_{\mathbf{b}})$ и соответствующей вершины $v_{\mathbf{u}}$ выполняем аналогичные действия: находим M–ортогональные множества для каждого вектора $\mathbf{u}$ и соединяем соответствующие вершины дугами. На каждой такой последующей итерации алгоритма вершины, помеченные векторами из M–ортогональных множеств, являются стоками промежуточного графа. Алгоритм заканчивает свое выполнение, когда стоками графа становятся вершины, помеченные векторами из кода $C(J-1)$, т.е. когда для какого-то промежуточного кода $C(k)$ выполняется условие $C(k) \xleftarrow{d_{k,J-1}} C(J-1)$, а также все вершины, помеченные необходимыми векторами из $C(k)$, соединяются с вершинами, помеченными векторами из $C(J-1)$, составляющими M–ортогональное множество для каждого такого вектора из $C(k)$. Таким образом, направление дуг в результирующем графе определяется описанным выше процессом, при этом вершина, помеченная вектором $\mathbf{b}$, является истоком графа, а вершины, помеченные векторами из кода $C(J-1)$ являются стоками графа, что соответствует условию g3. Из описанного алгоритма построения графа, с учетом направления дуг, вытекает выполнение условия g4.

Отметим, что если все векторы из M–ортогональных множеств, построенных на одной итерации алгоритма, не пересекаются для каждой такой итерации, то декодирующий граф представляет собой дерево. В этом случае на каждой итерации алгоритма достраивается соответствующий уровень дерева. Количество вершин $\kappa(T)$ в дереве T для произвольного кода определяется формулой

$$\kappa(T) = 1 + \sum_{i=1}^{L} \prod_{k=1}^{i} \ell_k,$$

где ℓ_k – количество потомков для какого-либо узла дерева на уровне k, т.к. количество потомков для узлов, находящихся на одном уровне, совпадает, а L – высота дерева. Действительно, на нулевом уровне находится ровно одна вершина $v_{\mathbf{b}}$, являющаяся корнем, количество вершин на первом уровне равно мощности M–ортогонального множества для вектора $\mathbf{b}$, количество вершин на втором уровне дерева равно произведению мощности M–ортогонального множества для вектора $\mathbf{b}$ и мощности M–ортогонального множества для одного из векторов, который принадлежит M–ортогональному множеству для вектора $\mathbf{b}$ и т.д.

В общем случае, когда M–ортогональные множества для разных векторов, которыми помечены вершины графа, могут пересекаться, количество вершин в декодирующем графе $G(\mathcal{S},\mathbf{b})$ для вектора $\mathbf{b}$ и кода $\overline{C(J-1)}$ может быть меньше, чем в случае дерева, но, как и в случае дерева, определяется мощностью M–ортогональных множеств для векторов, которыми помечены вершины графа. Поскольку код $\overline{C(J-1)}$ является $l-MLD$–кодом для некоторого $l \leqslant J-1$, то существует такая последовательность $j_1 = 0, j_2, \ldots, j_l = J-1$, что выполняется условие $C(j_k) \xleftarrow{d_{j_k,j_{k+1}}} C(j_{k+1})$, и, таким образом, количество вершин оценивается по формуле

$$\kappa(G) \leqslant 1 + \sum_{i=1}^{l-1} \prod_{k=1}^{i} (d_{j_k,j_{k+1}} - 1),$$

где $d_{j_k,j_{k+1}} - 1$ – количество потомков, определяемых (2.2) и условием s3.

Пример 4. *Пусть* $S = \{\mathbb{F}_2^8, \mathrm{RM}_2(2,3), \mathrm{RM}_2(1,3), \mathrm{RM}_2(0,3), \{\bar{0}\}\}$ *– семейство кодов Рида–Маллера. Декодирующий граф для вектора* $\mathbf{b} = (10000000)$ *и кода* $\overline{\mathrm{RM}_2(1,3)} = \mathrm{RM}_2(1,3)$ *изображен на рисунке 2.1.*

Ниже с помощью графа $G(\mathcal{S},\mathbf{b})$ опишем процесс мажоритарного декодирования для кодов семейства $\mathcal{S}$. Сначала рассмотрим кодовый вектор $\mathbf{c}(\in \overline{C(J-1)})$, принятый из канала вектор $\mathbf{x} = \mathbf{c} + \mathbf{e}$, где $\mathrm{wt}(\mathbf{e}) \leqslant \left\lfloor (d(\overline{C(J-1)}) - 1)/2 \right\rfloor$, $d(\overline{C(J-1)})$ – минимальное кодовое расстояние кода $\overline{C(J-1)}$, и рассмотрим разложение $\sum_{\mathbf{b}\in B} e_{\mathbf{b}}\mathbf{b}$ вектора ошибок $\mathbf{e}$ по базису

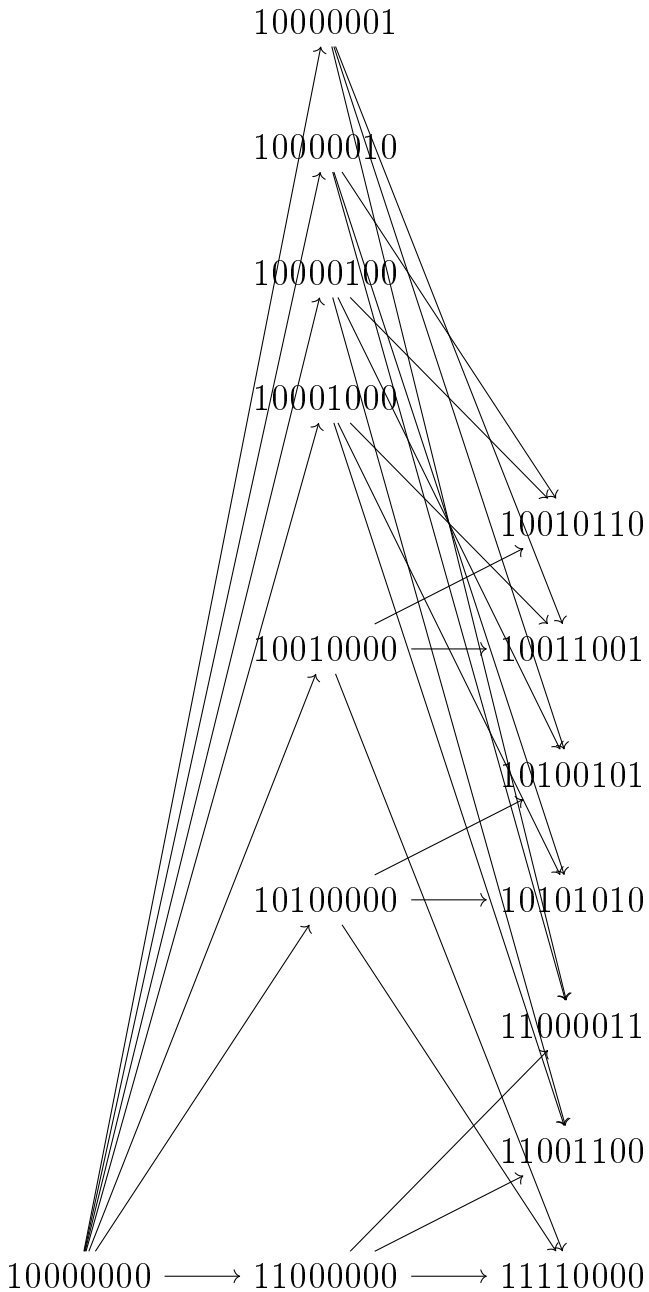

Рисунок 2.1 — Декодирующий граф для кода $\mathrm{RM}_2(1,3)$.

B. Если для $\mathbf{b}$-координаты построен *декодирующий граф* $G(\mathcal{S},\mathbf{b}) = (V,E)$, то значение $e_{\mathbf{b}}$ для $\mathbf{b}$–координаты вектора $\mathbf{e}$, как показано ниже, вычисляется однозначно с помощью метода мажоритарного декодирования. Если для каждой координаты кода C существует декодирующий граф, то существует такая последовательность $j_1 = 0,j_2,\ldots,j_l = j$, что выполняется условие $C(j_k) \xleftarrow{d_{j_k,j_{k+1}}} C(j_{k+1})$, поэтому этот код является $l - MLD$–кодом.

Процесс декодирования кода $\overline{C(J-1)}$ в ненулевой координате вектора $\mathbf{b}$ заключается в сопоставлении каждой вершине графа $G = G(\mathcal{S},\mathbf{b})$ элемента из

$\mathbb{F}_q$, которым будет помечаться соответствующая вершина. Вычисление этих дополнительных «декодирующих» меток выполняется посредством обхода графа, к примеру, с помощью рекурсивного алгоритма. Обход графа начинается с истока. Для каждой рассматриваемой вершины по графу находится множество вершин, являющихся прямыми потомками для данной вершины. Далее в зависимости от того, является ли полученное множество пустым или нет, выполняются следующие действия. Если множество потомков пусто, значит рассматриваемая вершина является стоком и помечена парой $(J-1, J)$ и, соответственно, вектором из кода $C(J-1)$. Для таких вершин считается скалярное произведение принятого по каналу вектора и вектора, которым помечена данная вершина. Результатом в этом случае будет являться сумма координат вектора ошибок, соответствующих ненулевым координатам вектора, которым помечена данная вершина. В другом случае, когда множество потомков не пусто, вычисляется множество «декодирующих» меток потомков M_l. Для каждого потомка выполняется следующая проверка. Если метка потомка вычислена, то есть алгоритм обхода уже проходил эту вершину, то метка добавляется ко множеству M_l. В противном случае для этой вершины выполняются все вышеописанные действия, и обход продолжается уже в этой вершине. После вычисления меток всех потомков, полученное множество M_l подается на вход алгоритму MajorVote из [39], который выполняет голосование. Для полноты изложения этот алгоритм приводится и в данной работе. Результат голосования является элементом из $\mathbb{F}_q$, которым и помечается текущая вершина. После обхода всего графа значением «декодирующей» метки истока будет являться значением координаты вектора ошибки, соответствующей вектору $\mathbf{b}$.

В соответствии с вышеописанным процессом декодирования рассмотрим формальные алгоритмы decode, decode_node для декодирования координаты зашумленного кодового слова $\mathbf{x} = \mathbf{c} + \mathbf{e}$, соответствующей вектору $\mathbf{b} \in \mathbb{F}_q^n$, где $\mathbf{c} \in \overline{C(J-1)}$, $\mathbf{e} \in \mathbb{F}_q^n$. Будем использовать алгоритм descendants(v,G), который, получая на вход вершину $v \in G$ и граф G, возвращает множество вершин графа, являющихся прямыми потомками для вершины в графе G. Этот алгоритм

ввиду его простоты приводить не будем. К алгоритму descendants обращается рекурсивный алгоритм decode_node, который выполняет обход графа, помечая его вершины дополнительными «декодирующими» метками. В алгоритме decode на вход подается зашумленное кодовое слово $\mathbf{x}$, граф G и осуществляется вызов алгоритма decode_node для истока графа. На выходе алгоритма decode получается принятый по каналу кодовый вектор, в котором исправлена координата, соответствующая вектору $\mathbf{b}$.

Алгоритм 9 decode_node

Исходные параметры: $\mathbf{y}$ – принятый по каналу зашумленный кодовый вектор, $v_k(\mathbf{x_k},(i,j))$ – вершина графа G, G – декодирующий граф для кода $\overline{C(J-1)}$

Результат: $l(v_k)(\in \mathbb{F}_q)$ – значение метки для вершины v_k

1: $D :=$ descendants$(v_k(\mathbf{x_k},(i,j)),G)$;
2: **если** $D = \{\varnothing\}$ **то**
3: $\quad l(v_k) := (\mathbf{y},\mathbf{x_k})$; ▷ где $\mathbf{x_k}$ – вектор, которым помечена вершина v_k.
4: **иначе**
5: $\quad M_l := \varnothing$;
6: $\quad$ **для каждого** $v_l \in D$ **выполнять**
7: $\quad\quad$ **если** $l(v_l)! = \varnothing$ **то**
8: $\quad\quad\quad$ к M_l добавить $l(v_l)$.
9: $\quad\quad$ **иначе**
10: $\quad\quad\quad$ к M_l добавить decode_node$(\mathbf{y}, v_l, G)$;
11: $\quad l(v_k) :=$ MajorVote(M_l);
12: Пометить вершину v_k меткой $l(v_k)$.
13: **вернуть** $l(v_k)$

Алгоритм 10 decode

Исходные параметры: $\mathbf{x}$ – зашумленный кодовый вектор, G – декодирующий граф для кода $\overline{C(J-1)}$

Результат: $\widetilde{c}(\in \mathbb{F}_q^n)$ – зашумленный кодовый вектор с декодированной координатой, соответствующей вектору, которым помечен исток $v_0(\mathbf{b},(0,j))$ графа G

1: $t :=$ decode_node$(\mathbf{x}, v_0(\mathbf{b},(0,j)), G)$;
2: $\widetilde{\mathbf{c}} = \mathbf{x} + \mathbf{b}_t$; ▷ где $\mathbf{b}_t$ – вектор $\mathbf{b}$, в котором ненулевая координата заменена на t.
3: **вернуть** $\widetilde{x}$

Пример 5. *Проиллюстрируем процесс мажоритарного декодирования на примере кода* $\mathrm{RM}_2(1,3)$ *и декодирующего графа* $G = (V,E)$*, изображенного на рисунке 2.1. Пусть* $\mathbf{c} = (01010101)(\in \overline{\mathrm{RM}_2(1,3)} = \mathrm{RM}_2(1,3))$ *– кодовый вектор,* $\mathbf{x} = (10010101)$ *– принятый по каналу вектор,* $\mathbf{e} = (11000000)$ *– вектор ошибок. Выполним декодирование первой координаты вектора* $\mathbf{x}$*. Для вершин, являющихся стоками графа, вычислим скалярное произведение* $\mathbf{x}$ *на векторы, которыми помечены эти вершины, и пометим рассмотренные вершины результатом скалярного произведения. Для остальных вершин значение дополнительных декодирующих меток вычисляется с помощью голосования по меткам потомков. Результирующий помеченный граф изображен на рисунке 2.2. Метка истока графа соответствует значению ошибки в первой координате.*

Построение декодирующего графа для D-кодов

Теперь опишем процедуру построения декодирующего графа для кода $\overline{C(D)}$ (см. (2.20)). В предыдущем разделе разработана конструкция декодирующего графа $G(\mathcal{S}, \mathbf{b}) = (V, E)$ для произвольного M–ортогонального семейства $\mathcal{S}$, содержащая условия g1–g4. Пусть $G_1(\mathcal{S}_1, \mathbf{b}^1) = (V_1, E_1)$, $G_2(\mathcal{S}_2, \mathbf{b}^2) = (V_2, E_2)$ – декодирующие графы для семейств $\mathcal{S}_1$, $\mathcal{S}_2$ соответственно. Ниже мы будем использовать обозначения из g1–g4, адаптированные к семействам $\mathcal{S}_1$, $\mathcal{S}_2$, т.е. заменять $\mathbf{b}$ на $\mathbf{b}^t$, $\mathcal{S}$ на $\mathcal{S}_t$, где $t \in \{1,2\}$, и добавлять при необходимости к обозначениям из g1–g4 справа вверху индекс t, соответствующий семейству $\mathcal{S}_t$. По этим графам определим граф $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2) = (V^{\otimes}, E^{\otimes})$, с помощью которого выполняется декодирование кода $\overline{C(D)}$.

1) Вершины графа будем помечать одним из трех типов type вектора в соответствии с леммой 7, а также некоторыми дополнительными метками. Именно, вершины первого и второго типов будем дополнительно помечать трой-

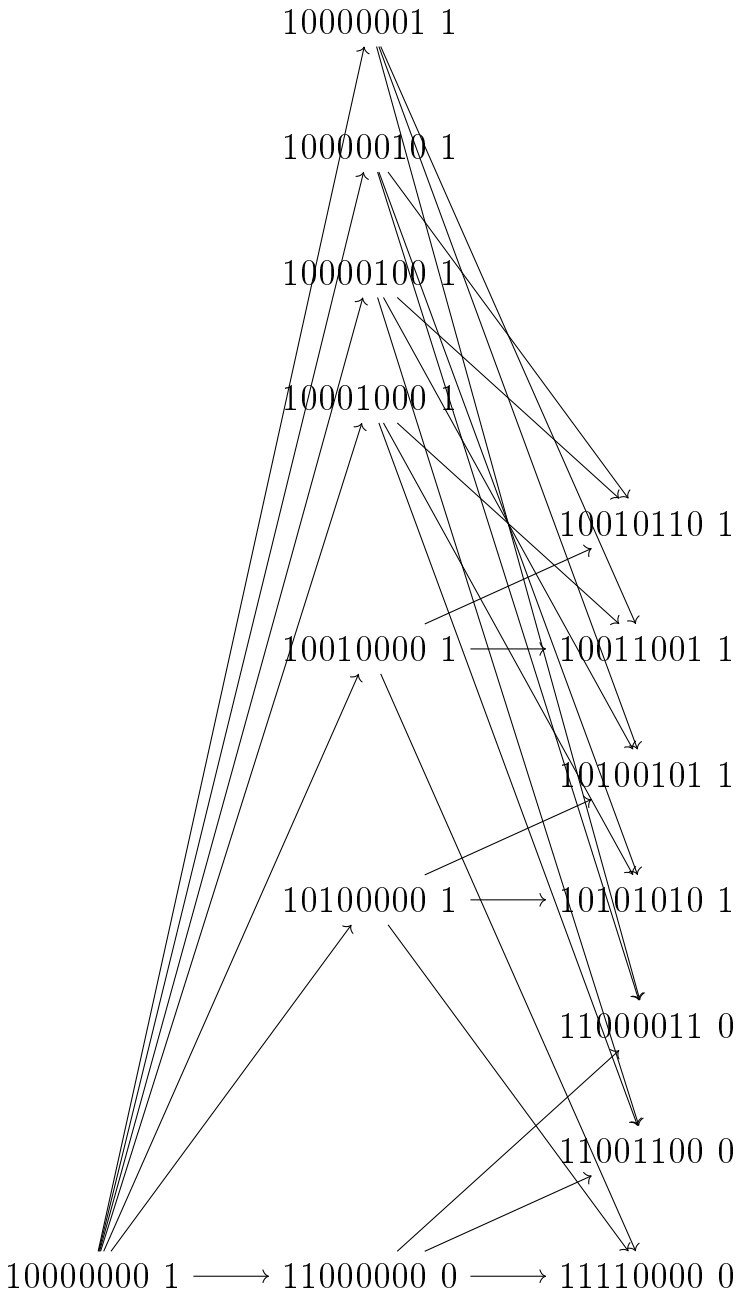

Рисунок 2.2 — Помеченный декодирующий граф для кода $\mathrm{RM}_2(1,3)$.

кой: вектором $\mathbf{x}_k \otimes \mathbf{x}_l$ и парой вершин (v_k^1,v_l^2), где $v_k^1(\mathbf{x}_k,i) \in V_1, v_l^2(\mathbf{x}_l,j) \in V_2$, а $\mathbf{c}_1 \in C_1(i), \mathbf{c}_2 \in C_2(j)$. Вершины третьего типа будем дополнительно помечать вектором $\mathbf{c} \in \mathbb{F}_q(\mathcal{G} \times \mathcal{H})$.

2) В графе существует единственный исток, который помечен набором $(\mathbf{b}^1 \otimes \mathbf{b}^2, v_0^1(\mathbf{b}^1,0), v_0^2(\mathbf{b}^2,0), \mathrm{type})$, где $\mathrm{type} = 1$, а v_0^t – исток графа G_t семейства $\mathcal{S}_t$.

3) Стоки графа помечены наборами $(\mathbf{x}_k^1 \otimes \mathbf{x}_k^2, v^1(\mathbf{x}_k^1,i), v^2(\mathbf{x}_k^2,j), \mathrm{type})$, такими, что $(i,j) \in D_b$.

4) Любая вершина $v(\mathbf{x}_k^1 \otimes \mathbf{x}_k^2, v^1(\mathbf{x}_k^1,i), v^2(\mathbf{x}_k^2,j), \text{type})$ с type $\in \{1,2\}$, не являющаяся стоком графа, является началом $d_i d_j - 1$ ориентированных дуг (d_i и d_j определяются в соответствии с условием g4), на концах которых находятся такие вершины, что векторы, которыми помечены эти вершины, составляют M–ортогональное множество для вектора $\mathbf{x}_k^1 \otimes \mathbf{x}_k^2$.

Отметим, что из любой вершины существует путь к одному из стоков графа. Вершины графа обозначим именем вершины и списком всех меток, при этом длина списка зависит от типа. Кроме того, будем использовать сокращенные обозначения $v(\mathbf{x}_k^1 \otimes \mathbf{x}_k^2)$ или просто v.

Сконструируем алгоритмы build_tens_node и build_tens_graph, реализующие процедуру построения декодирующего графа для кода $\overline{C(D)}$.

Алгоритм 11 build_tens_node

Исходные параметры: $G_t(\mathcal{S}_t, \mathbf{b}^t) = (V_t, E_t)$ – декодирующий граф для семейства $\mathcal{S}_t$, $t \in \{1,2\}$, $v_z(\mathbf{x}_k^1 \otimes \mathbf{x}_k^2, v_k^1, v_k^2, \text{type}) \in V^{\otimes}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ – промежуточный декодирующий граф для кода $\overline{C(D)}$.

Результат: $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ – следующая промежуточная версия декодирующего графа для кода $\overline{C(D)}$.

1: $Desc_1 :=$ descendants$(v_k^1(\mathbf{x}_k^1, (i^1,j^1)), G(\mathcal{S}_1))$;
2: $Desc_2 :=$ descendants$(v_k^2(\mathbf{x}_k^2, (i^2,j^2)), G(\mathcal{S}_1))$;
3: **для каждого** $v_m^2(\mathbf{x}_m^2, (j_2, p_2)) \in Desc_2$ **выполнять**
4: **если** $v_r(\mathbf{x}_k^1 \otimes \mathbf{x}_m^2, v_k^1, v_m^2, 1) \notin V^{\otimes}$ **то**
5: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ вершину $v_r(\mathbf{x}_k^1 \otimes \mathbf{x}_m^2, v_k^1, v_m^2, 1)$.
6: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу (v_z, v_r).
7: **если** $(i_1, j_2) \notin D$ **то**
8: build_tens_node$(G_1, G_2, v_r, G(\mathcal{S}_1 \otimes \mathcal{S}_2))$;
9: **для каждого** $v_l^1(\mathbf{x}_l^1, (j_1, p_1)) \in Desc_1$ **выполнять**
10: **если** $v_r(\mathbf{x}_l^1 \otimes \mathbf{x}_k^2, v_l^1, v_k^2, 2) \notin V^{\otimes}$ **то**
11: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ вершину $v_r(\mathbf{x}_l^1 \otimes \mathbf{x}_k^2, v_l^1, v_k^2, 2)$.
12: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу (v_z, v_r).
13: **если** $(j_1, i_2) \notin D$ **то**
14: build_tens_node$(G_1, G_2, v_r, G(\mathcal{S}_1 \otimes \mathcal{S}_2))$; ▷ (продолжение следует)

Алгоритм build_tens_node получает на вход два графа G_1, G_2, соответствующие семействам $\mathcal{S}_1$, $\mathcal{S}_2$, промежуточную версию итогового графа $G(\mathcal{S}_1 \times$

Алгоритм 11 build_tens_node (продолжение)

15: **для каждого** $v_m^2(\mathbf{x}_m^2, (j_2, p_2)) \in Desc_2$ **выполнять**
16: **для каждого** $v_l^1(\mathbf{x}_l^1, (j_1, p_1)) \in Desc_1$ **выполнять**
17: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ вершину $v_{t3}(\mathbf{x}_k^1 \otimes \mathbf{x}_m^2 + \mathbf{x}_l^1 \otimes \mathbf{x}_k^2 + \mathbf{x}_l^1 \otimes \mathbf{x}_m^2, 3)$.
18: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу (v_z, v_{t3}).
19: **если** $\mathbf{x}_{t3} \notin \overline{C(D)}$ **то**
20: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу $(v_{t3}, v_{z1}(\mathbf{x}_k^1 \otimes \mathbf{x}_m^2))$, где $v_{z1} \in V^{\otimes}$ – вершина графа G, помеченная вектором $\mathbf{x}_k^1 \otimes \mathbf{x}_m^2$.
21: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу $(v_{t3}, v_{z2}(\mathbf{x}_l^1 \otimes \mathbf{x}_k^2))$, где $v_{z2} \in V^{\otimes}$ – вершина графа G, помеченная вектором $\mathbf{x}_l^1 \otimes \mathbf{x}_k^2$.
22: Добавить в граф $G(\mathcal{S}_1 \times \mathcal{S}_2)$ дугу $(v_{t3}, v_{z3}(\mathbf{x}_l^1 \otimes \mathbf{x}_m^2))$, где $v_{z3} \in V^{\otimes}$ – вершина графа G, помеченная вектором $\mathbf{x}_l^1 \otimes \mathbf{x}_m^2$.
23: **вернуть** $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$

$\mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2$) с текущей вершиной этого графа и возвращает следующую промежуточную версию этого графа.

Пусть текущая вершина имеет первый или второй тип. Метки вершин исходных графов, которыми помечена текущая вершина, соответствуют точкам множества D_0. Алгоритм выполняет свою работу для точек, не принадлежащих множеству D. Для текущей вершины находятся множества потомков для вершин исходных графов G_1 и G_2, которыми помечена текущая. Далее, в соответствии с леммой 7, для текущей вершины сначала строится множество потомков первого типа, при этом, после нахождения каждого такого потомка, вершина добавляется ко множеству вершин итогового графа, добавляется дуга, соединяющая текущую вершину и потомка и, если новая вершина не принадлежит D_b и тем более D, то выполняется рекурсивный вызов алгоритма уже для потомка текущей вершины. После добавления в граф потомков первого типа

происходит вычисление потомков второго типа и выполняются аналогичные действия.

В заключение строятся потомки третьего типа, каждый из которых соединяется дугами с вершинами первого и второго типов. Каждая вершина третьего типа помечена вектором, который является суммой трех векторов. При добавлении исходящих дуг из вершины третьего типа выполняется поиск вершин первого или второго типов, которые помечены этими векторами, и найденные вершины будут концами этих дуг.

Как было указано ранее, каждая вершина графа соответствует точке множества D_0. При добавлении в граф вершин проверяется принадлежность соответствующей точки множеству D. Алгоритм build_tens_node заканчивает свою работу после добавления всех вершин, соответствующих точкам множества D_b, т.е. стоков.

Алгоритм 12 build_tens_graph

Исходные параметры: $G_t(\mathcal{S}_t, \mathbf{b}^t) = (V_t, E_t)$ – декодирующий граф для семейства $\mathcal{S}_t$, $v_0^t(\mathbf{b}^t, (0, i^t)) \in V_t$ – исток графа G_t, $t \in \{1,2\}$.

Результат: $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ – декодирующий граф для кода $\overline{C(D)}$.

1: $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2) := \varnothing$;
2: Добавить в граф G вершину $v_0(\mathbf{b}^1 \otimes \mathbf{b}^2, v_0^1, v_0^2, 1)$;
3: $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2) :=$ build_tens_node($\mathsf{G}_1, \mathsf{G}_2, \mathsf{v}_0, \mathsf{G}$);
4: **вернуть** $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$

Алгоритм build_tens_graph, получая на вход два графа G_1, G_2 для семейств $\mathcal{S}_1$, $\mathcal{S}_2$ и истоки этих графов, строит декодирующий граф $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ для кода $\overline{C(D)}$. В алгоритме, после инициализации искомого графа, добавляется исток и выполняется обращение к алгоритму build_tens_node, на вход которому подается этот исток. Результатом работы этого алгоритма является граф для декодирования координаты кода $\overline{C(D)}$, соответствующей вектору $\mathbf{b}^1 \otimes \mathbf{b}^2$.

Декодирование одной координаты

Выше построен декодирующий граф $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ для кода $\overline{C(D)}$. Теперь построим алгоритмы декодирования с помощью этого графа: decode_tens_node и decode_tens_graph. Как и в алгоритме из раздела 2.3.2 будем пользоваться вспомогательным алгоритмом descendants, который возвращает множество потомков заданной вершины. Также будем пользоваться вспомогательным алгоритмом type, возвращающим тип заданной вершины.

Алгоритм 13 decode_tens_node

Исходные параметры: $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ – декодирующий граф для кода $\overline{C(D)}$, $v_r \in G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$, $\mathbf{y}$ – принятый по каналу вектор.
Результат: $l(v_r)(\in \mathbb{F}_q)$ – значение метки для вершины v_r.

$Desc :=$ descendants$(G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2), v_r)$
если $Desc = \{\varnothing\}$ **то**
    $l(v_r) := (\mathbf{x}_r, \mathbf{y})$;
иначе
    если type$(v_r) = 1$ or type$(v_r) = 2$ **то**
        $M_l := \varnothing$;
        для каждого $v_l \in Desc$ **выполнять**
            если $l(v_l) \neq \varnothing$ **то**
                к M_l добавить $l(v_l)$.
            иначе
                к M_l добавить decode_tens_node$(G, v_l, \mathbf{y})$;
        $l(v_r) :=$ MajorVote(M_l);
    иначе
        $S := 0$;
        для каждого $v_l \in Desc$ **выполнять**
            $S := S +$ decode_tens_node$(G, v_l, \mathbf{y}) \bmod q$;
        $l(v_r) := S$;
Пометить вершину v_r меткой $l(v_r)$.
вернуть $l(v_r)$

Первый из этих алгоритмов – decode_tens_node помечает вершину v_r графа $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ элементом из $\mathbb{F}_q$. Эти метки будем называть декодирующими метками. Вычисление меток выполняется по следующему правилу.

Для вершин первого и второго типов, не принадлежащих двойственному коду, значение декодирующей метки получается голосованием по меткам потомков с помощью алгоритма MajorVote. Для вершин третьего типа, не принадлежащих двойственному коду, вычисляется сумма декодирующих меток потомков. Для вершин, помеченных векторами из двойственного кода, значения декодирующих меток получаются вычислением скалярного произведения векторов, которыми помечены эти вершины, на зашумленный кодовый вектор. Результатом работы алгоритма является значение декодирующей метки для вершины v_r.

Алгоритм 14 decode_tens_bit

Исходные параметры: $\mathbf{y}$ – зашумленный кодовый вектор, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$ – декодирующий граф для кода $\overline{C(D)}$, $v_0(\mathbf{b}^1 \otimes \mathbf{b}^2)$ – исток графа G.

Результат: $\widetilde{\mathbf{x}}(\in \mathbb{F}_q^n)$ – зашумленный кодовый вектор с декодированной координатой, соответствующей вектору, которым помечен исток v_0 графа G.

1: $l := \mathsf{decode\_tens\_node}(G, v_0, \mathbf{y})$;
2: $\widetilde{\mathbf{x}} = \mathbf{y} + \mathbf{b}_l$; ▷ где $\mathbf{b}_l$ – вектор $\mathbf{b} = \mathbf{b}^1 \otimes \mathbf{b}^2$, в котором ненулевая координата заменена на l.
3: **вернуть** $\widetilde{x}$

Алгоритм decode_tens_bit декодирует координату зашумленного вектора, соответствующую базисному вектору $\mathbf{b}^1 \otimes \mathbf{b}^2$. Сначала, с помощью алгоритма decode_tens_node вычисляется значение метки для истока графа $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}^1 \otimes \mathbf{b}^2)$. После чего, в соответствии со значением этой метки, декодируется координата зашумленного вектора.

Декодирование кодового слова

В [39] для групповых кодов построен алгоритм CloneTree, позволяющий по декодирующему дереву с одной меткой у корня построить декодирующее дерево для корня с любой другой меткой. Приведем формальное описание нового алгоритма CloneGraph, который является модификацией алгоритма CloneTree на

случай использования декодирующего графа. Алгоритм CloneGraph по графу $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b})$ строит граф $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}')$, где $\mathbf{b},\mathbf{b}' \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}$.

Алгоритм 15 CloneGraph

Исходные параметры: $\mathcal{G}$, $G(\mathcal{S}_1\times\mathcal{S}_2, \mathbf{b})$ – декодирующий граф для кода $\overline{C(D)}$ и координаты, соответствующей вектору $\mathbf{b}$, $\mathbf{b}'$.
Результат: $\mathcal{G}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}')$.
1: $\mathcal{G}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}') := \mathcal{G}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b})$
2: Найти $g(\in \mathcal{G} \times \mathcal{H})$ такой, что $(g,\mathbf{b}) = \mathbf{b}'$
3: **для каждого** метки $\mathbf{p}$ графа $\mathcal{G}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}')$ **выполнять**
4: $\quad \mathbf{p} := (g,\mathbf{p})$ ▷ Действие элементом g на $\mathbf{p}$ по правилу (1.7)
5: **вернуть** $\mathcal{G}$, $G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}')$

Таким образом, набор декодирующих графов

$$G(\mathcal{S}_1 \times \mathcal{S}_2) = \{G(\mathcal{S}_1 \times \mathcal{S}_2, \delta_{(g,h)})\}_{\delta_{(g,h)} \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}}$$

для группового кода $\overline{C(D)}$ может быть построен по одному графу. Именно, для любой базисной функции $\delta_{(g,h)} \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}$:

$$G(\mathcal{S}_1 \times \mathcal{S}_2, \delta_{(g,h)}) = \mathsf{CloneGraph}(\mathcal{G} \times \mathcal{H}, G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{1}_\mathcal{G} \otimes \mathbf{1}_\mathcal{H}), \delta_{(g,h)}).$$

Алгоритм decode_tensor_vector декодирования зашумленного кодового вектора $\mathbf{x}$, в котором каждая координата декодируется с помощью алгоритма decode_tens_bit, построен по аналогии с алгоритмом Decoder3 из [39].

Алгоритм 16 decode_tensor_vector

Исходные параметры: принятый вектор $\mathbf{x} = \mathbf{c} + \mathbf{e}$, набор декодирующих графов $G(\mathcal{S}_1 \times \mathcal{S}_2)$.
Результат: вектор $\mathbf{c}'$ – результат декодирования.
1: **для каждого** $\mathbf{b} \in B^{\mathbb{F}_q(\mathcal{G}\times\mathcal{H})}$ **выполнять**
2: $\quad x_\mathbf{b} := x_\mathbf{b} - \mathsf{decode\_tens\_bit}(\mathbf{x}, G(\mathcal{S}_1 \times \mathcal{S}_2, \mathbf{b}))$
3: **вернуть** $\mathbf{c}'$

2.3.3 Вероятностное декодирование

Особая структура D-кодов и наличие вероятностных декодеров для кодов Рида–Маллера позволяют построить вероятностный декодер и для D-кодов. Использование вероятностных декодеров позволяет исправлять гораздо больше ошибок по сравнению с гарантированными декодерами, что, в свою очередь, может уменьшить размер открытого ключа соответствующей криптосистемы и/или повысить ее стойкость к атакам на шифрограмму. В данном разделе построены два вероятностных декодера: декодер с использованием декодирования по информационным совокупностям (ДИС) (алгоритм ISDDecoder) и декодер на основе декодирования кодов–произведений, в которых кодовое слово может быть представлено в виде матрицы (алгоритм MatrixDecoder). Построенные декодеры программно реализованы на языке C++, с реализацией можно ознакомиться в [58] (см. `https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/McED%20Enc%20Dec%20Prob`). Результаты данного раздела опубликованы в [76].

Для описания вероятностных декодеров нам потребуется полиномиальный алгоритм $\mathsf{ExitCond}(G,\mathbf{m}',\mathbf{z},\mathsf{op}_1,\mathsf{op}_2)$ проверки правильности информационного вектора $\mathbf{m}' \in \mathbb{F}_2^k$, получаемого в ходе декодирования алгоритмом DDecoder зашумленного вектора $\mathbf{z} \in \mathbb{F}_2^n$ D-кода, заданного порождающей матрицей G. Этот алгоритм возвращает 1, если $\mathbf{m}'$ соответствует $\mathbf{z}$, и возвращает 0 в противном случае. Алгоритм ExitCond будет использоваться как параметр алгоритма декодирования DDecoder для D-кода. Заметим, что вектор $\mathbf{m}'$ вычисляется внутри алгоритма декодирования, а так как перед проверкой к вектору $\mathbf{m}'$ в ряде случаев должно быть применено некоторое преобразование (такими преобразованиями могут быть, например, умножение вектора на матрицу, отбрасывание фиксированных координат и т.п.), четвертым параметром алгоритма ExitCond является преобразование op_1 вектора $\mathbf{m}'$ перед проверкой. Например, при шифровании вес добавляемого вектора ошибок может быть фиксированным. В этом

случае алгоритм проверки соответствия вектора $\mathbf{m}'$ вектору $\mathbf{z}$ может заключаться в сравнении фактического и ожидаемого количества ошибок, то есть в проверке правильности равенства $\mathrm{wt}(\mathbf{z} - \mathbf{m}'G) = t$ (в этом случае алгоритм op_1 реализует тождественное преобразование: $\mathsf{op}_1(\mathbf{m}') = \mathsf{id}(\mathbf{m}') = \mathbf{m}'$). Аналогичное преобразование может быть применено перед проверкой и к вектору $\mathbf{z}$; этому преобразованию отвечает пятый параметр op_2.

Отметим, что проверка только на основе веса может быть ложно положительной при $t > \lfloor (d-1)/2 \rfloor$. Снизить вероятность такого события можно, например, в случае, когда известно, что добавляемая ошибка генерируется с помощью некоторого детерминированного инъективного алгоритма $\mathsf{G} : \mathbb{F}_2^k \to \mathcal{E}_{n,t}$. На рис. 2.3 приведены примеры двух алгоритмов проверки правильности информационного вектора: на основе только веса ошибки и на основе значения инъективного отображения G.

$\mathsf{ExitCond}_{\mathsf{wt}}(G,\mathbf{z},\mathbf{m}',\mathsf{op}_1,\mathsf{op}_2)$:

---

1. $\mathbf{e} = \mathbf{z} - \mathsf{op}_1(\mathbf{m}')G$
2. *если* $\mathrm{wt}(\mathbf{e}) = t$
3. ***вернуть*** 1
4. *иначе*
5. ***вернуть*** 0

$\mathsf{ExitCond}_{\mathsf{G}}(G,\mathbf{z},\mathbf{m}',\mathsf{op}_1,\mathsf{op}_2)$:

---

1. $\mathbf{e} = \mathsf{op}_2(\mathbf{z}) - \mathsf{op}_1(\mathbf{m}')G$
2. *если* $\mathrm{wt}(\mathbf{e}) = t$ *и* $\mathsf{G}(\mathsf{op}_1(\mathbf{m}')) = \mathbf{e}$
3. ***вернуть*** 1
4. *иначе*
5. ***вернуть*** 0

Рисунок 2.3 — Алгоритмы проверки правильности информационного вектора: $\mathsf{ExitCond}_{\mathrm{wt}}$ – на основе только веса вектора ошибок, $\mathsf{ExitCond}_{\mathsf{G}}$ – на основе веса и значения инъективного отображения G

Введем обозначение для вероятностей ξ_{G}, ξ_{wt} ложно положительной проверки правильности информационного вектора для соответствующих алго-

ритмов $\mathsf{ExitCond}_{\mathsf{G}}$, $\mathsf{ExitCond}_{\mathsf{wt}}$:

$$\xi_{\mathsf{A}} = \Pr(\mathsf{ExitCond}_{\mathsf{A}}(G,\mathbf{m}',\mathbf{z},\mathsf{op}_1,\mathsf{op}_2) = 1 | \mathbf{m}' \neq \mathbf{m}), \mathsf{A} \in \{\mathsf{wt},\mathsf{G}\}. \qquad (2.26)$$

Стоит отметить, что во всех проведенных в работе экспериментах проверка только на основе веса не дала ложно положительных ответов, хотя теоретические оценки вероятности ложно положительного ответа в этом случае отсутствуют. В то же время проверка на основе инъективного отображения G дает возможность оценить эту вероятность. Действительно, $\mathsf{ExitCond}_{\mathsf{A}}(G,\mathbf{m}',\mathbf{z},\mathsf{op}_1,\mathsf{op}_2) = 1$ при $\mathbf{m}' \neq \mathbf{m}$ означает, что

$$\mathsf{op}_2(\mathbf{z}) = \mathsf{op}_1(\mathbf{m})G + \mathsf{G}(\mathsf{op}_1(\mathbf{m})) = \mathsf{op}_1(\mathbf{m}')G + \mathsf{G}(\mathsf{op}_1(\mathbf{m}')).$$

Откуда получаем, что $\mathsf{G}(\mathsf{op}_1(\mathbf{m}')) - \mathsf{G}(\mathsf{op}_1(\mathbf{m})) \in C$. Если при $n - k > k$ отображение G можно выбрать так, что для разных $\mathbf{m}$ и $\mathbf{m}'$ векторы $\mathsf{G}(\mathsf{op}_1(\mathbf{m}))$ и $\mathsf{G}(\mathsf{op}_1(\mathbf{m}'))$ принадлежат разным смежным классам фактор–множества $\mathbb{F}_2^n/C$, то $\xi_{\mathsf{G}} = 0$. Построение такого отображения составляет отдельную задачу, тем не менее представляется, что можно построить такое отображение G, для которого вероятностью ξ_{G} можно пренебречь.

Отметим, что проверки правильности декодирования нередко используются для раннего выхода в итерационных алгоритмах декодирования, используемых при защите от помех. Однако в таких алгоритмах обычно выполняется проверка того, что декодированный вектор является кодовым, но проверка правильности информационного вектора не выполняется. При синтезе схем KEM, как отмечено в разделе 1.1, важную роль играет значение вероятности (1.2) ошибочной декапсуляции. Поэтому при выборе конкретного алгоритма проверки следует исходить из допустимой вероятности этой ошибки. Таким условием для механизмов KEM является условие (1.3): для IND–CCA–стойкости λ_{KEM} механизма KEM необходимо, чтобы $\xi_{\mathsf{A}} \leqslant 2^{-\lambda_{\mathsf{KEM}}}$, а для отказоустойчивости γ механизма KEM с IND–CPA–стойкостью необходимо выполнение неравенства $\xi_{\mathsf{A}} \leqslant 10^{-\gamma}$. Далее в работе будем полагать, что алгоритм проверки выбирается, исходя из этих условий. Отметим, что проверка правильности информационного вектора может проводиться также, например, на основе контрольной

суммы, являющейся частью информационного сообщения. Однако в случае, когда такая проверка применяется в алгоритме декодирования, используемом при синтезе KEM, необходимо обеспечить соблюдение условия (1.1). Это обусловлено тем, что из OW–стойкости схемы PKE не вытекает такая же OW–стойкость схемы PKE, в которой часть сообщения отводится под контрольную сумму. В настоящей работе проверки на основе контрольной суммы не рассматриваются.

Декодер ISDDecoder

Опишем первый из построенных вероятностных декодеров – ISDDecoder.

Из вида (2.12) с учетом обозначений (2.14) в случае использования кодов Рида–Маллера D-код представим в следующем виде:

$$\overline{C(D)} = \sum_{i=1}^{s} \mathrm{RM}_2(\overline{r}^1_{k_i}, m_1) \otimes \mathrm{RM}_2(\overline{r}^2_{\ell_i}, m_1). \tag{2.27}$$

Пусть C – код вида (2.27). Из представления (2.27), обозначений (2.14) и того факта, что последовательность $(k_i)_{i=1}^s$ является возрастающей, а $(\ell_i)_{i=1}^s$ – убывающей, следует, что

$$C \subseteq \mathbb{F}_2^{n_1} \otimes \mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2) = \underbrace{\mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2) \times \cdots \times \mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2)}_{n_1}. \tag{2.28}$$

Следовательно, кодовое слово кода C представляет собой конкатенацию кодовых слов кода $\mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2)$. Поэтому декодирование кода C может заключаться в применении некоторого известного декодера

$$\mathsf{DecRow} : \{0,1\}^{2^{m_2}} \to \mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2) \cup \{\perp\}$$

кода $\mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2)$ к каждому из n_1 блоков длины n_2 (см. алгоритм 17).

Отметим, что в качестве DecRow могут использоваться как гарантированные, так и вероятностные алгоритмы декодирования, например, упомянутые в разделе 1.4.2 декодеры Рида, Мэсси, Сидельникова–Першакова и Йе–Аббе. При

Алгоритм 17 BlockDecoder – блочный декодер D-кода

Исходные параметры: $\mathbf{z} = \mathbf{z}_1 \parallel \cdots \parallel \mathbf{z}_{n_1}$, DecRow

1: $i = 1$, $\mathbf{z}' = \varnothing$
2: **пока** $i \leqslant n_1$ **выполнять**
3: $\quad \mathbf{z}' = \mathbf{z}' || \mathsf{DecRow}(\mathbf{z}_i)$
4: $\quad i = i + 1$
5: **вернуть** $\mathbf{z}'$

этом в обоих случаях алгоритм 17 может вернуть неправильный результат, то есть некоторое количество блоков $\mathbf{z}_i$ зашумленного кодового вектора $\mathbf{z}$ будут декодированы неправильно. Если блок был декодирован правильно, то назовем его *хорошим*, в противном случае – *плохим*.

Отметим, что в худшем случае после работы алгоритма 17 не известно, какие блоки хорошие, а какие – плохие. Для того, чтобы повысить вероятность правильного декодирования, целесообразно после блочного декодирования воспользоваться декодированием по информационным совокупностям. Отметим, что такой подход – блочное декодирование и применение ДИС – будет использован в следующей главе при разработке комбинированной атаки на шифртекст для слабых ключей криптосистемы на D-кодах. Здесь этот подход использован для декодирования и представлен в алгоритме 18.

Напомним, что информационной совокупностью линейного $[n,k]_q$–кода называется такое множество $\boldsymbol{\tau} \subset \{1,...,n\}$ мощности k, что столбцы любой порождающей матрицы этого кода с номерами из $\boldsymbol{\tau}$ линейно независимы. Обычно при декодировании по информационным совокупностям (см., например, оригинальную работу [43]) случайным образом выполняется поиск такой информационной совокупности $\boldsymbol{\tau}$, чтобы номера ненулевых координат вектора ошибки не принадлежали $\boldsymbol{\tau}$. В [77] и приведенном далее алгоритме 18 выполняется поиск номеров хороших *блоков*, на основе которых далее может быть построена информационная совокупность.

Для описания алгоритма декодирования нам понадобится блочное представление порождающей матрицы D-кода. Из (2.27) вытекает, что любая

порождающая $(k \times n)$–матрица G этого кода представима в виде

$$G = [G_1|G_2|...|G_{n_1}], \tag{2.29}$$

где строки $(k \times n_2)$–матрицы (блока) G_i – кодовые слова кода $\mathrm{RM}_2(\overline{r}^2_{\ell_1},m_2)$, $i \in [\![1,n_1]\!]$. Через τ_0 обозначим заранее известную фиксированную информационную совокупность кода C, $M_0 = \Pi_{\tau_0}(G)$.

Параметрами алгоритма 18, помимо принятого слова $\mathbf{z}$, декодера DecRow, являются K – количество выбираемых блоков для применения ДИС, I_{max} – максимальное количество итераций ДИС (под одной итерацией понимаются шаги с 10 по 18) и ExitCond – условие раннего выхода из ДИС. Отметим, что параметр K выбирается исходя из особенностей кода, в частности, необходимо, чтобы вероятность того, что выбранные K блоков составляют матрицу ранга k, была непренебрежимой. В данной работе в качестве нижней границы такой вероятности для выбора параметра K использовалось значение 0.01. Параметр I_{max} выбирается, во–первых, исходя из вычислительных возможностей, а во–вторых, исходя из допустимой вероятности ошибочного декодирования.

Оценим вероятность ошибочного декодирования (DFR) для алгоритма 18. В случае $i \in \mathbb{N}$ итераций ДИС эту вероятность можно вычислить по следующей формуле:

$$\mathrm{DFR}_1(i) = 1 - (P^{(0)}_{dec} + \sum_{j=1}^{n_1}(1 - (1 - P_1^{(j)}P_2)^i)P^{(j)}_{dec}), \tag{2.30}$$

где $P_1^{(j)} = C^K_{n_1-j}/C^K_{n_1}$ – вероятность выбора на шаге 11 K хороших блоков из n_1 блоков при наличии j плохих блоков, P_2 – вероятность того, что выбранные K блоков образуют матрицу ранга k (шаг 13), а $P^{(j)}_{dec}$ – вероятность появления j плохих блоков в зашумленном кодовом векторе. Вероятность $P^{(j)}_{dec}$ рассчитывается по формуле:

$$P^{(j)}_{dec} = C^j_{n_1}P^j_{bad}(1 - P_{bad})^{n_1-j}, \tag{2.31}$$

где P_{bad} – вероятность того, что некоторый фиксированный блок станет плохим после применения на шаге 3 алгоритма BlockDecoder. Заметим, что P_{bad}

Алгоритм 18 ISDDecoder – декодер D-кода с применением ДИС

Исходные параметры: $\mathbf{z}; \tau_0, G, \mathsf{ExitCond}, \mathsf{op}_1, \mathsf{op}_2, \mathsf{DecRow}, K, I_{max}$

1: $result = \perp$; $\mathbf{m}' = \varnothing$
2: $\mathbf{z}' = (\mathbf{z}'_1 \parallel ... \parallel \mathbf{z}'_{n_1}) = \mathsf{BlockDecoder}(\mathbf{z}, \mathsf{DecRow})$
3: $B := \{j \in [\![1,n_1]\!] : \mathbf{z}'_j = \perp\}$
4: **если** $(\cup_{j \in B}[\![(j-1)\cdot n_2 + 1, j \cdot n_2]\!]) \cap \tau_0 = \varnothing$ **то**
5: $\quad \mathbf{m}' = \Pi_{\tau_0}(\mathbf{z}')M_0^{-1}$ ▷ попытка декодирования без ДИС
6: **если** $\mathsf{ExitCond}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1$ **то**
7: $\quad result = \mathbf{m}'$
8: **иначе**
9: $\quad i = I_{max}$
10: $\quad$ **пока** $I > 0$ **и** $result = \perp$ **выполнять** ▷ попытка декодирования с ДИС
11: $\quad\quad$ Выбрать случайно $W \subset [\![1,n_1]\!] \setminus B$, $|W| = K$
12: $\quad\quad G' = \Pi_{\omega}(G)$, где $\omega = \cup_{j \in W}[\![(j-1)\cdot n_2 + 1, j \cdot n_2]\!]$
13: $\quad\quad$ **если** $\mathrm{rank}(G') = k$ **то**
14: $\quad\quad\quad$ Найти такое $\tau \subset \omega$, $|\tau| = k$, $\mathrm{rank}(\Pi_\tau(G)) = k$
15: $\quad\quad\quad \mathbf{m}' = \Pi_\tau(\mathbf{z}')(\Pi_\tau(G))^{-1}$
16: $\quad\quad\quad$ **если** $\mathsf{ExitCond}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1$ **то**
17: $\quad\quad\quad\quad result = \mathbf{m}'$
18: $\quad\quad i = i - 1$
19: **вернуть** $result$

зависит от кода, декодера DecRow и числа ошибок t. В настоящей работе P_{bad} оценивается *экспериментально*, соответствующие оценки будут представлены далее в этой главе.

Поясним формулу (2.30). Значение $\mathrm{DFR}_1(i)$ получается вычитанием из единицы вероятности правильного декодирования за i итераций ДИС, которая, в свою очередь, рассчитывается как сумма двух вероятностей. Первая – вероятность $P^{(0)}_{dec}$ того, что на шаге 3 после работы алгоритма $\mathsf{BlockDecoder}$ будет получено правильное кодовое слово. Вторая – вероятность правильного декодирования после i итераций ДИС, в которой учитываются случаи появления j плохих блоков, где $j \in [\![1,n_1]\!]$. При этом, $(1 - P_1^{(j)}P_2)^i$ – вероятность того, что за i итераций не будет найдено набора из K блоков матрицы G в представлении (2.29), соответствующих хорошим блокам вектора $\mathbf{z}$ и составляющих матрицу ранга k.

Оценим вычислительную эффективность алгоритма 18. Пусть T_{block} – среднее время, необходимое для выполнения алгоритма BlockDecoder, а T_{ISD} – среднее время, необходимое для выполнения одной итерации ДИС, тогда среднее время T_1, необходимое для правильного декодирования алгоритмом 18, можно вычислить по формуле:

$$T_1 = T_{block} + \sum_{i=1}^{I_{max}} iT_{\text{ISD}} \sum_{j=0}^{n_1} (1 - P_1^{(j)} P_2)^{i-1} P_1^{(j)} P_2 P_{dec}^{(j)}. \tag{2.32}$$

Поясним формулу (2.32). Для нахождения среднего времени T_1 фактически необходимо ко времени T_{block} добавить среднее время работы ДИС, которое в свою очередь напрямую зависит от среднего количества итераций. Поэтому во втором слагаемом в (2.32) вычисляется математическое ожидание времени работы ДИС, где для каждого слагаемого в сумме по количеству необходимых итераций i вычисляется вероятность того, что ДИС закончит свою работу после i–й итерации. При этом, благодаря сумме по j учитываются все возможные варианты количества плохих блоков и вероятности их появления.

Декодер MatrixDecoder

Далее опишем второй построенный вероятностный декодер – MatrixDecoder. Из вида (2.27) следует, что

$$\begin{aligned} C &\subseteq \left(\mathbb{F}_2^{n_1} \otimes \mathrm{RM}_2(\overline{r}_{\ell_1}^2, m_2)\right) \cap \left(\mathrm{RM}_2(\overline{r}_{k_s}^1, m_1) \otimes \mathbb{F}_2^{n_2}\right) \\ &\subseteq \mathrm{RM}_2(\overline{r}_{k_s}^1, m_1) \otimes \mathrm{RM}_2(\overline{r}_{\ell_1}^2, m_2), n_i = 2^{m_i}, i = 1{,}2. \end{aligned}$$

Следовательно, алгоритм декодирования испорченного кодового слова $\mathbf{z}$ кода C может быть основан на способе декодирования кодов–произведений:

1. представить $\mathbf{z}$ в виде $(n_1 \times n_2)$–матрицы, строки которой – испорченные кодовые слова кода $\mathrm{RM}_2(\overline{r}_{\ell_1}^2, m_2)$, а столбцы – испорченные кодовые слова кода $\mathrm{RM}_2(\overline{r}_{k_s}^1, m_1)$;

2. применить декодер по строкам, а затем к результату декодирования применить декодер по столбцам.

Отметим, что декодеры для строк и столбцов могут быть разными. Например, к строкам может быть применен декодер из работы [57] для исправления большого числа ошибок, а к столбцам может быть применен более вычислительно простой декодер, например, мажоритарный, так как ожидается, что строчный декодер исправит «почти все» ошибки, и для декодера по столбцам достаточно простого алгоритма.

Заметим, что декодер по строкам может вернуть символ $\perp$, в этом случае соответствующий блок может быть заменен на вектор $(\star,\ldots,\star)$ длины n_2, где $\star$ – символ стирания. Такая замена позволит, например, использовать возможность исправления стираний при декодировании по столбцам.

Заметим также, что всегда можно переставить координаты вектора $\mathbf{z}$ так, что вектор с переставленными координатами будет представлять собой конкатенацию n_2 зашумленных кодовых слов кода $\mathrm{RM}_2(\overline{r}^1_{k_s},m_1)$ длины n_1 каждое. Такая перестановка зависит только от параметров n_1 и n_2; обозначим ее $\boldsymbol{\pi}$. Следовательно, при декодировании вектора $\mathbf{z}$ необязательно представлять его в виде матрицы: при декодировании по строкам к вектору $\mathbf{z}$ применяется алгоритм BlockDecoder с декодером по строкам DecRow в качестве второго аргумента, а при декодировании по столбцам к вектору $\boldsymbol{\pi}(\mathbf{z}')$ применяется алгоритм BlockDecoder, в котором вторым аргументом является декодер по столбцам DecCol; здесь $\mathbf{z}'$ обозначает результат после декодирования по строкам. Описанный способ декодирования приведен в алгоритме 19.

Если предполагать, что декодер по столбцам является гарантированным, то есть исправляет ошибки в пределах половины кодового расстояния, то вероятность ошибочного декодирования для декодера, представленного в алгоритме 19, можно вычислить как вероятность появления плохих блоков после работы строчного декодера в количестве, превышающем половину кодового расстоя-

Алгоритм 19 MatrixDecoder – декодер D-кода без применения ДИС

Исходные параметры: $\mathbf{z}; \tau_0, G, \mathsf{ExitCond}, \mathsf{op}_1, \mathsf{op}_2, \mathsf{DecRow}, \mathsf{DecCol}$

1: $result = \perp$
2: $\mathbf{z}' = (\mathbf{z}'_1 \parallel ... \parallel \mathbf{z}'_{n_1}) = \mathsf{BlockDecoder}(\mathbf{z}, \mathsf{DecRow})$
3: $B_r := \{j \in [\![1,n_1]\!] : \mathbf{z}'_j = \perp\}$
4: **если** $(\cup_{j \in B_r} [\![(j-1) \cdot n_2 + 1, j \cdot n_2]\!]) \cap \tau_0 = \varnothing$ **то**
5: $\quad \mathbf{m}' = \Pi_{\tau_0}(\mathbf{z}') M_0^{-1}$ ▷ первая попытка декодирования
6: $\quad$ **если** $\mathsf{ExitCond}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1$ **то**
7: $\quad\quad result = \mathbf{m}'$
8: **иначе**
9: $\quad$ Заменить в $\mathbf{z}'$ блоки с номерами из B_r на вектор $(\star,...,\star)$ длины n_2.
10: $\quad \mathbf{z}'' = (\mathbf{z}''_1 \parallel ... \parallel \mathbf{z}''_{n_2}) = \mathsf{BlockDecoder}(\pi(\mathbf{z}'), \mathsf{DecCol})$
11: $\quad B_c := \{j \in [\![1,n_2]\!] : \mathbf{z}''_j = \perp\}$.
12: $\quad$ **если** $B_c = \varnothing$ **то**
13: $\quad\quad \mathbf{m}' = \Pi_{\tau_0}(\pi^{-1}(\mathbf{z}'')) M_0^{-1}$ ▷ вторая попытка декодирования
14: $\quad\quad$ **если** $\mathsf{ExitCond}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1$ **то**
15: $\quad\quad\quad result = \mathbf{m}'$
16: **вернуть** $result$

ния кода $\mathrm{RM}_2(\overline{r}^1_{k_s}, m_1)$. То есть

$$\mathrm{DFR}_2 = \sum_{j=\lfloor (d_2-1)/2 \rfloor + 1}^{n_1} P^{(j)}_{dec}, \tag{2.33}$$

где d_2 – минимальное кодовое расстояние кода $\mathrm{RM}_2(\overline{r}^1_{k_s}, m_1)$, $P^{(j)}_{dec}$ имеет вид (2.31).

Отметим, что алгоритм 19, в отличии от алгоритма 18, имеет постоянное время выполнения. Оно складывается из времени работы BlockDecoder с декодером DecRow на шаге 2 (T_{DecRow}) и времени работы BlockDecoder с декодером DecCol на шаге 10 (T_{DecCol}), то есть

$$T_2 = T_{\mathsf{DecRow}} + T_{\mathsf{DecCol}}. \tag{2.34}$$

Оценка эффективности декодеров ISDDecoder и MatrixDecoder

В данном разделе получены результаты по оценке эффективности работы построенных вероятностных декодеров ISDDecoder и MatrixDecoder. Для этого были проведены эксперименты по оценке вероятности ошибочного декодирования DFR и времени декодирования построенных декодеров. В качестве DecRow в настоящей работе в обоих алгоритмах используется декодер Йе–Аббе [57], корректирующая способность которого выше, чем у декодера Сидельникова–Першакова. Декодер Йе–Аббе используется также в качестве алгоритма DecCol в MatrixDecoder. Для применения условия проверки ExitCond, которое используется в алгоритмах декодирования ISDDecoder и MatrixDecoder, при расшифровании в качестве алгоритма op_1 будем использовать алгоритм $\mathsf{mul}_{S^{-1}}$, реализующий умножение вектора на матрицу S^{-1}, так как при проверке необходимо «снять» преобразование информационного вектора, полученное с помощью секретного ключа S, а в качестве алгоритма op_2 – алгоритм mul_P, реализующий умножение вектора на матрицу P.

Как было неоднократно сказано, декодеры ISDDecoder и MatrixDecoder характеризуются ненулевой вероятностью ошибочного декодирования (см. (2.30) и (2.33) соответственно). Из (2.30) вытекает, что $\lim_{i\to\infty} \mathrm{DFR}_1(i) = 0$, то есть при увеличении числа итераций ДИС в ISDDecoder при фиксированном числе ошибок t вероятность ошибочного декодирования можно сделать сколь угодно малой. Однако в этом случае возрастает время декодирования и, как следствие, время расшифрования. Поэтому количество итераций ДИС ограничено сверху числом I_{max}, которое зависит от максимально допустимого времени расшифрования и среднего времени выполнения одной итерации ДИС, которые, в свою очередь, зависят от прикладной задачи и реализации декодера соответственно. (В настоящей работе при исследовании характеристик декодера ISDDecoder значение I_{max} не превышает 10^6.) Для декодера MatrDecoder, как следует из (2.33), вероятность ошибочного декодирования зависит только от вероятности

P_{bad}, то есть от числа добавляемых ошибок при шифровании, поэтому алгоритмически снизить вероятность ошибочного декодирования не удается. Однако, как отмечено выше, в отличие от ISDDecoder декодер MatrDecoder является декодером с постоянным временем декодирования. Это может быть преимуществом декодера MatrDecoder, так как в случае, когда для криптографического протокола актуальны атаки на кодовую криптосистему по побочным каналам, в частности, атаки по времени декодирования, предпочтение отдается декодерам с постоянным временем декодирования.

Для удобства, при фиксированном D-коде C, числе ошибок t и числе K хороших блоков минимальное число итераций ДИС, обеспечивающее вероятность ошибочного декодирования (2.30) в алгоритме ISDDecoder не более $10^{-\gamma}$, обозначим I_γ:

$$I_\gamma = \min\{i \in \mathbb{N} : \mathrm{DFR}_1(i) \leqslant 10^{-\gamma}\}. \tag{2.35}$$

Найденное значение I_γ предполагается использовать в качестве I_{max} в алгоритме ISDDecoder для ограничения сверху вероятности ошибочного декодирования и, как следствие, обеспечения требуемой отказоустойчивости γ.

В таблице 2 представлены параметры D-кодов вида (2.27), выбранные для использования в криптосистеме типа Мак–Элиса. Эти коды получены следующим образом. В качестве $\mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2)$ из вида (2.28) выбраны коды $\mathrm{RM}_2(2,7)$, $\mathrm{RM}_2(3,7)$, $\mathrm{RM}_2(2,8)$, $\mathrm{RM}_2(3,8)$, $\mathrm{RM}_2(2,9)$, обладающие высокой корректирующей способностью при использовании декодера Йе–Аббе [57]. Для каждого из этих кодов, перебрав значения параметра m_1, найдены D-коды вида (2.27) такие, что соответствующие им криптосистемы, во–первых, являются стойкими к комбинированной атаке, которая будет построена в следующей главе, а во–вторых, стойкость λ_{OW} (см. (1.23)) к классической атаке декодированием по информационным совокупностям находится в пределах стойкости систем из таблицы 1. При этом, для начальной оценки стойкости λ_{OW} использовалось значение t, полученное умножением количества ошибок, исправляемых декодером Йе–Аббе для кода $\mathrm{RM}_2(\overline{r}^2_{\ell_1}, m_2)$ (с вероятностью ошибочного декодирования не

более 10^{-4}), на количество блоков 2^{m_1}. (В дальнейшем оценка значения λ_{OW} уточнялась путем подбора для построенных декодеров количества добавляемых ошибок t при экспериментальных вычислениях вероятности ошибочного декодирования, результаты которых содержатся в таблицах 3,5.) В итоговую выборку попали D-коды с минимальными значениями размера открытого ключа соответствующих систем типа Мак–Элиса. При этом в таблице 2 приводятся номер кода и обозначение $[(\hat{r}_1^1, \hat{r}_1^2), (\hat{r}_2^1, \hat{r}_2^2), ...]$ соответствующего D-кода, где

$$[(\hat{r}_1^1, \hat{r}_1^2), (\hat{r}_2^1, \hat{r}_2^2), ...] = \mathrm{RM}_2(\hat{r}_1^1, m_1) \otimes \mathrm{RM}_2(\hat{r}_1^2, m_2) + \mathrm{RM}_2(\hat{r}_2^1, m_1) \otimes \mathrm{RM}_2(\hat{r}_2^2, m_2) + ..., \quad (2.36)$$

а также приведены параметры m_1, m_2 и $[n, k, d]$. Для автоматизации вышеописанного процесса реализован код на Python, с которым можно ознакомится в [58] (см. `https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/D-codes%20for%20Prob%20Decoders`).

Таблица 2 — D-коды

Номер кода	D-код	$[n,k,d]$	m_1	m_2
1	$[(1,2),(2,1)]$	[16384,376,2048]	5	9
2	$[(1,2),(2,1),(3,0)]$	[16384,414,2048]	6	8
3	$[(0,2),(3,1)]$	[16384,406,1024]	6	8
4	$[(1,2),(3,1)]$	[8192,402,512]	5	8
5	$[(1,2),(4,1)]$	[8192,603,256]	6	7
6	$[(1,2),(3,1)]$	[16384,680,1024]	7	7
7	$[(1,2),(3,1)]$	[16384,476,1024]	5	9
8	$[(2,3),(3,0)]$	[8192,1498,256]	5	8
9	$[(2,3),(3,1)]$	[8192,1578,256]	5	8
10	$[(1,2),(3,1)]$	[16384,574,1024]	6	8
11	$[(2,2),(3,0)]$	[16384,876,1024]	7	7
12	$[(0,2),(4,1)]$	[16384,813,512]	7	7

Продолжение таблицы 2

Номер кода	D-код	$[n,k,d]$	m_1	m_2
13	$[(1,2),(3,1)]$	[32768,672,2048]	6	9
14	$[(2,3),(3,2)]$	[8192,1858,256]	5	8
15	$[(2,2),(3,1)]$	[16384,1121,1024]	7	7
16	$[(2,3),(3,0)]$	[16384,2066,512]	6	8
17	$[(2,2),(5,1)]$	[16384,1569,256]	7	7
18	$[(1,2),(4,1)]$	[32768,822,1024]	6	9
19	$[(2,3),(3,2)]$	[16384,2786,512]	6	8

В таблице 3 приводится экспериментальная оценка корректирующей способности декодера ISDDecoder для кодов из таблицы 2. Оценка выполнена следующим образом. Для каждого кода из таблицы 2 и заданного значения количества ошибок t производилась генерация случайного информационного вектора длины k и случайного вектора ошибок веса t. Затем информационный вектор кодировался и добавлялась ошибка, после чего полученный зашумленный кодовый вектор подавался на вход алгоритму BlockDecoder. После выполнения алгоритма происходил подсчет количества блоков зашумленного кодового слова, которые были декодированы неправильно (подсчет плохих блоков). Перечисленные действия повторялись указанное в таблице 3 количество раз (столбец «Количество экспериментов») и на основе итогового количества плохих блоков было рассчитано значение P_{bad} из (2.31) и затем $\mathrm{DFR}_1(0) = 1-(1-P_{bad})^{2^{m_1}}$ в соответствии с (2.30). Значение $\mathrm{DFR}_1(0)$ фактически означает вероятность ошибочного декодирования алгоритмом ISDDecoder до использования ДИС и может быть использовано для оценки адекватности выбранного количества ошибок для обеспечения требуемой отказоустойчивости. Также для каждого кода вычислены значения количества блоков K, выбираемых на шаге 11 алгоритма ISDDecoder, и соответствующие им значения вероятностей P_2 из (2.30). При этом значения K вычислены экспериментально. Именно, для

каждого кода в качестве K было выбрано минимальное количество блоков, при котором за 100 попыток составления подматрицы из K случайных блоков соответствующей порождающей матрицы, хотя бы одна попытка давала подматрицу ранга, равного размерности кода. При этом, помимо найденного значения K, использовались также значения $K+1$ и $K+2$ для демонстрации влияния на характеристики декодера. С помощью полученных значений, в соответствии с (2.30), было вычислено количество I_9 итераций ДИС, необходимых для обеспечения отказоустойчивости $\gamma = 9$ IND–CPA–стойкого механизма KEM. В дополнение в таблице указано максимальное за наблюдаемое количество экспериментов количество плохих блоков после декодирования зашумленного кодового вектора.

Как видно из таблицы 3, для большинства кодов используемые значения количества добавляемых ошибок t позволяют получить приемлемые значения вероятности $\mathrm{DFR}_1(0)$ и, как следствие, количества итераций I_9. Однако для кодов 2 и 3 вероятность $\mathrm{DFR}_1(0)$ получилась достаточно большой, что привело к значительному увеличению I_9. Также стоит отметить, что для кода 17, несмотря на относительно небольшую вероятность $\mathrm{DFR}_1(0)$, количество итераций I_9 все равно получается большим. Это происходит из–за того, что для кода 17 найденное значение K получается слишком большим относительно общего количества блоков (120 из 128), что приводит к тому, что вероятность нахождения требуемой информационной совокупности во время работы каждой итерации ДИС оказывается маленькой и, соответственно, требуется большее количество итераций для обеспечения отказоустойчивости $\gamma = 9$.

Таблица 3 — Оценка корректирующей способности декодера ISDDecoder

Номер кода	K	P_2	t	Кол-во экспери-ментов	Макс. кол-во плохих блоков	$\mathrm{DFR}_1(0)$	I_9
1	16	0.3233	4000	500000	2	0.00259	100
1	17	0.6313	4000	500000	2	0.00259	57
1	18	0.8301	4000	500000	2	0.00259	49
1	16	0.3233	3749	500000	1	0.000136	67
1	17	0.6313	3749	500000	1	0.000136	33
1	18	0.8301	3749	500000	1	0.000136	26
2	42	0.2064	3392	500000	5	0.266	67814
2	43	0.4634	3392	500000	5	0.266	44720
2	44	0.6617	3392	500000	5	0.266	47471
3	42	0.1995	3392	500000	5	0.266	70160
3	43	0.4572	3392	500000	5	0.266	45327
3	44	0.6558	3392	500000	5	0.266	47899
6	64	0.1528	1706	500000	2	0.0152	351
6	65	0.3763	1706	500000	2	0.0152	147
6	66	0.5552	1706	500000	2	0.0152	103
7	26	0.4935	3830	500000	1	0.000332	264
7	27	0.8251	3830	500000	1	0.000332	237
7	28	0.9642	3830	500000	1	0.000332	339
8	26	0.4878	612	50000	1	0.00337	901
8	27	0.8265	612	50000	1	0.00337	1059
8	28	0.9645	612	50000	1	0.00337	2270
9	26	0.4880	612	50000	1	0.00337	901
9	27	0.8238	612	50000	1	0.00337	1062

Продолжение таблицы 3

Номер кода	K	P_2	t	Кол-во экспери-ментов	Макс. кол-во плохих блоков	$\mathrm{DFR}_1(0)$	I_9
9	28	0.9645	612	50000	1	0.00337	2270
10	42	0.2072	3237	500000	4	0.101	6920
10	43	0.4565	3237	500000	4	0.101	4099
10	44	0.6628	3237	500000	4	0.101	3742
11	64	0.1598	1706	500000	2	0.0152	336
11	65	0.3660	1706	500000	2	0.0152	151
11	66	0.5648	1706	500000	2	0.0152	101
12	99	0.1772	1706	500000	2	0.0152	3717
12	100	0.3972	1706	500000	2	0.0152	1861
12	101	0.5871	1706	500000	2	0.0152	1420
13	42	0.2126	7687	500000	1	0.000772	231
13	43	0.4690	7687	500000	1	0.000772	113
13	44	0.6626	7687	500000	1	0.000772	88
13	42	0.2126	7772	500000	2	0.00131	275
13	43	0.4690	7772	500000	2	0.00131	135
13	44	0.6626	7772	500000	2	0.00131	105
14	26	0.4883	673	50000	2	0.0147	3827
14	27	0.8288	673	50000	2	0.0147	5088
14	28	0.9667	673	50000	2	0.0147	18011
15	64	0.1550	1706	500000	2	0.0152	346
15	65	0.3624	1706	500000	2	0.0152	152
15	66	0.5592	1706	500000	2	0.0152	102
16	42	0.2090	1292	50000	2	0.0177	941

Продолжение таблицы 3

Номер кода	K	P_2	t	Кол-во экспери-ментов	Макс. кол-во плохих блоков	$\mathrm{DFR}_1(0)$	I_9
16	43	0.4511	1292	50000	2	0.0177	508
16	44	0.6626	1292	50000	2	0.0177	407
17	120	0.3592	1706	500000	2	0.0152	329479
17	121	0.6908	1706	500000	2	0.0152	342659
17	122	0.8802	1706	500000	2	0.0152	627527
18	57	0.4184	7279	500000	1	0.0000356	229
18	58	0.7468	7279	500000	1	0.0000356	149
18	59	0.9161	7279	500000	1	0.0000356	146
19	42	0.2060	1083	50000	1	0.00094	255
19	43	0.4562	1083	50000	1	0.00094	125
19	44	0.6560	1083	50000	1	0.00094	95

В таблице 4 приводится оценка времени декодирования с помощью алгоритма ISDDecoder кодов из таблицы 2. Для каждого набора параметров экспериментов из таблицы 3, а именно, D-кода, количества блоков K и значения количества ошибок t, приводится время декодирования в миллисекундах. В частности, T_{block} – среднее время, необходимое для выполнения алгоритма BlockDecoder на шаге 2, T_{ISD} – среднее время, необходимое для выполнения одной итерации ДИС (шаги 10–18), T_1 – среднее время декодирования с помощью алгоритма ISDDecoder, вычисленное в соответствии с (2.32), а T_1' – время выполнения алгоритма для обеспечения отказоустойчивости $\gamma = 9$ в худшем случае (без раннего выхода из ДИС), то есть $T_1' = T_{block} + T_{\mathrm{ISD}} I_9$.

Таблица 4 — Оценка времени декодирования алгоритмом ISDDecoder

Номер кода	K	t	T_{block}	T_{ISD}	T_1'	T_1
1	16	4000	1718	33	5018	1821
1	17	4000	1742	31	3509	1792
1	18	4000	1726	34	3392	1768
1	16	3749	1718	33	3929	1821
1	17	3749	1742	31	2765	1792
1	18	3749	1726	34	2610	1767
2	42	3392	897	42	2849085	1268
2	43	3392	884	40	1789684	1049
2	44	3392	875	42	1994657	1003
3	42	3392	896	44	3087936	1298
3	43	3392	895	43	1949956	1075
3	44	3392	890	39	1868951	1010
6	64	1706	428	86	30614	1000
6	65	1706	430	86	13072	663
6	66	1706	422	89	9589	585
7	26	3830	1725	49	14661	1825
7	27	3830	1706	50	13556	1767
7	28	3830	1707	51	18996	1761
8	26	612	43635	208	231043	44068
8	27	612	43615	206	261769	43869
8	28	612	43114	180	451714	43306
9	26	612	44601	193	218494	45003
9	27	612	44765	193	249731	45004
9	28	612	45051	192	480891	45255
10	42	3237	868	62	429908	1236

Продолжение таблицы 4

Номер кода	K	t	T_{block}	T_{ISD}	T_1'	T_1
10	43	3237	867	68	279599	1053
10	44	3237	864	67	251578	993
11	64	1706	423	131	44439	1256
11	65	1706	422	131	20203	786
11	66	1706	415	132	13747	653
12	99	1706	421	121	450178	1141
12	100	1706	429	120	223749	749
12	101	1706	428	123	175088	650
13	42	7687	3528	161	40719	4287
13	43	7687	3581	158	21435	3919
13	44	7687	3535	158	17439	3774
13	42	7772	3528	161	47803	4288
13	43	7772	3581	158	24911	3919
13	44	7772	3535	158	20125	3775
14	26	673	45288	314	1246966	45974
14	27	673	45792	260	1368672	46133
14	28	673	45173	263	4782066	45476
15	64	1706	425	207	72047	1782
15	65	1706	412	206	31724	990
15	66	1706	418	204	21226	789
16	42	1292	85186	637	684603	88288
16	43	1292	85381	637	408977	86819
16	44	1292	85451	635	343896	86428
17	120	1706	431	385	126849846	1785
17	121	1706	429	386	132266803	1162

Продолжение таблицы 4

Номер кода	K	t	T_{block}	T_{ISD}	T_1'	T_1
17	122	1706	429	381	239088216	1026
18	57	7279	3416	226	55170	3957
18	58	7279	3430	224	36806	3731
18	59	7279	3440	228	36728	3689
19	42	1083	81554	1151	375059	87152
19	43	1083	82954	1153	227079	85487
19	44	1083	82074	1150	191324	83831

В таблице 5 приводится оценка корректирующей способности декодера MatrixDecoder и его времени декодирования для кодов из таблицы 2. Отметим, что в качестве обоих алгоритмов DecRow и DecCol использовался декодер Йе–Аббе из [57]. В таблице содержится оценка DFR_2 в соответствии с (2.33), а также оценка времени выполнения алгоритма T_2 в миллисекундах в соответствии с (2.34).

Таблица 5 — Оценка корректирующей способности декодера MatrixDecoder и его времени декодирования

Номер кода	t	DFR_2	T_2
1	4000	$1.54E$–12	2375
1	3749	$1.173E$–17	2375
2	3392	0.000273	3172
3	3392	0.000273	3235
4	1744	0.0296	906
5	1194	0.119	9892
6	1706	$5.963E$–20	7828

Продолжение таблицы 5

Номер кода	t	DFR_2	T_2
7	3830	$5.338E{-}8$	2907
8	612	$5.522E{-}6$	43597
9	612	$5.522E{-}6$	46722
10	3237	$4.557E{-}6$	3156
11	1706	$5.963E{-}20$	7985
12	1706	$2.167E{-}9$	127822
13	7687	$1.344E{-}14$	8673
13	7772	$1.128E{-}13$	8673
14	673	0.000104	45800
15	1706	$5.963E{-}20$	8001
16	1292	$3.802E{-}9$	81084
17	1706	0.000115	1233434
18	7279	$6.379E{-}10$	41893
19	1083	$2.955E{-}14$	80396

Отметим, что алгоритмы декодирования ISDDecoder и MatrixDecoder имеют свои особенности и могут применяться на практике в зависимости от задачи. Так, декодер MatrixDecoder для некоторых кодов обладает меньшим значением DFR по сравнению с ISDDecoder при сопоставимом времени декодирования. Однако этот DFR для заданного кода и количества добавляемых ошибок является фиксированным. В то время как при использовании ISDDecoder можно добиться сколь угодно малых значений DFR за счет увеличения максимального времени декодирования. Например, для кода 1 из таблицы 2 и $t = 3749$ декодеру ISDDecoder в худшем случае потребуется 26 итераций ДИС и около 2600 миллисекунд для обеспечения вероятности ошибочного декодирования 10^{-9}. В то время как для тех же параметров декодер MatrixDecoder при меньшем времени

декодирования (2300 миллисекунд) обеспечивает DFR около 10^{-17}, что значительно меньше. Однако, например, для кодов 4 и 5 декодер MatrixDecoder не может обеспечить даже DFR $= 10^{-9}$, в отличие от ISDDecoder.

2.4 Выводы

В главе были определены исследуемые в работе D-коды и получены их криптографические свойства. В частности, найдены условия, при которых некоторая степень Шура–Адамара D-кода на основе кодов Рида–Маллера разложима в прямую сумму неразложимых кодов. Полученные условия позволяют упростить структурный анализ криптосистемы на D-кодах.

Для применения D-кодов в криптосистеме типа Мак–Элиса необходимо наличие эффективного декодера. Поэтому в главе решается задача декодирования D-кодов. В качестве гарантированного декодера строится мажоритарный декодер на основе подхода Мэсси. Именно, сначала строится алгоритмическая модель декодирования тензорного произведения кодов с использованием конструкции декодирующих деревьев. Затем эта модель обобщается для D-кодов с применением оптимизированной конструкции декодирующего графа. В качестве вероятностных строятся два декодера – ISDDecoder и MatrixDecoder, использующие блочную структуру кодового слова D-кода. В конце главы приведены результаты проведенных экспериментов по оценке эффективности построенных вероятностных декодеров, показывающие их более высокую корректирующую способность относительно гарантированных декодеров.

Глава 3. Синтез и анализ схемы KEM на основе системы типа Мак–Элиса на D-кодах

3.1 Криптосистема типа Мак–Элиса на D-кодах

Разработанные в предыдущей главе алгоритмы декодирования D-кодов позволяют применять их в криптосистеме типа Мак–Элиса, а ее в свою очередь можно применить в схеме KEM. В этом случае публичная матрица системы $\mathrm{McE}(\overline{C(D)})$ представима в виде:

$$\tilde{G} = S(G_{\overline{C(D)}})P, \tag{3.1}$$

где S – невырожденная $(\dim(\overline{C(D)}) \times \dim(\overline{C(D)}))$–матрица, P – перестановочная $(n_1 n_2 \times n_1 n_2)$-матрица, а $G_{\overline{C(D)}}$ – порождающая матрица выбранного D-кода.

Для удобства мажоритарный декодер с гарантированным исправлением ошибок, построенный для D-кодов в главе 2.3.2, обозначим как $\mathsf{GraphDecoder}$. Схему типа Мак–Элиса на основе D-кода $\overline{C(D)}$ вида (2.27) с характеристиками $[n,k,d]_2$ и декодером $\mathsf{DDecoder} \in \{\mathsf{GraphDecoder}, \mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}$, также будем обозначать $\mathrm{McE}(D,\mathsf{DDecoder})$. Тройка соответствующих алгоритмов системы $\mathrm{McE}(D,\mathsf{DDecoder})$ приведена на рис. 3.1. Формально схема 3.1 отличается от 1.3 только конкретизацией поля $\mathbb{F}_2$, над которым рассматриваются D-коды, а также указанием на использование декодера $\mathsf{DDecoder}$ для D-кодов в правиле расшифрования. Вес t векторов ошибок, добавляемых в правиле шифрования Enc, определяется из условия

$$t = \begin{cases} \lfloor (d-1)/2 \rfloor, \mathsf{DDecoder} = \mathsf{GraphDecoder}, \\ t' > \lfloor (d-1)/2 \rfloor, \mathsf{DDecoder} \in \{\mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}, \end{cases} \tag{3.2}$$

где t' выбирается исходя из кода, декодера, вероятности ошибочного расшифрования и требуемой стойкости λ_{OW} соответствующей криптосистемы.

$\mathsf{Gen}^{\mathrm{McE}(D)}(t)$:

---

1. $S \leftarrow \mathrm{GL}_k(\mathbb{F}_q)$
2. $P \leftarrow \mathcal{P}_n$
3. $pk = (\tilde{G} = SGP, t)$
4. $sk = (S, P)$
5. ***вернуть*** (pk, sk)

$\mathsf{Enc}(pk, \mathbf{m}; \mathbf{e})$:

---

1. $\mathbf{c} = \mathbf{m}\tilde{G} + \mathbf{e}$, $\mathrm{wt}(\mathbf{e}) = t$
2. ***вернуть*** $\mathbf{c}$

$\mathsf{Dec}^{\mathsf{DDecoder}}(sk, \mathbf{c})$:

---

1. $\mathbf{x} \leftarrow \mathsf{DDecoder}(\mathbf{c}P^{-1})$
2. *если* $\mathbf{x} \neq \perp$
3. $\quad \mathbf{m}' = \mathbf{x}G^{-1}S^{-1}$
4. $\quad$ ***вернуть*** $\mathbf{m}'$
5. *иначе*
6. $\quad$ ***вернуть*** $\perp$

Рисунок 3.1 — Схема $\mathrm{McE}(D, \mathsf{DDecoder})$, $\mathsf{DDecoder} \in \{\mathsf{GraphDecoder}, \mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}$.

Более подробно выбор t' для некоторых систем $\mathrm{McE}(D, \mathsf{DDecoder})$ обсуждается в следующем разделе. В правиле расшифрования Dec на рис. 3.1 алгоритм $\mathsf{DDecoder}$ вызывается с одним параметром – значением декодируемого вектора, при этом конкретные реализации декодера $\mathsf{DDecoder}$ могут принимать и дополнительные параметры. В частности, алгоритмы декодирования $\mathsf{ISDDecoder}$ и $\mathsf{MatrixDecoder}$ в качестве дополнительных принимают такие параметры, как информационная совокупность τ_0, порождающая матрица G, условие $\mathsf{ExitCond}$, алгоритмы op_1 и op_2 преобразования информационного и декодируемого векторов перед проверкой условия декодирования и т.п. Можно считать, что при реализации схемы шифрования значения таких параметров фиксированы, поэтому они могут рассматриваться как часть реализации деко-

дера, а не как его параметры. В [58] (см. https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/McED%20Enc%20Dec%20Prob) можно ознакомиться с реализацией на C++ алгоритмов шифрования и расшифрования с использованием построенных вероятностных декодеров. Вместе с программой содержится текстовый документ с примерами работы для конкретного D-кода (см. https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/McED%20Enc%20Dec%20Prob/mced/x64/Release).

3.2 Атака на основе произведения Шура–Адамара

Так как D-код в общем случае принадлежит классу, отличному от классов базовых кодов, то структурные атаки на криптосистемы Мак–Элиса на основе базовых кодов непосредственно не применимы к криптосистеме Мак–Элиса на D-кодах. В частности, если базовыми кодами являются обобщенные коды Рида–Соломона, то не применима атака Сидельникова–Шестакова. Когда базовые коды – двоичные коды Рида–Маллера, то в общем случае не применимы атаки Миндера–Шокроллахи и Чижова–Бородина. Поэтому актуальна задача анализа структурной стойкости криптосистемы $\mathrm{McE}(\overline{C(D)})$.

В данном разделе проводится анализ структурной стойкости системы типа Мак–Элиса на D-кодах. В частности показано, что при соблюдении определенных условий удается построить атаку, получающую информацию о секретном ключе, то есть во введенной в разделе 1.5.2 классификации строится структурная атака с частичным восстановлением ключа.

Рассмотрим код $\overline{C(D)}$ вида (2.12). Пусть $K = \dim(\overline{C(D)})$, $k = \dim(\overline{C_2(\ell_1)})$. Так как код $\overline{C(D)}$ имеет вид (2.12), то для $\boldsymbol{\tau}_i = \{(i-1)n_2+1,\ldots,in_2\}$ проекция кода $\overline{C(D)}$ на множество $\boldsymbol{\tau}_i$, получаемая путем отбрасывания в кодовых словах всех координат, за исключением координат из множества $\boldsymbol{\tau}_i$, совпадает с $\overline{C_2(\ell_1)}$, $i \in [\![1,n_1]\!]$. Поэтому найдутся такие $(K \times k)$–матрицы M_1,

..., M_{n_1} ранга k, что

$$G_{\overline{C(D)}} = (M_1|...|M_{n_1})\mathrm{diag}(G_{\overline{C_2(\ell_1)}},...,G_{\overline{C_2(\ell_1)}}). \tag{3.3}$$

Для кода $\overline{C_2(\ell_1)}$ и криптосистемы McE($\overline{C_2(\ell_1)}$) обозначим через Attack алгоритм, принимающий на вход публичный ключ $\tilde{G}'$ криптосистемы Мак–Элиса на коде Рида–Маллера $\overline{C_2(\ell_1)}$ с порождающей матрицей $G_{\overline{C_2(\ell_1)}}$ и возвращающий невырожденную $(k \times k)$–матрицу S' и перестановочную $(n_2 \times n_2)$–матрицу P', для которых $\tilde{G}' = S'G_{\overline{C_2(\ell_1)}}P'$.

Теорема 5. *Пусть Attack – алгоритм полиномиальной сложности. Если* $\overline{C(D)}^v = \mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v$ *для некоторого* $v(\in \mathbb{N})$, *код* $\overline{C_2(\ell_1)}^v$ *неразложимый, то существует алгоритм полиномиальной сложности, который по публичной матрице* $\tilde{G}$ *вида* (3.1) *находит такую перестановку* π, *что* $\pi(\mathcal{L}(\tilde{G})) \subseteq \mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}$.

Доказательство. Публичная матрица $\tilde{G}$ вида (3.1), с учетом представления (3.3) имеет вид

$$\tilde{G} = S(G_{\overline{C(D)}})P = (\mathbf{M}_1 G_{\overline{C_2(\ell_1)}}|...|\mathbf{M}_{n_1} G_{\overline{C_2(\ell_1)}})P, \tag{3.4}$$

где $\mathbf{M}_i = SM_i$, $i \in [\![1,n_1]\!]$. Обозначим через $\mathcal{P}_{n_1 n_2}$ симметрическую группу из $n_1 n_2$ элементов. Для удобства перестановку из этой группы, соответствующую матрице P в (3.4), обозначим φ. Вместо P иногда будем писать P_φ.

Из условия вытекает, что код с матрицей $\tilde{G}^v$ перестановочно эквивалентен тензорному произведению $\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v$. Так как, по условию, код $\overline{C_2(\ell_1)}^v$ неразложимый, то, применяя полиномиальный алгоритм Decomposition из [78] для разложения кода $\mathcal{L}(\tilde{G}^v)$ в прямую сумму подкодов, по матрице $\tilde{G}^v$ можно получить такую перестановку $\sigma(\in \mathcal{P}_{n_1 n_2})$, что

$$\tilde{G}^v P_\sigma \sim \mathrm{diag}(\dot{\mathbf{G}}_1,...,\dot{\mathbf{G}}_{n_1}), \tag{3.5}$$

где матрицы $\dot{\mathbf{G}}_i$ полного ранга порождают коды $C_{2,i} = \mathcal{L}(\dot{\mathbf{G}}_i) \sim \overline{C_2(\ell_1)}^v$. Тогда

$$\begin{aligned}\mathcal{L}(\tilde{G}^v P_\sigma) &= \sigma(\varphi(\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v)) \\ &= \sigma(\varphi(\underbrace{\overline{C_2(\ell_1)}^v \oplus \cdots \oplus \overline{C_2(\ell_1)}^v}_{n_1})) = C_{2,1} \oplus \cdots \oplus C_{2,n_1}. \end{aligned} \tag{3.6}$$

Из неразложимости кода $\overline{C_2(\ell_1)}^v$ вытекает, что группа перестановочных автоморфизмов кода $\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v$ в этом случае состоит из перестановок вида:

$$\omega = \begin{pmatrix} 1,...,n_2,n_2+1,...,2n_2,...,(n_1-1)n_2+1,...,n_1n_2 \\ x_1,...,x_{n_2},x_{n_2+1},...,x_{2n_2},...,x_{(n_1-1)n_2+1},...,x_{n_1n_2} \end{pmatrix}, \tag{3.7}$$

с двумя ограничениями: 1) для каждого $i \in [\![0,n_1-1]\!]$ найдется единственное $j \in [\![0,n_1-1]\!]$, что $\{x_{in_2+1},...,x_{(i+1)n_2}\} = \{jn_2+1,...,(j+1)n_2\}$ как множества, 2) каждый набор $(x_{in_2+1},...,x_{(i+1)n_2})$ упорядочен так, что проекция кода $\omega(\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v)$, получаемая путем отбрасывания в кодовых словах всех координат, за исключением координат из множества $\{x_{in_2+1},...,x_{(i+1)n_2}\}$, совпадает с $\overline{C_2(\ell_1)}^v$ (см. [42], Лемма 11). Отсюда вытекает, что перестановка $\xi = \sigma \circ \varphi$ в (3.6) имеет вид (3.7). Так как алгоритм Decomposition по разложимому коду находит перестановку, переводящую этот код в прямую сумму неразложимых кодов с точностью до перестановки координат в кодах–слагаемых, то для ξ выполняется только ограничение 1).

Учитывая (3.4), получаем

$$\begin{aligned}\tilde{G}P_\sigma &= S(G_{\overline{C_2(\ell_1)}})P_\varphi P_\sigma = S(G_{\overline{C_2(\ell_1)}})P_\xi \\ &= (\mathbf{M}_1 G_{\overline{C_2(\ell_1)}}|...|\mathbf{M}_{n_1} G_{\overline{C_2(\ell_1)}})P_\xi \\ &= (\mathbf{M}_{\theta^{-1}(1)} G_{\overline{C_2(\ell_1)}} P_{\gamma_1}|...|\mathbf{M}_{\theta^{-1}(n_1)} G_{\overline{C_2(\ell_1)}} P_{\gamma_{n_1}}) \\ &= [\tilde{\mathbf{G}}_1|...|\tilde{\mathbf{G}}_{n_1}], \gamma_j \in \mathcal{P}_{n_2}, j = 1,...,n_1, \end{aligned} \tag{3.8}$$

где θ – некоторая перестановка из $\mathcal{P}_{n_1}$, соответствующая ξ, а $\tilde{\mathbf{G}}_i = \mathbf{M}_{\theta^{-1}(i)} G_{\overline{C_2(\ell_1)}} P_{\gamma_i}$. При этом запись $\theta^{-1}(i)$ обозначает порядковый номер j матрицы M_j, которая после действия P_ξ находится в блочном представлении (3.8) на месте i–го блока.

Очевидно, что $\mathrm{rank}(\tilde{\mathbf{G}}_i) = k$ для всех $i \in [\![1,n_1]\!]$. По $\tilde{\mathbf{G}}_i$ легко построить соответствующую невырожденную $(k \times n_2)$–матрицу $\hat{\mathbf{G}}_i$, состоящую из k линейно-независимых строк матрицы $\tilde{\mathbf{G}}_i$. Обозначим через F_i такую невырожденную $(K \times K)$–матрицу, что

$$F_i \tilde{\mathbf{G}}_i = \left[\frac{\hat{\mathbf{G}}_i}{O}\right],$$

где O – нулевая $((K-k)\times n_2)$–матрица. Матрица $\hat{\mathbf{G}}_i$ может рассматриваться как публичная матрица криптосистемы $\mathrm{McE}(\overline{C_2(\ell_1)})$. Применяя к каждой матрице $\hat{\mathbf{G}}_i$ алгоритм Attack, можно найти перестановочную матрицу P_{δ_i} $(\delta_i \in \mathcal{P}_{n_2})$ и невырожденную матрицу S'_i, что $\hat{\mathbf{G}}_i = S'_i G_{\overline{C_2(\ell_1)}} P_{\delta_i}$. Откуда получаем

$$\begin{aligned}
\tilde{G} P_\sigma \mathrm{diag}(P_{\delta_1}^{-1},...,P_{\delta_{n_1}}^{-1}) &= [\tilde{\mathbf{G}}_1 P_{\delta_1}^{-1} | ... | \tilde{\mathbf{G}}_{n_1} P_{\delta_{n_1}}^{-1}] \\
&= \left[F_1^{-1}\left[\frac{\hat{\mathbf{G}}_1}{O}\right] P_{\delta_1}^{-1} \middle| ... \middle| F_{n_1}^{-1}\left[\frac{\hat{\mathbf{G}}_{n_1}}{O}\right] P_{\delta_{n_1}}^{-1}\right] \\
&= \left[F_1^{-1}\left[\frac{S'_1}{O'}\right] G_{\overline{C_2(\ell_1)}} \middle| ... \middle| F_{n_1}^{-1}\left[\frac{S'_{n_1}}{O'}\right] G_{\overline{C_2(\ell_1)}}\right] \\
&= [\hat{\mathbf{S}}_1 G_{\overline{C_2(\ell_1)}} | ... | \hat{\mathbf{S}}_{n_1} G_{\overline{C_2(\ell_1)}}] = [\hat{\mathbf{S}}_1 | ... | \hat{\mathbf{S}}_{n_1}](I_{n_1} \otimes G_{\overline{C_2(\ell_1)}}),
\end{aligned}$$

где O' – нулевая $((K-k)\times k)$–матрица, $\hat{\mathbf{S}}_i = F_i^{-1}\left[\frac{S'_i}{O'}\right]$. Следовательно, искомая перестановка имеет вид $\pi = \delta_1^{-1} \circ ... \circ \delta_{n_1}^{-1} \circ \sigma$. Так как $\mathsf{Decomposition}$, Attack – алгоритмы полиномиальной сложности, то описанный алгоритм нахождения π также имеет полиномиальную сложность. □

Отметим, что для произвольных линейных кодов C_1, C_2 над полем $\mathbb{F}_q$ существует такая перестановка σ, зависящая только от длин кодов C_1, C_2, что $C_1 \otimes C_2 = \sigma(C_2 \otimes C_1)$. Поэтому теорема 5 остается справедливой, если вместо $\overline{C(D)}^v = \mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v$ рассмотреть произведение $\overline{C(D)}^u = \overline{C_1(k_s)}^u \otimes \mathbb{F}_q^{n_2}$.

Теорема 5 показывает, что в ряде случаев по публичному ключу криптосистемы $\mathrm{McE}(\overline{C(D)})$ за полиномиальное время может быть найдена перестановка, позволяющая преобразовать этот ключ к более простому виду. Алгоритм нахождения этой перестановки фактически описан в доказательстве теоремы 5, а упрощенная схема представлена на рисунке 3.2.

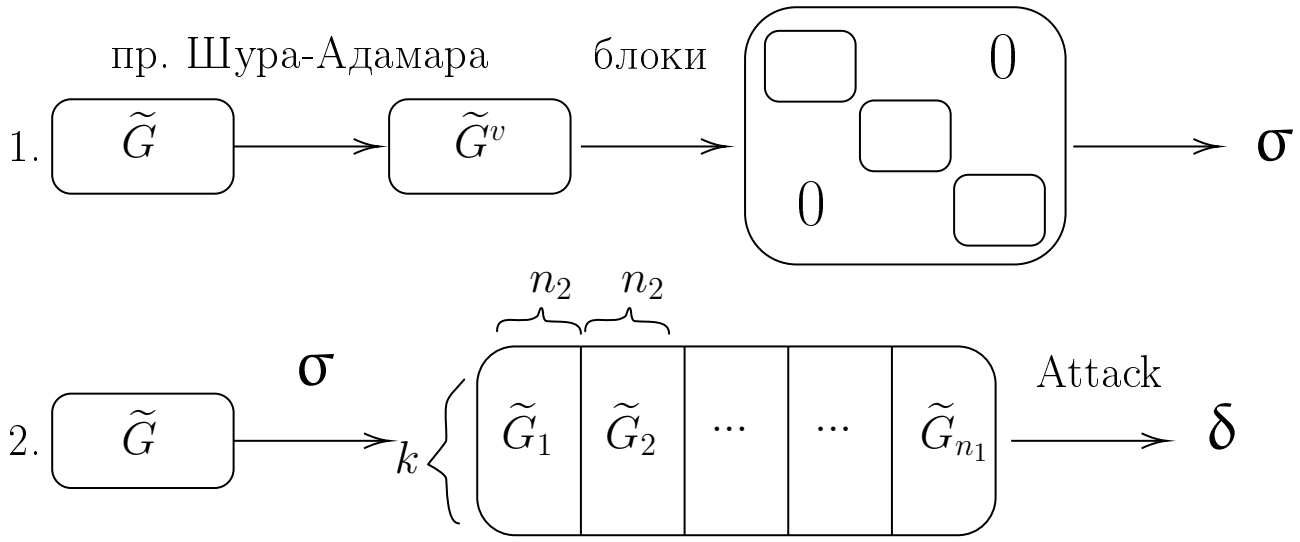

Рисунок 3.2 — Схема алгоритма структурной атаки

Для D-кода $\overline{C(D)}$, удовлетворяющего условиям теоремы 1, выполняются условия теоремы 5. Это позволяет сделать вывод, что коды с такими параметрами не подходят для использования в криптосистеме Мак–Элиса. С другой стороны, D-коды с параметрами, удовлетворяющими условиям одной из теорем 2, 3, 4, обеспечивают устойчивость к нахождению перестановки π. Это вытекает из того факта, что их квадрат совпадает со всем пространством или является неразложимым кодом. При этом, в случае неразложимости квадрата D-кода, его третья степень совпадает со всем пространством.

3.3 Анализ стойкости криптосистем Мак–Элиса, основанных на подкоде прямой суммы кодов

Построенная в предыдущем разделе атака упрощает вид открытого ключа, что в свою очередь позволяет, например, использовать для дешифрования сообщений декодер кода $\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}$, заключающийся в применении декодера кода Рида–Маллера $\overline{C_2(\ell_1)}$ к каждому из n_1 блоков длины n_2. Ясно, что

с большой вероятностью такой декодер будет ошибаться, так как кодовое расстояние кода $\mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}$ меньше кодового расстояния D-кода, используемого в криптосистеме. Представляется, что уменьшить вероятность неправильного декодирования можно, например, применяя метод декодирования по информационным совокупностям и учитывая структуру кода. Следуя этой идее, на основе результатов, полученных в разделе 2.1.1, в следующих разделах строится криптоаналитическая модель для системы типа Мак–Элиса на основе D-кодов. Именно, строится комбинированная атака, позволяющая значительно повысить вероятность атаки на шифрограмму по сравнению с простой атакой декодированием по информационным совокупностям. Для упрощения анализа предполагается, что количество добавляемых при шифровании (см. рис. 3.1) ошибок t равно половине кодового расстояния используемого D-кода, то есть в системе используется гарантированный декодер. Также для упрощения анализа вектор ошибок генерируется следующим образом. Каждый бит вектора $\mathbf{e}$ принимает значение 1 с вероятностью t/n и 0 с вероятностью $1-t/n$. В рамках такой модели генерации ошибок можно проверять вес вектора $\mathbf{e}$ при шифровании и в случае необходимости повторять процесс генерации. Тогда можно считать, что $\mathrm{wt}(\mathbf{e}) = t$, то есть вес вектора ошибок находится в пределах исправляющей способности кода. Отметим, что такой способ генерации ошибки применен в [79] также для упрощения анализа.

Пусть $n_1, n_2 \in \mathbb{N}$, $n = n_1 n_2$, C_2 – $[n_2,k_2,d_2]$–код. Рассмотрим $[n,k,d]$–код C, для которого выполняются следующие условия:

$$C \subset \mathbb{F}_q^{n_1} \otimes C_2, \mathrm{rank}(\boldsymbol{\tau}_i(G_C)) = k_2,$$
$$\boldsymbol{\tau}_i = \{(i-1)n_2 + 1, ..., in_2\}, i \in [\![1,n_1]\!]. \qquad (3.9)$$

Подматрицу $\boldsymbol{\tau}_i(G_C)$ назовем блоком матрицы G_C с номером i. Порождающая матрица G_C может быть представлена в виде:

$$G_C = M\mathrm{diag}(G_{C_2},...,G_{C_2}) = (M_1 G_{C_2} \parallel ... \parallel M_{n_1} G_{C_2})$$

для $(k \times n_1 \dim(C_2))$–матрицы $M = (M_1 \parallel ... \parallel M_{n_1})$, $\mathrm{rank}(M_i) = k_2$.

Рассмотрим криптосистему Мак–Элиса McE(C) на коде C. Пусть для натурального $s \geqslant 2$

$$C^s = \mathbb{F}_q^{n_1} \otimes C_2^s \neq \mathbb{F}_q^{n_1 n_2} \tag{3.10}$$

и C_2^s – неразложимый, тогда к McE(C) можно применить алгоритм атаки из предыдущего раздела, который находит такую перестановочную матрицу Π, что

$$\tilde{G}\Pi = (\hat{S}_1 G_{C_2} \parallel ... \parallel \hat{S}_{n_1} G_{C_2}) = (\hat{G}_1 \parallel ... \parallel \hat{G}_{n_1}) = \hat{G}, \tag{3.11}$$

где $\hat{S}_i$ – $(k \times k_2)$–матрица ранга k_2. Обозначим этот алгоритм AttackDKey.

Замечание 1. *В случае, когда C^{s-1} является неразложимым кодом, а $C^s = \mathbb{F}_q^{n_1 n_2}$, то алгоритм AttackDKey не применим к криптосистеме* McE(C).

Пусть $\mathbf{z} = \mathbf{m}\tilde{G} + \mathbf{e}$ – полученная шифрограмма, где $\mathbf{m}$ – информационное сообщение, а $\mathbf{e}$ – вектор ошибок. Тогда

$$\begin{aligned} \mathbf{z}\Pi &= \mathbf{m}\hat{G} + \mathbf{e}\Pi \\ &= [c_1 \parallel c_2 \parallel ... \parallel c_{n_1}] + [\hat{e}_1 \parallel \hat{e}_2 \parallel ... \parallel \hat{e}_{n_1}] \\ &= [\hat{z}_1 \parallel \hat{z}_2 \parallel ... \parallel \hat{z}_{n_1}] = \hat{\mathbf{z}}, c_i \in C_2. \end{aligned}$$

Обозначим $\hat{\mathbf{c}} = [\hat{c}_1 \parallel ... \parallel \hat{c}_{n_1}]$ вектор, полученный по вектору $\hat{\mathbf{z}}$ путем применения гарантированного декодера для кода C_2 к каждому блоку $\hat{z}_i$. Для удобства декодер, принимающий на вход $\hat{\mathbf{z}}$ и возвращающий $\hat{\mathbf{c}}$ обозначим DecoderSum. Тогда к шифрограмме $\mathbf{z}$ может быть применена атака AttackDCipher (см. алгоритм 20), которая заключается в применении алгоритма AttackDKey для нахождения матрицы Π (шаг 1), декодировании вектора $\mathbf{z}\Pi$ с помощью алгоритма DecoderSum (шаг 3) и дальнейшем применении к полученному вектору алгоритма декодирования по информационным совокупностям (шаги 6-14). В случае успеха атаки алгоритм AttackDCipher вернет значение $\hat{\mathbf{m}} = \mathbf{m}$, в противном случае будет возвращено значение $\hat{\mathbf{m}} = \perp$.

Алгоритм 20 AttackDCipher

Исходные параметры: $\tilde{G}$, $\mathbf{z}$, p, I_{max}
Результат: $\hat{\mathbf{m}}$

1: $\Pi := \mathsf{AttackDKey}(\tilde{G})$.
2: $\hat{G} := \tilde{G}\Pi$, $\hat{\mathbf{z}} := \mathbf{z}\Pi$.
3: $\hat{\mathbf{c}} := \mathsf{DecoderSum}(\hat{\mathbf{z}})$.
4: Выбрать из $\{1,...,n_1\}$ минимальное K_p, при котором матрица, составленная из K_p случайно выбранных подматриц $\hat{G}_i$ матрицы $\hat{G}$, будет иметь ранг k с вероятностью больше p.
5: $i := 0$
6: **повторять**
7: Случайно выбрать K_p блоков $\hat{c}_i$, множество номеров координат в выбранных блоках обозначим T.
8: **если** $\mathrm{rank}(T(\hat{G})) = k$, **то**
9: выбрать $\boldsymbol{\tau} \subset T$ такое, что $|\boldsymbol{\tau}| = k$, $\mathrm{rank}(\boldsymbol{\tau}(\hat{G})) = k$,
10: **иначе**
11: Перейти на шаг 7.
12: $\hat{\mathbf{m}} := \boldsymbol{\tau}(\hat{\mathbf{c}})\boldsymbol{\tau}(\hat{G})^{-1}$.
13: $i := i + 1$
14: **пока** $\mathrm{wt}(\hat{\mathbf{z}} - \hat{\mathbf{m}}\hat{G}) \neq t$ и $i < I_{max}$.
15: **если** $i = I_{max}$ и $\mathrm{wt}(\hat{\mathbf{z}} - \hat{\mathbf{m}}\hat{G}) \neq t$, **то**
16: $\hat{\mathbf{m}} := \perp$.
17: **вернуть** $\hat{\mathbf{m}}$.

Замечание 2. *Параметр p алгоритма AttackDCipher влияет на выбор K_p. Чем больше p, тем больше K_p, т.е. чем с большей вероятностью нужно находить на шаге 8 матрицу заданного ранга, тем большее количество блоков нужно выбирать. С другой стороны, чем больше значение K_p, тем меньше вероятность того, что среди выбранных блоков не будет плохих. Так как на шаге 4 алгоритма для заданного p нужно провести минимум $1/p$ экспериментов для проверки очередного выбранного значения K_p, то значение параметра p имеет смысл выбирать исходя из вычислительных возможностей атакующего.*

Замечание 3. *Найденное на шаге 4 значение K_p может оказаться слишком большим, чтобы нашлось K_p безошибочных блоков $\hat{c}_i$. Тогда проверка $\mathrm{wt}(\hat{\mathbf{z}}-\hat{\mathbf{m}}\hat{G}) \neq t$ будет всегда выполняться. Чтобы не произошло зацикливания алгоритма введен параметр I_{max}. Значение I_{max} нужно выбирать разумным*

образом, то есть так, чтобы оно не влияло на вероятность успеха атаки. Далее при оценке вероятности успеха атаки рекомендации по выбору значения параметра I_{max} уточняются.

Проведем анализ вероятности успеха атаки AttackDCipher, который основан на оценке количества N_g безошибочных блоков $\hat{c}_i$ в векторе, полученном после применения декодера DecoderSum. Если $wt(\hat{e}_i) \leqslant t_2 = \lfloor (d_2-1)/2 \rfloor$, то вектор $\hat{z}_i$ будет декодирован правильно, в таком случае блок $\hat{z}_i$ назовем *хорошим*, в противном случае, то есть при $wt(\hat{e}_i) > t_2$, блок $\hat{z}_i$ назовем *плохим*. Рассмотрим варианты распределения $t = \lfloor (d-1)/2 \rfloor$ ошибок по n_1 блокам. Наибольшее количество «плохих» блоков достигается в случае, когда в каждом таком блоке будет ровно $t_2 + 1$ ошибка. При другом варианте распределения ошибочных координат количество «плохих» блоков может быть только меньше. Тогда максимальное количество плохих блоков равно $\lfloor \lfloor (d-1)/2 \rfloor / \lfloor (d_2-1)/2+1 \rfloor \rfloor$, а минимальное количество хороших соответственно

$$N_g^{min} = n_1 - \lfloor (d-1)/(d_2+1) \rfloor. \tag{3.12}$$

Оценим среднее количество хороших блоков. В рамках рассматриваемой модели генерации ошибки, каждая из t ошибок попадает в i–й блок с вероятностью $1/n_1$, $i \in [\![1,n_1]\!]$, то есть вероятность появления ошибки в i–м блоке подчиняется распределению Бернулли с параметром t/n_1. Тогда среднее число ошибок в блоке равно t/n_1 и, если $t/n_1 <= t_2 + 1$, то, при весе плохого блока более t_2+1, вероятность появления такого блока уменьшается. Поэтому среднее количество плохих блоков будет также уменьшаться. Если

$$t/n_1 > t_2 + 1, \tag{3.13}$$

то в среднем все блоки плохие и описываемая далее атака не применима. Тогда далее при анализе будем считать, что вес плохого блока равен $t_2 + 1$. Пусть A_i – свойство, состоящее в том, что i–й блок вектора является плохим, $i \in [\![1,n_1]\!]$. Пусть $C_r(n_1,n_2,t_1,t_2,t)$ – количество векторов ошибок, которые имеют ровно r

плохих блоков. Тогда согласно формуле включения–исключения [80] (см. гл.2, формула (1.9)) выполняется

$$C_r(n_1,n_2,t_1,t_2,t) = \sum_{k=r}^{n_1}(-1)^{k-r}C_k^r S_k,$$

где $S_k = \sum_{1\leqslant i_1<\ldots<i_k\leqslant n_1} M(A_{i_1},\ldots,A_{i_k}) = C_{n_1}^k (C_{n_2}^{t_2+1})^k C_{(n_1-k)n_2}^{t-(t_2+1)k}$, а $M(A_{i_1},\ldots,A_{i_k})$ – количество векторов, обладающих фиксированными свойствами $A_{i_1},\ldots,A_{i_k}$, то есть имеющие плохие блоки в номерах $i_1,i_2,\ldots,i_k$. Тогда вероятность Q_r того, что в векторе ошибок будет ровно r плохих блоков, равна

$$Q_r = \frac{C_r(n_1,n_2,t_1,t_2,t)}{C_n^t}, \tag{3.14}$$

а среднее количество хороших блоков N_g^{avg} вычисляется по формуле

$$N_g^{avg} = \lfloor n_1 - \sum_{r=0}^{n_1} r\cdot Q_r \rfloor = \lfloor n_1 - \sum_{r=1}^{n_1-N_g^{min}} r\cdot Q_r \rfloor. \tag{3.15}$$

Согласно замечанию 2, вероятность $P_{attack}(p)$ успеха атаки зависит от параметра p алгоритма AttackDCipher, так как фактически влияет на значение K_p, которое в свою очередь влияет на вероятность выполнения условий на шагах 8,14. Тогда вероятность $P_{attack}(p)$ оценивается снизу произведением: $P_{attack}(p) \geqslant P_1(p)\cdot P_2(p)$, где $P_1(p) = C_{N_g}^{K_p}/C_{n_1}^{K_p}$ – вероятность выбора K_p хороших блоков из n_1 блоков на шаге 7 алгоритма 20, а $P_2(p)$ – вероятность того, что $K_p n_2$ выбранных столбцов образуют матрицу ранга k. Далее для удобства в обозначениях вероятностей зависимость от p будет опускаться.

Строго говоря, вероятность P_{attack} зависит также от параметра I_{max} алгоритма AttackDCipher. В частности, выбор достаточно малого I_{max} приводит к раннему завершению атаки с ошибкой. Для нивелирования этой зависимости имеет смысл выбирать $I_{max} \approx 1/(P_1\cdot P_2)$.

Так как минимальное количество хороших блоков равно N_g^{min}, а среднее – N_g^{avg}, то $P_1^{min} = C_{N_g^{min}}^{K_p}/C_{n_1}^{K_p}$, $P_1^{avg} = C_{N_g^{avg}}^{K_p}/C_{n_1}^{K_p}$. Тогда

$$P_{attack}^{min} \geqslant P_1^{min}\cdot P_2,\ P_{attack}^{avg} \geqslant P_1^{avg}\cdot P_2. \tag{3.16}$$

3.4 Анализ стойкости системы $\mathrm{McE}(\overline{C(D)})$

В данном разделе анализируется стойкость криптосистемы $\mathrm{McE}(C)$ на основе D-кода C с применением результатов предыдущего раздела. Сначала рассматривается частный случай D-кода – тензорное произведение, а затем рассматривается общий случай, когда D-код имеет вид (2.12).

3.4.1 Случай $C = C_1 \otimes C_2$

Рассмотрим криптосистему $\mathrm{McE}(C)$ на $[n, k, d]$–коде $C = C_1 \otimes C_2$, где C_i – $[n_i, k_i, d_i]$–код, $n = n_1 n_2$, $k = k_1 k_2$, $d = d_1 d_2$, $t = \lfloor (d_1 d_2 - 1)/2 \rfloor$, $t_i = \lfloor (d_i - 1)/2 \rfloor$ для $i \in \{1,2\}$. Для кода C неравенство (3.13) не выполняется, так как

$$t \leqslant (d_1 d_2 - 1)/2 \leqslant (n_1 d_1 - 1)/2$$

и

$$t_2 + 1 = \lfloor (d_2 - 1)/2 \rfloor + 1 = \lfloor (d_2 + 1)/2 \rfloor,$$

откуда

$$t/n_1 \leqslant (d_2 - 1/n_1)/2 \leqslant d_2/2 \leqslant \lfloor (d_2 + 1)/2 \rfloor = t_2 + 1.$$

Также отметим, что для кода C выполняются условия из (3.9). Поэтому, если $C^s = \mathbb{F}_q^{n_1} \otimes C_2^s$ и C_2^s – неразложимый код, то к криптосистеме $\mathrm{McE}(C)$ с публичной матрицей вида (1.22) может быть применена атака из раздела 3.3. В частности, может быть найдена перестановочная матрица Π на шаге 1 алгоритма 20 и вычислена матрица $\hat{G}$ вида (3.11) на шаге 2 того же алгоритма.

Требуемое на шаге 4 минимальное значение K_p при $p = 0$ для кода $C = C_1 \otimes C_2$ удается найти аналитически. Именно, пусть $\boldsymbol{\omega} \subset \{1,\ldots,n_1\}$ – множество номеров блоков $\hat{G}_i$. Обозначим через T множество номеров $|\boldsymbol{\omega}|n_2$ столбцов

матрицы $\hat{G}$, соответствующих $\boldsymbol{\omega}$. Из [49] известно, что перестановочная матрица Π является такой, что матрица $Q = P\Pi$ переставляет в матрице G_C между собой блоки $\boldsymbol{\tau}_i(G_C)$ и столбцы внутри этих блоков. То есть Q представима в виде

$$Q = (W \otimes I_{n_2})\mathrm{diag}(V_1, ..., V_{n_1}),$$

где W – перестановочная $(n_1 \times n_1)$–матрица, а V_i – перестановочные $(n_2 \times n_2)$–матрицы. Тогда

$$\begin{aligned}\mathrm{rank}(T(\hat{G})) &= \mathrm{rank}(T(S(G_{C_1} \otimes G_{C_2})Q)) = \mathrm{rank}(T((G_{C_1} \otimes G_{C_2})Q)) \\ &= \mathrm{rank}(T((G_{C_1} \otimes G_{C_2})(W \otimes I_{n_2})\mathrm{diag}(V_1, ..., V_{n_1}))) \\ &= \mathrm{rank}(T((G_{C_1}W) \otimes G_{C_2})\mathrm{diag}(V_1, ..., V_{n_1})) \\ &= \mathrm{rank}(\boldsymbol{\omega}(G_{C_1}W))\mathrm{rank}(G_{C_2}) = \mathrm{rank}(\boldsymbol{\omega}(G_{C_1}W))k_2.\end{aligned}$$

Так как в алгоритме 20 на шаге 7 блоки выбираются случайно, то вероятность события $\mathrm{rank}(\boldsymbol{\omega}(G_{C_1}W)) = k_1$ равна вероятности события $\mathrm{rank}(\boldsymbol{\tau}(G_{C_1})) = k_1$ для $|\boldsymbol{\tau}| = |\boldsymbol{\omega}|$. Поэтому вероятность P_2 того, что $\mathrm{rank}(T(\hat{G})) = k_1k_2$, равна вероятности события $\mathrm{rank}(\boldsymbol{\tau}(G_{C_1})) = k_1$ для случайно выбранного $\boldsymbol{\tau}$. При $|\boldsymbol{\tau}| < k_1$ вероятность этого события равна нулю, а уже при $|\boldsymbol{\tau}| = k_1$ для большинства кодов эта вероятность непренебрежима [81]. Поэтому, задав в алгоритме 20 значение параметра p, равное нулю, на шаге 4 будет выбрано $K_p = k_1$.

Рассмотрим случай, когда C_i – $[n_i, k_i, d_i]_2$–код Рида–Маллера $\mathrm{RM}_2(r_i, m_i)$, $i \in \{1,2\}$. Для кода $C = C_1 \otimes C_2$ в предыдущей главе построены эффективные декодеры и предлагается использовать C в криптосистеме Мак–Элиса $\mathrm{McE}(C)$. Стойкость к атакам на ключ и шифрограмму исследована в работах [72], [82]. Покажем, как построенная выше атака позволяет уточнить стойкость $\mathrm{McE}(C)$ к атакам на шифрограмму.

Согласно (1.11), (1.16), для $r_1 \in [\![\lceil m_1/2 \rceil, m_1]\!]$, $r_2 \in [\![0, \lceil m_2/2 \rceil - 1]\!]$ выполняется $C^2 = \mathbb{F}_2^{n_1} \otimes C_2^2$. Также C_2^2 является неразложимым, поэтому к $\mathrm{McE}(C)$ может быть применена атака $\mathsf{AttackDCipher}$. В таблице 6 для некоторых случаев при $m_1, m_2 \in \{7,8\}$ представлена оценка вероятности успеха этой атаки. Для

этого рассчитываются значения P^{min}_{attack} и P^{avg}_{attack} в соответствии с (3.16), при этом P_2 оценивается экспериментально путем вычисления ранга 10000 псевдослучайно выбранных $(k_1 \times K_p)$–подматриц порождающей матрицы кода C_1 и расчета доли матриц ранга k_1. Также в таблице 6 приводится расчет вероятности P_{ISD} успеха атаки на шифрограмму путем обычного декодирования по информационным совокупностям (см. раздел 1.3.1). Для каждого кода $C = C_1 \otimes C_2$ оценивается P^{min}_{attack} и P^{avg}_{attack} для разных значений p с целью анализа поведения этих вероятностей.

В таблице 7 для кодов из таблицы 6 приводится расчет значений N^{min}_g и N^{avg}_g по формулам (3.12) и (3.15) соответственно, расчет вероятности $Q_{N^{avg}_g}$ появления N^{avg}_g хороших блоков, а также вероятность $Q_{N^{min}_g}$ ситуации, когда количество хороших блоков будет минимальным, по формуле (3.14).

Согласно таблице 6, для рассмотренных кодов вероятность дешифрования с помощью построенной атаки AttackDCipher значительно больше вероятности дешифрования путем декодирования по информационным совокупностям, даже в худшем для атакующего случае распределения ошибок. Отметим при этом, что ранее в [82] (см. таблицу 3) коды

$$\mathrm{RM}_2(4,8) \otimes \mathrm{RM}_2(3,7), \mathrm{RM}_2(4,8) \otimes \mathrm{RM}_2(3,8), \mathrm{RM}_2(4,8) \otimes \mathrm{RM}_2(2,8) \qquad (3.17)$$

считались стойкими.

Согласно таблице 6 выбор p, при котором K_p получается больше k_1, позволяет увеличить вероятность успеха атаки за счет значительного увеличения вероятности P_2 и незначительного уменьшения вероятностей P^{min}_1 и P^{avg}_1. В частности, в таблице приведены значения K_p от k_1 до первого такого значения, при котором вероятность P^{min}_{attack} уменьшается. Также видно, что при росте r_1 вероятность успеха атаки увеличивается.

Также отметим, что при вычислении N^{avg}_g по формуле (3.15) для оценки вероятности P^{avg}_{attack} происходит округление в меньшую сторону. Согласно таблице 7, для рассматриваемых в таблице 6 кодов имеем $N^{avg}_g = n_1 - 1$, то есть количество плохих блоков равно 1. Однако даже в этом случае вероятность $Q_{N^{avg}_g}$

Таблица 6 — Вероятность успеха атаки AttackDCipher для тензорного произведения кодов Рида–Маллера

$\mathrm{RM}_2(r_1, m_1) \otimes \mathrm{RM}_2(r_2, m_2)$	p	K_p	P_{ISD}	P_{attack}^{min}	P_{attack}^{avg}
RM_2(4,7) $\otimes$ RM_2(3,7)	0.1	99	2.379E-14	2.883E-06	3.956E-02
RM_2(4,7) $\otimes$ RM_2(3,7)	0.3	100	2.379E-14	4.904E-06	8.564E-02
RM_2(4,7) $\otimes$ RM_2(3,7)	0.5	101	2.379E-14	5.427E-06	1.219E-01
RM_2(4,7) $\otimes$ RM_2(3,7)	0.7	102	2.379E-14	5.017E-06	1.464E-01
RM_2(5,7) $\otimes$ RM_2(3,7)	0.3	120	1.577E-09	5.945E-05	2.265E-02
RM_2(5,7) $\otimes$ RM_2(3,7)	0.6	121	1.577E-09	7.046E-05	3.758E-02
RM_2(5,7) $\otimes$ RM_2(3,7)	0.8	122	1.577E-09	5.109E-05	4.088E-02
RM_2(6,7) $\otimes$ RM_2(3,7)	0.9	127	1.716E-05	7.812E-03	7.812E-03
RM_2(4,8) $\otimes$ RM_2(3,8)	0.2	163	4.868E-30	2.603E-08	8.101E-02
RM_2(4,8) $\otimes$ RM_2(3,8)	0.4	164	4.868E-30	4.690E-08	1.721E-01
RM_2(4,8) $\otimes$ RM_2(3,8)	0.6	165	4.868E-30	5.445E-08	2.362E-01
RM_2(4,8) $\otimes$ RM_2(3,8)	0.7	166	4.868E-30	5.304E-08	2.725E-01
RM_2(5,8) $\otimes$ RM_2(3,8)	0.1	219	1.931E-21	1.488E-07	2.749E-02
RM_2(5,8) $\otimes$ RM_2(3,8)	0.4	220	1.931E-21	2.725E-07	6.043E-02
RM_2(5,8) $\otimes$ RM_2(3,8)	0.6	221	1.931E-21	3.090E-07	8.269E-02
RM_2(5,8) $\otimes$ RM_2(3,8)	0.7	222	1.931E-21	2.993E-07	9.726E-02
RM_2(6,8) $\otimes$ RM_2(3,8)	0.3	247	9.941E-13	1.019E-05	1.179E-02
RM_2(6,8) $\otimes$ RM_2(3,8)	0.6	248	9.941E-13	1.303E-05	2.010E-02
RM_2(6,8) $\otimes$ RM_2(3,8)	0.8	249	9.941E-13	1.063E-05	2.296E-02
RM_2(7,8) $\otimes$ RM_2(3,8)	0.9	255	5.694E-07	3.906E-03	3.906E-03
RM_2(4,8) $\otimes$ RM_2(3,7)	0.2	163	4.412E-22	2.575E-08	8.014E-02
RM_2(4,8) $\otimes$ RM_2(3,7)	0.4	164	4.412E-22	4.611E-08	1.692E-01
RM_2(4,8) $\otimes$ RM_2(3,7)	0.6	165	4.412E-22	5.396E-08	2.341E-01
RM_2(4,8) $\otimes$ RM_2(3,7)	0.7	166	4.412E-22	5.394E-08	2.771E-01
RM_2(4,8) $\otimes$ RM_2(2,8)	0.2	163	2.788E-22	2.551E-08	7.938E-02
RM_2(4,8) $\otimes$ RM_2(2,8)	0.4	164	2.788E-22	4.645E-08	1.705E-01
RM_2(4,8) $\otimes$ RM_2(2,8)	0.6	165	2.788E-22	5.392E-08	2.339E-01
RM_2(4,8) $\otimes$ RM_2(2,8)	0.7	166	2.788E-22	5.320E-08	2.733E-01

достаточно мала. Это означает, что в действительности количество плохих блоков с большой вероятностью равно нулю и вероятность P_{attack}^{avg}, приведенная в таблице 6, значительно больше.

Для криптосистемы McE(C), $C = \mathrm{RM}_2(r_1, m_1) \otimes \mathrm{RM}_2(r_2, m_2)$, согласно (1.11), (1.16) при $r_1 \geqslant \lceil m_1/2 \rceil$ и $r_2 \geqslant \lceil m_2/2 \rceil$ условие (3.10) не выполня-

Таблица 7 — Количество хороших блоков для тензорного произведения кодов Рида–Маллера

$\mathrm{RM}_2(r_1, m_1) \otimes \mathrm{RM}_2(r_2, m_2)$	n_1	N_g^{min}	N_g^{avg}	$Q_{N_g^{min}}$	$Q_{N_g^{avg}}$
RM_2(4,7) $\otimes$ RM_2(3,7)	128	121	127	3.387E-57	3.676E-06
RM_2(5,7) $\otimes$ RM_2(3,7)	128	125	127	1.012E-29	9.481E-09
RM_2(6,7) $\otimes$ RM_2(3,7)	128	127	127	8.701E-12	8.701E-12
RM_2(4,8) $\otimes$ RM_2(3,8)	256	241	255	1.415E-265	1.833E-12
RM_2(5,8) $\otimes$ RM_2(3,8)	256	249	255	8.543E-151	2.538E-17
RM_2(6,8) $\otimes$ RM_2(3,8)	256	253	255	2.381E-75	1.440E-22
RM_2(7,8) $\otimes$ RM_2(3,8)	256	255	255	1.329E-28	1.329E-28
RM_2(4,8) $\otimes$ RM_2(3,7)	256	241	255	9.978E-127	9.616E-06
RM_2(4,8) $\otimes$ RM_2(2,8)	256	241	255	2.294E-547	3.743E-26

ется, а именно $C^2 = \mathbb{F}_2^{n_1 n_2}$. Отметим, что при $\lceil m_1/3 \rceil \leqslant r_1 < \lceil m_1/2 \rceil$ и $\lceil m_2/3 \rceil \leqslant r_2 < \lceil m_2/2 \rceil$ код C^2 является неразложимым как тензорное произведение двух неразложимых кодов, а $C^3 = \mathbb{F}_2^{n_1 n_2}$. Поэтому в указанных случаях, согласно замечанию 1, алгоритм AttackDKey не применим и соответственно не применима атака AttackDCipher. Следовательно, криптосистема на кодах из таблицы 3 работы [82], отличных от (3.17), является стойкой к атаке AttackDCipher.

3.4.2 Общий случай $C = \overline{C(D)}$

Рассмотрим криптосистему McE($\overline{C(D)}$). Так как код $\overline{C(D)}$ имеет вид (2.12), то $\mathrm{rank}(\boldsymbol{\tau}_i(G_{\overline{C(D)}})) = \dim(\overline{C_2(\ell_1)})$ для $\boldsymbol{\tau}_i = \{(i-1)n_2 + 1, ..., in_2\}, i \in [\![1,n_1]\!]$. При этом из $\overline{C_2(\ell_1)} \supset \overline{C_2(\ell_2)} \supset ... \supset \overline{C_2(\ell_s)}$ следует $\overline{C(D)} \subset \mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}$. Поэтому для кода $\overline{C(D)}$ выполняются условия (3.9).

В разделе 2.1.1 найдены условия, при которых выполняется (3.10) для случая, когда $C_1(k_i), C_2(\ell_i)$ – коды Рида–Маллера. То есть $\overline{C(D)}^v = \mathbb{F}_q^{n_1} \otimes \overline{C_2(\ell_1)}^v \neq \mathbb{F}_q^{n_1 n_2}$. Поэтому к таким кодам может быть применена атака AttackDCipher. В таблице 8 приводится оценка вероятности P_{attack} успеха атаки AttackDCipher для

$[n, k, d]$–кода $\overline{C(D)}$ на основе кодов Рида–Маллера $\mathrm{RM}_2(r,m)$ длины $n_1 = 2^{m_1} = n_2 = 2^{m_2}$, при $m_1 = m_2 = 8$. При этом в первом столбце таблицы приводится обозначение D-кода вида (2.36), где $m_1 = m_2 = 8$. Также для каждого D-кода экспериментально найдено значение K_p для шага 4 алгоритма 20 при $p = 0.01$. Видно, что вероятность P^{avg}_{attack} нахождения информационного сообщения с помощью AttackDCipher существенно больше, чем вероятность P_{ISD} успеха атаки на шифрограмму на основе декодирования по информационным совокупностям. Согласно формуле (3.12), минимальное количество хороших блоков для кодов из таблицы 8 равно 241. Отметим, что в этом случае вероятность успеха атаки P^{min}_{attack} для некоторых кодов из таблицы 8 может быть нулевой. Однако, согласно таблице 9, вероятность такого события пренебрежима.

Таблица 8 — Вероятность успеха атаки AttackDCipher для D-кодов на основе кодов Рида–Маллера

D-код	p	K_p	P_{ISD}	P^{min}_{attack}	P^{avg}_{attack}
[[4, 3], [5, 2]]	0.01	219	1.013E-34	1.117E-16	0.145
[[4, 3], [5, 2], [6, 1]]	0.01	247	2.646E-35	0	0.011
[[4, 3], [5, 2], [6, 1], [7, 0]]	0.01	255	2.536E-35	0	0.004

Таблица 9 — Вероятность появления плохих блоков для D-кодов из таблицы 8

N_g	$n_1 - N_g$	$Q_{N_g^{avg}}$	N_g	$n_1 - N_g$	$Q_{N_g^{avg}}$
256	0	0.999	248	8	3.850E-113
255	1	1.832E-12	247	9	1.724E-130
254	2	6.462E-25	246	10	7.830E-149
253	3	5.416E-38	245	11	2.466E-168
252	4	1.111E-51	244	12	3.142E-189
251	5	5.377E-66	243	13	7.092E-212
250	6	5.656E-81	242	14	6.895E-237
249	7	1.149E-96	241	15	1.415E-265

3.5 Параметры стойких систем $\mathrm{McE}(\overline{C(D)})$

В данном разделе определяются параметры D-кодов, для которых соответствующая криптосистема типа Мак–Элиса $\mathrm{McE}(\overline{C(D)}) = \mathrm{McE}(D,\mathsf{DDecoder})$ является стойкой к построенной выше атаке и может быть применена на практике в задаче инкапсуляции сеансового ключа, а также проводится сравнение предложенной криптосистемы на D-кодах с оригинальной системой Мак–Элиса на кодах Гоппы. Результаты данного раздела опубликованы в [49], [77], [76].

В [49] найдены условия на D-коды, при которых ключи криптосистемы типа Мак–Элиса на этих кодах являются слабыми: построена структурная атака, позволяющая найти часть секретного ключа. Эта атака не зависит от декодера, поэтому применима к слабым ключам всех вариантов схемы $\mathrm{McE}(D,\mathsf{DDecoder})$. В работе [77] и разделе 3.3 на основе структурной атаки из [49] и раздела 3.2 разработана комбинированная атака на шифрограмму схемы $\mathrm{McE}(D,\mathsf{DDecoder})$. Вероятность успеха комбинированной атаки оказывается достаточно большой, что численно продемонстрировано в случае слабых ключей схемы $\mathrm{McE}(D,\mathsf{GraphDecoder})$. Несмотря на отсутствие соответствующих численных подтверждений для схем $\mathrm{McE}(D,\mathsf{ISDDecoder})$ и $\mathrm{McE}(D,\mathsf{MatrixDecoder})$, естественно предположить нецелесообразность применения слабых ключей и в этих реализациях. Поэтому параметры схемы $\mathrm{McE}(D,\mathsf{DDecoder})$ следует определять, с одной стороны, исходя из обеспечения стойкости к структурной атаке, построенной в [49] и [77], а с другой стороны, исходя из требования обеспечения заданной OW–стойкости λ_{OW} (см. (1.23)).

В разделе 2.1.1 найдены также условия на D-код на кодах Рида–Маллера, при которых криптосистема $\mathrm{McE}(\overline{C(D)})$ является гарантированно стойкой к структурной атаке $\mathsf{AttackDKey}$ и, соответственно, к атаке $\mathsf{AttackDCipher}$. В частности, в теоремах 2, 3, 4 получены условия, при которых $\overline{C(D)}^2 = \mathbb{F}_2^{n_1 n_2}$. На основе этих результатов построен алгоритм $\mathsf{IsDCodeStrong}$, позволяющий для заданного D-кода на основе кодов Рида–Маллера определить его принадлежность

множеству слабых или сильных ключей (в смысле применимости построенной комбинированной атаки AttackDCipher). Алгоритм IsDCodeStrong программно реализован на языке Python (см. [58], `https://github.com/lelukevgeniy/Dissertation-D-codes/tree/main/Is%20D-code%20Strong`), получено свидетельство о государственной регистрации [83].

Алгоритм 21 IsDCodeStrong

Исходные параметры: $C = \overline{C(D)}$ – D-код вида (2.12)

если выполнены условия теоремы 1 **то**
    вернуть $False$;
если выполнены условия теоремы 2 или выполнены условия теоремы 3 или выполнены условия теоремы 4 **то**
    вернуть $True$;
$C' := C^2$;
пока $C' \neq \mathbb{F}_q^{n_1 n_2}$ **выполнять**
    если $\dim(C') \bmod 2^{m_1} = 0$ или $\dim(C') \bmod 2^{m_2} = 0$ **то**
        вернуть $False$;
    $C' := C' \star C$;
вернуть $True$.

В таблице 10 приводятся стойкие к атакам AttackDKey и AttackDCipher D-коды на кодах Рида–Маллера при $m_1 = m_2 = 8$, то есть для этих кодов алгоритм IsDCodeStrong возвращает значение $True$. В первом столбце таблицы приводится обозначение D-кода, полностью аналогичное представлению из таблицы 8. Также в таблице представлены параметры соответствующих кодов и оценка вероятности P_{ISD} успеха атаки по информационным совокупностям. При этом D-коды выбирались по следующему правилу. Среди D-кодов вида (2.12), обладающих гарантированной стойкостью к указанным выше атакам, для каждого $s \in [\![2,9]\!]$ выбирался код с наибольшей стойкостью к атаке по информационным совокупностям.

Согласно результатам, представленным в таблице 10, увеличение количества слагаемых в D-коде позволяет увеличить стойкость к атаке по информационным совокупностям за счет увеличения размерности и сохранения кодового расстояния. Однако отметим, что в соответствии с леммой 5, код

Таблица 10 — Стойкие к атакам AttackDKey, AttackDCipher D-коды

№	D-код	k	d	P_{ISD}
1*	$[[0,8],[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1],[8,0]]$	39203	256	1.715E-51
2*	$[[0,8],[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1]]$	39202	256	1.723E-51
3	$[[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1]]$	39201	256	1.732E-51
4	$[[1,7],[2,6],[3,5],[4,4],[5,3],[6,2]]$	39129	256	2.458E-51
5	$[[2,6],[3,5],[4,4],[5,3],[6,2]]$	39057	256	3.486E-51
6	$[[2,6],[3,5],[4,4],[5,3]]$	38021	256	4.796E-49
7	$[[3,5],[4,4],[5,3]]$	36985	256	5.498E-47
8	$[[2,5],[4,3]]$	19821	512	7.279E-41
9	$[[4,4]]$	26569	256	1.156E-29

с номером 1 из таблицы 10 является кодом Рида–Маллера $\mathrm{RM}_2(8,16)$, а код под номером 2 его подкодом коразмерности 1. Согласно [13], криптосистема $\mathrm{McE}(\mathrm{RM}_2(r,m))$ не является стойкой к структурным атакам, а в [59] показано, что подкоды кодов Рида–Маллера коразмерности 1 также не являются стойкими к структурным атакам. Следовательно, несмотря на стойкость к атакам AttackDKey и AttackDCipher, D-коды 1, 2 из таблицы 10 *не целесообразно* использовать в криптосистеме Мак–Элиса.

Далее приводится сравнение криптосистемы Мак–Элиса $\mathrm{McE}(G) = \mathrm{McE}(C)$, где C – код Гоппы, с предлагаемой криптосистемой $\mathrm{McE}(D) = \mathrm{McE}(\overline{C(D)})$ в случае применения декодеров для кода $\overline{C(D)}$, работающих как в пределах половины кодового расстояния, так и за этими пределами. Важным параметром асимметричных криптосистем является размер открытого ключа. При отсутствии структурных атак размер ключа системы Мак–Элиса $\mathrm{McE}(C)$ на $[n,k,d]_q$–коде C выбирается, исходя из параметра стойкости λ_{OW}. На значение λ_{OW}, согласно (1.23), влияет, в том числе, количество добавляемых при шифровании ошибок t (см. рис. 1.3). Далее сравнение проводится, в том числе, по следующим характеристикам: pk_size – размер открытого ключа (в мегабайтах, МБ), $R = k/n$ – информационная скорость кода, λ_{OW} – параметр стойкости.

В таблице 11 представлено сравнение криптосистем типа Мак–Элиса на кодах Гоппы ($\mathrm{McE}(G)$), двоичных кодах Рида–Маллера ($\mathrm{McE}(\mathrm{RM})$) и D-кодах

на основе двоичных кодов Рида–Маллера (McE(D)) (коды с номерами 3 и 8 из таблицы 10). Это сравнение выполнено для случая использования декодеров с гарантированным исправлением ошибок, то есть при шифровании вес вектора ошибок $\mathbf{e}$ равен $\lfloor (d-1)/2 \rfloor$. При этом параметры выбранного кода Гоппы соответствуют первому уровню стойкости оригинальной системы из таблицы 1.

Из таблицы 11 видно, что криптосистема на D-кодах в случае использования мажоритарного декодера имеет значительно больший размер открытого ключа, чем в системе Original McEliece, при сравнимой OW–стойкости. Однако отметим, что криптосистема на D-кодах близка по характеристикам к системе на кодах Рида–Маллера при использовании декодеров с гарантированным исправлением ошибок. Но значительным преимуществом криптосистемы на D-кодах является то, что существующие эффективные структурные атаки из [12] и [13] на систему, использующую коды Рида–Маллера, непосредственно не применимы к системе на D-кодах.

Таблица 11 — Сравнение характеристик криптосистем типа Мак-Элиса с использованием гарантированного декодера

	McE(G)	McE(RM)		McE(D)	
$[n, k, d]_2$	[3488, 2720, $\geqslant 129$]	[65536, 14893, 1024]	[65536, 39203, 256]	[65536, 19821, 512]	[65536, 39201, 256]
t	64	511	127	255	127
λ_{OW}	142.8	192.62	169.37	136.16	169.37
pk_size, МБ	1.13	116.35	306.27	154.85	306.25
R	0.78	0.23	0.6	0.3	0.6
Decoder	декодер Паттерсона	декодер Рида		мажоритарный декодер	
Структурные атаки	–	см. в [12], [13]		–	

Можно добиться увеличения стойкости системы Мак–Элиса к атакам на шифрограмму путем увеличения веса добавляемого вектора ошибок. Это, в свою очередь, позволяет для заданного уровня стойкости уменьшить размер

ключа. Согласно лемме 5, в ряде случаев D-код является подкодом некоторого кода Рида–Маллера. Это означает, что к D-кодам могут быть применены декодеры, работающие за пределами половины кодового расстояния (см. обзор [84]), в частности, декодер Сидельникова–Першакова [54], его модификации и списочный декодер Думера, а также декодер для низкоплотностных кодов [55], перестановочный декодер [56], рекурсивный декодер Йе–Аббэ [57]. Как говорилось ранее, декодеры, работающие за пределами половины кодового расстояния, обладают ненулевой вероятностью ошибочного декодирования и для некоторых асимметричных кодовых криптосистем, при многоразовом использовании ключей, ненулевое значение DFR может позволить атакующему найти секретный ключ [29], [85]. Для того чтобы исключить такую возможность, то есть для обеспечения IND-CCA-стойкости, следует руководствоваться правилом (1.3). Однако, при одноразовом использовании ключей в этом нет необходимости и можно использовать больший DFR в соответствии с отказоустойчивостью системы, что позволит использовать векторы ошибок большего веса. Так, например, в работе [10] для этого случая используется $\mathrm{DFR} \leqslant 10^{-9}$.

В таблице 12 представлено сравнение системы $\mathrm{McE}(G)$ с системой $\mathrm{McE}(D)$ в случае использования для кода $\overline{C(D)}$ декодера Сидельникова–Першакова [54]. Отметим, что для этого декодера нет теоретической оценки на DFR, а экспериментальные оценки имеются для небольших кодов, причем полученный DFR для этих кодов существенно больше 10^{-9}. Экспериментальное нахождение параметров кодов, для которых $\mathrm{DFR} \leqslant 10^{-9}$, вычислительно сложно, а для $\mathrm{DFR} \leqslant 2^{-128}$ – вычислительно невозможно. Поэтому в таблице 12 для DFR использованы результаты для теоретического декодера двоичных кодов Рида–Маллера, на которые можно ориентироваться при оценке DFR для декодера Сидельникова–Першакова (см. [54], следствие 2).

В таблице 12 для $\mathrm{McE}(D)$ отдельно рассматриваются случаи многоразового и одноразового использования ключа. Параметры систем на кодах Гоппы соответствуют уровням стойкости из таблицы 1. При этом, параметры систем $\mathrm{McE}(D)$ выбирались, во–первых, так, чтобы обеспечивалась стойкость к атаке

AttackDKey, а во–вторых, чтобы эти системы обладали наименьшим размером открытого ключа при не меньшем уровне стойкости λ_{OW} относительно соответствующих криптосистем Original McEliece. Из таблицы 12 видно, что для каждой из рассматриваемых систем Мак-Элиса на кодах Гоппы удается построить систему на D-кодах, обладающую либо большей OW–стойкостью при сопоставимом размере ключа, либо меньшим размером ключа при сопоставимой OW-стойкости.

Таблица 12 — Сравнение характеристик криптосистем типа Мак–Элиса при использовании декодера Сидельникова–Першакова для D-кодов

	Номер уровня стойкости	1(348864)	2(460896)	3(6688128)	4(6960119)	5(8192128)
McE(G)	$[n,k]_2$	$[3488, 2720]$	$[4608, 3360]$	$[6688, 5024]$	$[6960, 5413]$	$[8192, 6528]$
	t	64	96	128	119	128
	λ_{OW}	142.8	185.92	262.28	265.6	302.21
	pk_size, МБ	1.13	1.85	4.01	4.49	6.38
	R	0.78	0.73	0.75	0.78	0.8
McE(D), DFR $\leqslant 2^{-\lambda_{\mathsf{OW}}}$	$[n,k]_2$	$[16384, 435]$	$[32768, 485]$	$[65536, 585]$		
	t	3264	8419	20096		
	λ_{OW}	142.8	212.48	312.08		
	pk_size, МБ	0.85	1.89	4.57		
	R	0.03	0.01	0.01		
McE(D), DFR $\leqslant 10^{-9}$	$[n,k]_2$	$[16384, 435]$	$[32768, 485]$	$[32768, 541]$		$[65536, 585]$
	t	3712	9414	9414		22016
	λ_{OW}	166	239.04	268.92		348.6
	pk_size, МБ	0.85	1.89	2.11		4.57
	R	0.03	0.01	0.02		0.01

Другим способом увеличения количества исправляемых ошибок является использование построенных вероятностных декодеров ISDDecoder и MatrixDecoder. В таблице 13 приводятся параметры криптосистемы типа Мак–Элиса на основе D-кодов из таблицы 2 при использовании декодеров ISDDecoder и MatrixDecoder. Отметим, что эти коды являются гарантированно стойкими к атакам AttackDKey и AttackDCipher. Как было сказано ранее, в обоих алгоритмах ISDDecoder и MatrixDecoder предполагается использова-

ние декодера Йе–Аббе, который, согласно [57], имеет лучшую корректирую способность, чем декодер Сидельникова–Першакова.

В таблице 13 приводятся параметры криптосистемы типа Мак–Элиса на основе D-кодов из таблицы 2. В частности, приводится значение размера публичного ключа в мегабайтах, а также значение OW–стойкости λ_{OW} в соответствии с (1.23) при выбранном значении количества ошибок t.

Таблица 13 — Характеристики системы типа Мак–Элиса на D-кодах с использованием вероятностных декодеров

Номер кода	pk_size, МБ	t	λ_{OW}
1	0.734	4000	153.87
1	0.734	3749	142.82
2	0.809	3392	140.55
3	0.793	3392	137.8
4	0.393	1744	142.82
5	0.589	1194	142.81
6	1.328	1706	110.3
7	0.93	3830	185.96
8	1.463	612	186.11
9	1.541	612	197.24
10	1.121	3237	185.94
11	1.711	1706	143.04
12	1.588	1706	132.47
13	2.625	7687	262.28
13	2.625	7772	265.61
14	1.814	673	262.22
15	2.189	1706	184.58
16	4.035	1292	262.47

Продолжение таблицы 13

Номер кода	pk_size, МБ	t	λ_{OW}
17	3.064	1706	262.40
18	3.211	7279	302.23
19	5.441	1083	302.32

В таблице 14 приводится сравнение характеристик криптосистем на кодах Гоппы (MeE(G) в таблице 14) и криптосистемы типа Мак–Элиса на D-кодах. В качестве кодов Гоппы рассматриваются коды с характеристиками из таблицы 1, причем размер ключа вычисляется как для случая, когда шифрование выполняется по правилу, предложенному Р. Мак–Элисом (Original McEliece), так и для случая шифрования по правилу, предложенному Х. Нидеррайтером Classic McEliece. Для сравнения были выбраны D-коды из таблицы 2, которые при меньшем размере ключа, чем в Original McEliece, обладают такой же OW–стойкостью λ_{OW}, как и соответствующая система Мак–Элиса на кодах Гоппы. При этом среди множества подходящих D-кодов выбирался код с наименьшем временем декодирования при $\mathrm{DFR} \leqslant 10^{-9}$.

Как видно из таблицы 14, использование D-кодов в криптосистеме типа Мак–Элиса позволяет сократить размер публичного ключа относительно системы Original McEliece. Отметим также, что в таблице имеются D-коды, для которых криптосистема обладает меньшим размером ключа даже при значительно большей стойкости λ_{OW}. Например, система на коде 7 обладает меньшим размером открытого ключа и большим уровнем стойкости относительно системы Original McEliece с параметрами mceliece348864, а система на коде 18 – относительно системы с параметрами mceliece6960119. При этом стоит отметить, что для отказоустойчивости $\gamma = 9$ размер публичных ключей системы на D-кодах не столь существенно, но все же превышает размер публичных ключей системы Classic McEliece. Представляется, что размер ключей системы на

D-кодах может быть уменьшен, если ослабить требование к отказоустойчивости, то есть при $\gamma < 9$.

Таблица 14 — Сравнение характеристик криптосистем типа Мак–Элиса при использовании вероятностных декодеров для D-кодов

McE(G)					McE(D)			
Номер уровня стойкости	t	λ_{OW}	pk_size, Original McEliece, МБ	pk_size, Classic McEliece, МБ	Номер кода	t	λ_{OW}	pk_size, МБ
1	64	142.8	1.13	0.25	1	3749	142.82	0.73
2	96	185.92	1.85	0.5	7	3830	185.96	0.93
3	128	262.28	4.01	1	13	7687	262.28	2.63
4	119	265.6	4.49	1	13	7772	265.61	2.63
5	128	302.21	6.38	1.29	18	7279	302.23	3.21

Сравнивая криптосистемы Мак–Элиса на кодах Гоппы и D-кодах, можно заметить, что время расшифрования у последней больше, как при декодировании алгоритмом ISDDecoder, так и алгоритмом MatrixDecoder. Однако отметим, что в таблице 1 приводится время расшифрования при использовании оптимизированной реализации декодера из [64], в то время как в таблицах 4 и 5 указано время работы алгоритмов, для которых не ставилась цель создания оптимальной реализации. В действительности существует большой потенциал для ускорения реализации алгоритмов ISDDecoder и MatrixDecoder. Например, они хорошо поддаются распараллеливанию, в том числе за счет параллельного декодирования блоков в BlockDecoder.

3.6 Механизм KEM на основе McE(D,DDecoder)

Тройку алгоритмов механизма инкапсуляции, полученного путем применения преобразования Фудзисаки–Окамото $\mathsf{FO}^{\not\perp}$ на основе схемы шифрования

McE(D,DDecoder), обозначим KEM(D,DDecoder) = $(\mathsf{Gen}^{\mathsf{KEM}(D)}, \mathsf{Encaps}, \mathsf{Decaps})$ (см. рис. 3.3).

Согласно [27], при выполнении условия (1.3) для многоразового использования ключей, механизм KEM, полученный применением преобразования Фудзисаки–Окамото $\mathsf{FO}^{\not\perp}$, обеспечивает IND–CCA–стойкость λ_{KEM}, если для схемы асимметричного шифрования McE(D,DDecoder) выполняется условие (1.1). Отсюда вытекает и IND–CPA–стойкость механизма, при выполнении тех же условий. При одноразовом использовании ключей, когда для атакующего информация об ошибках декапсуляции не представляет ценности, IND–CPA–стойкость механизма, изображенного на рис. 3.3 может быть схематично обоснована также следующим образом. Пусть схема McE(D,DDecoder) обладает OW–стойкостью $\lambda_{\mathsf{OW}} \geqslant \lambda_{\mathsf{KEM}}$, а атакующему известным pk', $\mathbf{c}$ и $\mathbf{K}'$, где либо $\mathbf{K}' \leftarrow \{0,1\}^m$, либо $(\mathbf{K}',c) \leftarrow \mathsf{Encaps}(pk')$. В рамках модели со случайным оракулом (когда криптографическая хеш–функция моделируется «идеальной» случайной хеш–функцией) значение функции H неотличимо от случайной равномерно распределенной на $\{0,1\}^m$ величины. Поэтому, в рамках одноразового использования ключей, единственная возможность у атакующего определить способ генерации $\mathbf{K}'$ – это по $\mathbf{c}$ и pk' найти $\mathbf{r}$, что невозможно, в силу OW–стойкости схемы шифрования.

3.6.1 О характеристиках KEM(D,DDecoder)

Выше отмечалось, что выбор параметров схемы McE(D,DDecoder) следует выполнять, исходя из обеспечения стойкости к структурной атаке, построенной в данной главе, и исходя из требования обеспечения заданной OW–стойкости λ_{OW}. При синтезе механизма KEM(D,DDecoder) на основе McE(D,DDecoder) (см. рис. 3.3) дополнительными условиями на выбор параметров схемы McE(D,DDecoder), являются условия (1.1) и (1.3).

$\mathsf{Gen}^{\mathsf{KEM}(D)}(t)$:

---

1. $\mathbf{s} \leftarrow \mathbb{F}_2^k$
2. $(pk, sk) \leftarrow \mathsf{Gen}^{\mathrm{McE}(D)}(t)$
3. $pk' = pk$
4. $sk' = (sk, \mathbf{s})$
5. ***вернуть*** (pk', sk')

$\mathsf{Encaps}(pk')$:

---

1. $\mathbf{r} \leftarrow \mathbb{F}_2^k$
2. $\mathbf{c} \leftarrow \mathsf{Enc}(pk', \mathbf{r}; \mathsf{G}(\mathsf{r}))$
3. $\mathbf{K} \leftarrow \mathsf{H}(\mathbf{r} \parallel \mathbf{c})$
4. ***вернуть*** $(\mathbf{K}, \mathbf{c})$

$\mathsf{Decaps}(pk', sk' = (sk, \mathbf{s}), \mathbf{c})$:

---

1. $\mathbf{r}' \leftarrow \mathsf{Dec}^{\mathsf{DDecoder}}(sk, \mathbf{c})$
2. *если* $\mathbf{r}' \neq \perp$
3. wt : $\mathbf{c}' \leftarrow \mathsf{Enc}(pk', \mathsf{r}'; \mathsf{G}(\mathbf{r}'))$
4. wt : *если* $\mathbf{c} = \mathbf{c}'$
5. ***вернуть*** $\mathsf{H}(\mathbf{r}' \parallel \mathbf{c})$
6. ***вернуть*** $\mathsf{H}(\mathbf{s} \parallel \mathbf{c})$

Рисунок 3.3 — Механизм $\mathsf{KEM}(D,\mathsf{DDecoder}) = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$, полученный путем применения преобразования $\mathsf{FO}^{\not\perp}$ на основе схемы $\mathrm{McE}(D,\mathsf{DDecoder})$ с детерминированным правилом шифрования. Если в $\mathsf{DDecoder}$ применяется алгоритм $\mathsf{ExitCond}_{\mathsf{wt}}$, то в правиле Decaps выполняются все строки; если же используется алгоритм $\mathsf{ExitCond}_{\mathsf{G}}$, то выполняются строки 1,2,5,6.

Схема $\mathrm{McE}(D,\mathsf{DDecoder})$ в случае применения гарантированного декодера, например, $\mathsf{GraphDecoder}$, описанного в разделе 2.3.2, позволяет построить $\mathsf{KEM}(D,\mathsf{GraphDecoder})$ с многоразовым использованием ключей, так как условие (1.3) выполняется ($\delta = 0$). Однако результаты исследования [77] показали, что для обеспечения целевых значений OW–стойкости, сопоставимых со значениями из таблицы 1, приходится использовать ключи существенно большего размера, чем в Original McEliece.

Как показывают результаты раздела 3.5, при использовании разработанных вероятностных декодеров ISDDecoder и MatrixDecoder в настоящей работе не удалось найти D-коды, для которых бы размер ключа криптосистемы не превышал размер ключа системы Original McEliece из таблицы 1, обеспечивая при этом стойкость не менее соответствующего значения λ_{OW}, и для которых вероятность ошибочного декодирования не превышала бы $2^{-\lambda_{\mathsf{OW}}}$ (при $I_{max} \leqslant 10^6$). Как следствие, найденные D-коды (см. таблицу 2) для схемы McE(D,DDecoder) с построенными вероятностными декодерами не позволяют при меньшем размере ключа, чем в Original McEliece, построить механизм KEM(D,DDecoder) с соответствующей IND–CCA–стойкостью λ_{KEM}. Действительно, для обеспечения IND–CCA–стойкости необходимо, с одной стороны, чтобы уровень стойкости λ_{KEM} был не меньше, чем длина m сеансового ключа K (см. (1.1)), которая обычно не меньше 128, а с другой – чтобы DFR был меньше, чем $2^{-\lambda_{\mathsf{KEM}}}$ (см. (1.3)). В то же время, согласно таблицам 3 и 5, минимальный полученный DFR для используемых параметров D-кодов оказался значительно больше, чем 2^{-128}. Однако найденные коды позволяют обеспечить отказоустойчивость $\gamma = 9$ для IND–CPA–стойкого механизма KEM с одноразовым использованием ключей (см. условие (1.3) для одноразового использования). В частности, такие коды удалось найти за счет подбора максимального числа итераций I_{max} в алгоритме ISDDecoder.

Таким образом, система Мак–Элиса на кодах Гоппы (как Original McEliece, так и Classic McEliece), в силу нулевой вероятности ошибочного декодирования, позволяет построить KEM с многоразовым использованием ключей, и как следствие, этот же механизм может применяться для одноразового использования. В то же время схема Мак–Элиса на D-кодах при *сопоставимых размерах ключа* (меньшем размере по отношению к Original McEliece и несколько большем размере по отношению к Classic McEliece) в случае использования разработанных вероятностных алгоритмов декодирования 18 и 19 позволяет построить только KEM с одноразовым использованием ключей.

3.7 Выводы

В главе выполнен синтез схемы инкапсуляции сеансового ключа (KEM) на основе системы типа Мак–Элиса на D-кодах на основе кодов Рида–Маллера, а также исследована ее стойкость. А именно, описан алгоритм выделения множеств сильных и слабых ключей системы на D-кодах. Для слабых ключей криптосистемы построена комбинированная атака, позволяющая с помощью структурной атаки значительно повысить эффективность атаки на шифрограмму, и приводится оценка ее эффективности. Для сильных ключей криптосистемы найдены параметры для возможности практического применения. В конце главы проводится сравнение с оригинальной системой на кодах Гоппы при использовании разных подходов к декодированию D-кодов, где показана конкурентоспособность построенной схемы относительно оригинальной системы Мак-Элиса.

Известные асимметричные кодовые криптосистемы на основе кодов Рида–Маллера (см. [8; 19; 86]) оказались нестойкими к структурным атакам (см. [12; 13; 59; 87-89]). В данной главе описаны результаты попытки реабилитации двоичных кодов Рида–Маллера в контексте их применения в асимметричных системах шифрования. Для достижения этой цели используется, с одной стороны, конструкция D-кодов (для противодействия известным структурным атакам), а с другой – наличие относительно эффективных декодеров для кодов Рида–Маллера, способных исправлять ошибки веса более половины кодового расстояния (для уменьшения размера публичного ключа). Благодаря этому удалось построить механизм KEM с меньшим ключом, чем в оригинальной криптосистеме на кодах Гоппы Original McEliece, только для передачи эфемерных (одноразовых) сессионных криптографических ключей. Однако, учитывая интерес криптографического сообщества к KEM с одноразовым использованием ключей [33], это ограничение представляется не таким существенным. Кроме того, как было недавно показано, двоичные

коды Рида–Маллера достигают емкости в симметричных каналах без памяти [47; 90], поэтому ожидается дальнейшее развитие методов эффективного декодирования этих кодов, которые могут позволить не только уменьшить время расшифрования в предлагаемой криптосистеме, но и повысить класс стойкости за счет уменьшения вероятности ошибочного декодирования. Возможность исправлять ошибки большого веса также может быть использована для усиления структурной стойкости системы, например, за счет добавления случайных столбцов к порождающей матрице кода и/или замены матрицы перестановки P невырожденной матрицей специального вида.

Заключение

Основным результатом работы является решение актуальной научной задачи, имеющей практическую значимость: построена асимметричная кодовая криптосистема типа Мак–Элиса на D-кодах на основе кодов Рида–Маллера, которая обладает либо большей стойкостью при сопоставимом размере ключа, либо меньшим размером ключа при сопоставимой стойкости, либо большей стойкостью при меньшем размере ключа по сравнению с системой Original McEliece. Это позволяет применять предложенную систему в механизме инкапсуляции сеансового ключа симметричной криптосистемы, а также для повышения защищенности данных, циркулирующих в информационных системах, за счет использования рандомизированного шифрования.

В процессе работы были получены следующие результаты:

1. Разработана алгоритмическая модель мажоритарного декодирования D-кодов.
2. Разработаны алгоритмы вероятностного декодирования D-кодов на основе кодов Рида–Маллера.
3. Получены условия разложимости квадрата Шура–Адамара D-кодов на основе кодов Рида–Маллера в прямую сумму неразложимых кодов.
4. Построена комбинированная атака для слабых ключей криптосистемы типа Мак–Элиса на основе подкодов прямой суммы кодов, использующая структурную атаку для атаки на шифрограмму.
5. Найдены параметры криптосистемы типа Мак–Элиса на D-кодах на основе кодов Рида–Маллера, соответствующие сильным ключам, для которых размер открытого ключа меньше, чем у оригинальной системы Мак–Элиса на кодах Гоппы.

В завершение настоящего исследования считаю важным выразить глубокую признательность и искреннюю благодарность и почтить светлую память моего первого научного руководителя, к.ф.-м.н. Деундяка Владимира Михай-

ловича, сыгравшего ключевую роль в становлении меня как исследователя. Его вера в мои способности, неугасимый энтузиазм, трудолюбие и искреннее увлечение наукой вдохновили меня на выбор научного пути и заложили основу всей последующей работы. Он навсегда останется в памяти примером истинного ученого и прирожденного наставника.

Отдельную благодарность выражаю моему научному руководителю, к.т.н. Косолапову Юрию Владимировичу, под чьим руководством была завершена данная работа. Его поддержка, внимание к деталям, оптимизм и уверенность в успешном завершении исследования оказались неоценимыми в процессе преодоления научных и организационных трудностей. Благодаря его помощи и профессионализму удалось довести диссертационное исследование до логического и научно обоснованного завершения.

Список литературы

1. *Shor, P. W.* Algorithms for quantum computation: Discrete logarithms and factoring / P. W. Shor // Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press. – 1994. – P. 124-134.
2. *Sendrier, N.* Code-Based Cryptography: New Security Solutions Against a Quantum Adversary / N. Sendrier, J. P. Tillich // ERCIM News, ERCIM, 2016, Special Theme Cybersecurity (106). <hal-01410068>. – 2016.
3. *McEliece, R. J.* A Public-Key Cryptosystem Based on Algebraic Coding Theory / R. J. McEliece // JPL Deep Space Network Progress Report. – 1978. – Vol. 42. – P. 114-116.
4. Nts-kem / M. R. Albrecht [et al.] // NIST PQC Round. – 2019. – Vol. 2. – P. 4-13.
5. A Public-Key Cryptosystem Based on Algebraic Coding Theory / D. J. Bernstein [et al.] // NIST submissions. – 2017.
6. *Niederreiter, H.* Knapsack-type cryptosystems and algebraic coding theory / H. Niederreiter // Prob. Contr. Inform. Theory. – 1986. – Vol. 15, no. 2. – P. 157-166.
7. *Kobara, K.* Semantically secure McEliece public-key cryptosystems-conversions for McEliece pkc / K. Kobara, H. Imai // In International Workshop on Public Key Cryptography. Springer. – 2001. – Vol. 2. – P. 19-35.
8. *Sidel'nikov, V. M.* Open coding based on Reed–Muller binary codes / V. M. Sidel'nikov // Diskretnaya matematika. – 1994. – Vol. 6, no. 2. – P. 3-20.
9. *Janwa, H.* McEliece public key cryptosystems using algebraic-geometric codes / H. Janwa, O. Moreno // Designs, Codes and Cryptography. – 1996. – Vol. 8, no. 3. – P. 293-307.

10. *Baldi, M.* Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes / M. Baldi, M. Bianchi, F. Chiaraluce // IET Information Security. – 2013. – Vol. 7, no. 3. – P. 212-220.

11. *Сидельников, В. М.* О системе шифрования, построенной на основе обобщенных кодов Рида–Соломона / В. М. Сидельников, С. О. Шестаков // Дискретная математика. – 1992. – Т. 4, № 3. – С. 57-63.

12. *Minder, L.* Cryptanalysis of the Sidelnikov cryptosystem / L. Minder, A. Shokrollahi // In Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Springer Berlin Heidelberg. – 2007. – Vol. 26. – P. 347-360.

13. *Borodin, M. A.* Effective attack on the McEliece cryptosystem based on Reed-Muller codes / M. A. Borodin, I. V. Chizhov // Discrete Mathematics and Applications. – 2014. – Vol. 24, no. 5. – P. 273-280.

14. *Couvreur, A.* Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes / A. Couvreur, I. Márquez-Corbella, P. R. // IEEE Transactions on Information Theory. – 2017. – Vol. 63, no. 8. – P. 5404-5418.

15. A reaction attack on the QC-LDPC McEliece cryptosystem / T. Fabšič [et al.] // In Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8. – 2017. – P. 51-68.

16. *Couvreur, A.* Polynomial time attack on wild McEliece over quadratic extensions / A. Couvreur, A. Otmani, J.-P. Tillich // IEEE Transactions on Information Theory. – 2016. – Vol. 63, no. 1. – P. 404-427.

17. *Elbro, F.* An Algebraic Attack Against McEliece-like Cryptosystems Based on BCH Codes / F. Elbro, C. Majenz // Cryptology ePrint Archive. – 2022.

18. *Couvreur, A.* On the Security of Subspace Subcodes of Reed–Solomon Codes for Public Key Encryption / A. Couvreur, M. Lequesne // IEEE Transactions on Information Theory. – 2021.

19. A new code-based public-key cryptosystem resistant to quantum computer attacks. In Journal of Physics: Conference Series / E. Egorova [et al.] // IOP Publishing. – 2019. – Vol. 1163, no. 1. – P. 012061.

20. О новых задачах в асимметричной криптографии, основанной на помехоустойчивом кодировании / В. В. Зяблов [и др.] // Проблемы передачи информации. – 2022. – Т. 58, № 2. – С. 92-111.

21. *Deundyak, V. M.* On the decipherment of sidel'nikov-type cryptosystems / V. M. Deundyak, Y. V. Kosolapov, I. A. Maystrenko // In Code-Based Cryptography: 8th International Workshop, CBCrypto 2020, Zagreb, Croatia, May 9–10, 2020, Revised Selected Papers. – 2020. – P. 20-40.

22. *Vedenev, K.* Cryptanalysis of Ivanov-Krouk-Zyablov cryptosystem / K. Vedenev, Y. Kosolapov // In Code-Based Cryptography Workshop. Springer, Cham. – 2023. – P. 137-153.

23. U.S. Patent No. 7,861,131 / J. Xu [et al.] // Washington, DC: U.S. Patent and Trademark Office. – 2010.

24. U.S. Patent No. 10,333,554 / M. Twitto [et al.] // Washington, DC: U.S. Patent and Trademark Office. – 2019.

25. U.S. Patent No. 8,321,769 / E. Yeo [et al.] // Washington, DC: U.S. Patent and Trademark Office. – 2012.

26. *Chizhov, I.* A Hadamard Product of Linear Codes: Algebraic Properties and Algorithms for Calculating It / I. Chizhov // MoscowUniv.Comput.Math.Cybern. – 2023. – Vol. 47, no. 4. – P. 239-250.

27. *Hofheinz, D.* A modular analysis of the Fujisaki-Okamoto transformation / D. Hofheinz, K. Hovelmanns, E. Kiltz // In Theory of Cryptography Conference, Cham: Springer International Publishing. – 2017. – P. 341-371.

28. BIKE: bit flipping key encapsulation [Электронный ресурс] / N. Arago [и др.]. – 2017. – URL: http://bikesuite.org (дата обр. 07.02.2025).

29. *Guo, Q.* A key recovery reaction attack on QC-MDPC / Q. Guo, T. Johansson, P. S. Wagner // IEEE Transactions on Information Theory. – 2018. – Vol. 65, no. 3. – P. 1845-1861.

30. *Vedenev, K.* A Reaction Attack against Cryptosystems Based on Quasi–Group MDPC Codes / K. Vedenev, Y. Kosolapov // In 2023 XVIII International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY). – 2023. – P. 70-75.

31. *Joux, A.* Kleptographic Attacks against Implicit Rejection [Электронный ресурс] / A. Joux, J. Loss, B. Wagner. – 2024. – URL: https://eprint.iacr.org/2024/260 (дата обр. 07.02.2025).

32. *Campagna, M.* Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS) / M. Campagna, E. Crockett // Internet-Draft: draft-campagna-tls-bike-sike-hybrid-07. Internet Engineering Task Force. – 2021.

33. *Drucker, N.* A lean BIKE KEM design for ephemeral key agreement [Электронный ресурс] / N. Drucker, S. Gueron, D. Kostic. – 2024. – URL: https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/a-lean-bike-kem.pdf (дата обр. 07.02.2025).

34. Quantum-resistant Transport Layer Security / C. R. Garcia [et al.] // Computer Communications. – 2024. – Vol. 213. – P. 345-358.

35. *Сидельников, В. М.* Теория кодирования / В. М. Сидельников. – М. : ФИЗМАТЛИТ, 2008.

36. *Randriambololona, H.* On products and powers of linear codes under componentwise multiplication / H. Randriambololona // Algorithmic arithmetic, geometry, and coding theory. – 2015. – Vol. 637. – P. 3-78.

37. *Циммерман, К.-Х.* Методы теории модулярных представлений в алгебраической теории кодирования / К.-Х. Циммерман. – М. : МЦНМО, 2011.

38. *Curtis, C. W.* Representation Theory of Finite Groups and Associative Algebras / C. W. Curtis, I. Reiner. – New York : Intersclence Publishers, 1962.

39. *Деундяк, В. М.* Алгоритмы для мажоритарного декодирования групповых кодов / В. М. Деундяк, Ю. В. Косолапов // Модел. и анализ информ. систем. – 2015. – Т. 22, № 4. – С. 464-482.

40. *Morelos-Zaragoza, R. H.* The Art of Error Correcting Coding / R. H. Morelos-Zaragoza. – John Wiley & Sons, Ltd, 2006.

41. *Grassl, M.* Quantum block and convolutional codes from self-orthogonal product codes / M. Grassl, M. Rotteler // Proc. IEEE Int. Symp. Inf. Theory, Sep. – 2005. – P. 1018-1022.

42. *Деундяк, В. М.* О некоторых свойствах произведения Шура - Адамара для линейных кодов и их приложениях / В. М. Деундяк, Ю. В. Косолапов // Прикладная дискретная математика. – 2020. – № 50. – С. 72-86.

43. *Prange, E.* The use of information sets in decoding cyclic codes / E. Prange // IRE Transactions on Information Theory. – 1962. – Vol. 8, no. 5. – P. 5-9.

44. *May, A.* Decoding random linear codes in $\tilde{O}(2^{0.54n})$ / A. May, A. Meurer, E. Thomae // International Conference on the Theory and Application of Cryptology and Information Security. – 2011. – P. 107-124.

45. Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding / A. Becker [et al.] // Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, Proceedings 31. – 2012. – P. 520-536.

46. *Massey, J. L.* Threshold Decoding / J. L. Massey // Cambridge:MIT Press. – 1963.

47. *Abbe, E.* A proof that Reed-Muller codes achieve Shannon capacity on symmetric channels / E. Abbe, C. Sandon // In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS). – 2023. – P. 177-193.

48. *Chizhov, I. V.* Effective attack on the McEliece cryptosystem based on Reed-Muller codes / I. V. Chizhov, M. A. Borodin // Discrete Math. Appl. – 2014. – Vol. 24, no. 5. – P. 273-280.

49. *Косолапов, Ю. В.* О структурной стойкости криптосистемы типа Мак-Элиса на сумме тензорных произведений бинарных кодов Рида-Маллера / Ю. В. Косолапов, Е. А. Лелюк // Прикладная дискретная математика. – 2022. – № 57. – С. 22-39.

50. *Берман, С. Д.* К теории групповых кодов / С. Д. Берман // Кибернетика. – 1967. – Т. 3, № 1. – С. 31-39.

51. *Логачев, О. А.* Булевы функции в теории кодирования и криптологии / О. А. Логачев, А. Ю. Сальников, В. В. Ященко. – М.:МЦМНО, 2004. – 470 с.

52. *Reed, I.* A class of multiple-error-correcting codes and the decoding scheme / I. Reed // IEEE Transactions on Information Theory. – 1954. – Vol. 4, no. 4. – P. 38-492.

53. *Циммерман, К.-Х.* Методы теории модулярных представлений в алгебраической теории кодирования / К.-Х. Циммерман. – М. : МЦНМО, 2011.

54. *Сидельников, В. М.* Декодирование кодов Рида–Маллера при большом числе ошибок / В. М. Сидельников, А. С. Першаков // Пробл. передачи информ. – 1992. – Т. 28, № 3. – С. 80-94.

55. Iterative Reed–Muller Decoding / M. Geiselhart [et al.] // In 2021 11th International Symposium on Topics in Coding (ISTC). IEEE. – 2021. – P. 1-5.

56. A new permutation decoding method for Reed-Muller codes / M. Kamenev [et al.] // In 2019 IEEE International Symposium on Information Theory (ISIT). IEEE. – 2019. – P. 26-30.

57. *Ye, M.* Recursive projection-aggregation decoding of Reed-Muller codes / M. Ye, E. Abbe // IEEE Transactions on Information Theory. – 2020. – Vol. 66, no. 8. – P. 4948-4965.

58. *Лелюк, Е. А.* Dissertation-D-codes [Электронный ресурс] / Е. А. Лелюк. – 2026. – URL: https://github.com/lelukevgeniy/Dissertation-D-codes (дата обр. 28.01.2026).

59. *Бородин, М. А.* Классификация произведений Адамара подкодов коразмерности 1 кодов Рида-Маллера / М. А. Бородин, И. В. Чижов // Дискретная математика. – 2020. – Т. 32, № 1. – С. 115-134.

60. Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes / A. Couvreur [et al.] // Designs, Codes and Cryptography. – 2014. – Vol. 73, no. 2. – P. 641-666.

61. *Otmani, A.* Square code attack on a modified Sidelnikov cryptosystem / A. Otmani, H. T. Kalachi // International Conference on Codes, Cryptology, and Information Security. – Springer. 2015. – P. 173-183.

62. *Wieschebrink, C.* Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes / C. Wieschebrink // International Workshop on Post-Quantum Cryptography. – Springer. 2010. – P. 61-72.

63. Classic McEliece: Conservative Code-Based Cryptography: Round-4 submission package [Электронный ресурс] / M. R. Albrecht [и др.]. – 2022. – URL: https://classic.mceliece.org/nist/mceliece-20221023.tar.gz (дата обр. 19.01.2026).

64. Classic McEliece: Conservative Code-Based Cryptography: cryptosystem specification [Электронный ресурс] / M. R. Albrecht [и др.]. – 2022. – URL: https://classic.mceliece.org/mceliece-spec-20221023.pdf (дата обр. 19.01.2026).

65. *Kasami, T.* On the Construction of a Class of Majority–Logic Decodable Codes / T. Kasami, S. Lin // IEEE Transactions on Information Theory. – 1971. – Vol. IT-17, no. 5. – P. 600-610.

66. *Блох, Э. Л.* Кодирование обобщенных каскадных кодов / Э. Л. Блох, В. В. Зяблов // Пробл. передачи информ. – 1974. – Т. 10, № 3. – С. 45-50.

67. *Зиновьев, В. А.* Обобщенные каскадные коды / В. А. Зиновьев // Пробл. передачи информ. – 1976. – Т. 12, № 1. – С. 5-15.

68. *Косолапов, Ю. В.* О разложимости произведения Шура — Адамара суммы тензорных произведений кодов Рида — Маллера / Ю. В. Косолапов, Е. А. Лелюк // Прикладная дискретная математика. Приложение. – 2021. – № 14. – С. 158-161.

69. *Deundyak, V. M.* On the decipherment of Sidel'nikov-type cryptosystems / V. M. Deundyak, Y. V. Kosolapov, I. A. Maystrenko // LNCS, 12087. – 2020. – P. 20-40.

70. *Henderson, H. V.* The vec-permutation matrix, the vec operator and Kronecker products: A review / H. V. Henderson, S. R. Searle // Linear and Multilinear Algebra. – 1981. – № 9. – С. 271-288.

71. *Deundyak, V. M.* A Graph-Theoretical Method for Decoding Some Group MLD-Codes / V. M. Deundyak, E. A. Lelyuk // Journal of Applied and Industrial Mathematics. – 2020. – Vol. 14, no. 2. – P. 265-280.

72. *Deundyak, V. M.* Decoding the Tensor Product of MLD Codes and Applications for Code Cryptosystems / V. M. Deundyak, Y. V. Kosolapov, E. A. Lelyuk // Automatic Control and Computer Sciences. – 2019. – Vol. 52, no. 7. – P. 647-657.

73. *Лелюк, Е. А.* Декодирование М-ортогонального семейства кодов / Е. А. Лелюк // Современные информационные технологии: тенденции и перспективы развития: материалы XXVI научной конференции, Ростов-на-Дону, 18–19 апреля 2019 г./ Редакционная коллегия: Г. В. Муратова, Я. М.

Ерусалимский, С. С. Михалкович, В. С. Пилиди, В. Ю. Тополов. – Ростов-на-Дону: Южный федеральный университет. – 2019. – С. 164-166.

74. *Деундяк, В. М.* О декодировании кодов на основе D-конструкции / В. М. Деундяк, Е. А. Лелюк // Современные информационные технологии: тенденции и перспективы развития : материалы XXV научной конференции, Ростов-на-Дону, 17–18 мая 2018 г./ Редакционная коллегия: Г. В. Муратова, Я. М. Ерусалимский, С. С. Михалкович, В. С. Пилиди, В. Ю. Тополов. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета. – 2018. – С. 63-64.

75. *Лелюк, Е. А.* Декодирование тензорных произведений кодов на некоммутативных группах / Е. А. Лелюк // Сборник тезисов участников форума "Наука будущего – наука молодых [Нижний Новгород, 12–14 сентября 2017 г. : в 2 т.]. Т. 1. – Москва ; Нижний Новгород : Инконсалт К. – 2017. – С. 126-127.

76. *Косолапов, Ю. В.* О параметрах системы шифрования типа Мак-Элиса на D-кодах, основанных на двоичных кодах Рида-Маллера / Ю. В. Косолапов, Е. А. Лелюк // Прикладная дискретная математика. – 2025. – № 67. – С. 7-35.

77. *Косолапов, Ю. В.* Криптосистема типа Мак-Элиса на D-кодах / Ю. В. Косолапов, Е. А. Лелюк // Математические вопросы криптографии. – 2024. – Т. 15, № 2. – С. 69-90.

78. *Деундяк, В. М.* Анализ стойкости некоторых кодовых криптосистем, основанный на разложении кодов в прямую сумму / В. М. Деундяк, Ю. В. Косолапов // Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование. – 2019. – Т. 12, № 3. – С. 89-101.

79. A CCA secure variant of the McEliece cryptosystem / N. Dottling [et al.] // IEEE Trans. Inf. Theory. – 2012. – Vol. 58. – P. 6672-6680.

80. *Сачков, В. Н.* Введение в комбинаторные методы дискретной математики / В. Н. Сачков. – М. : МЦНМО, 2004.

81. *Brent, R.* Random Krylov Spaces over Finite Fields / R. Brent, S. Gao, A. Lauder // SIAM Journal on Discrete Mathematics. – 2002. – Vol. 16, no. 2. – P. 276-287.

82. *Деундяк, В. М.* Использование тензорного произведения кодов Рида-Маллера в асимметричной криптосистеме типа Мак-Элиса и анализ ее стойкости к атакам на шифрограмму / В. М. Деундяк, Ю. В. Косолапов // Вычислительные технологии. – 2017. – Т. 22, № 4. – С. 43-60.

83. *Свидетельство о государственной регистрации программы для ЭВМ № 2024691736.* Программное средство определения множества сильных и слабых ключей криптосистемы на D-кодах на основе кодов Рида-Маллера / Е. А. Лелюк. – № 2024690988 ; заявл. 14.12.2024 ; опубл. 24.12.2024 (Российская Федерация).

84. *Abbe, E.* Reed–Muller codes: Theory and algorithms / E. Abbe, A. Shpilka, M. Ye // IEEE Transactions on Information Theory. – 2020. – Vol. 67, no. 6. – P. 3251-3277.

85. *Vedenev, K.* Theoretical analysis of decoding failure rate of non-binary QC-MDPC codes / K. Vedenev, Y. Kosolapov // Cryptology ePrint Archive, Paper 2023/1224. – 2023. – P. 1-20.

86. *Kabatiansky G.* A new code-based cryptosystem via pseudorepetition of codes / Kabatiansky G., Tavernier C. // Sixteenth International Workshop on Algebraic and Combinatorial Coding Theory, Svetlogorsk (Kaliningrad region), Russia. – 2018. – P. 189-191.

87. *Высоцкая, В. В.* Квадрат кода Рида-Маллера и классы эквивалентности секретных ключей криптосистемы Мак-Элиса-Сидельникова / В. В. Высоцкая // Прикладная дискретная математика. Приложение. – 2017. – № 10. – С. 66-68.

88. *Vysotskaya, V.* Equivalence classes of McEliece-Sidelnikov-type cryptosystems / V. Vysotskaya, I. Chizhov // Sixteenth International Workshop on Algebraic and Combinatorial Coding Theory, Svetlogorsk (Kaliningrad region), Russia. – 2018. – P. 121-124.

89. *Давлетшина, А. М.* Поиск эквивалентных ключей криптосистемы Мак-Элиса - Сидельникова, построенной на двоичных кодах Рида - Маллера / А. М. Давлетшина // Прикладная дискретная математика. Приложение. – 2019. – № 12. – С. 98-100.

90. Reed-Muller codes achieve capacity on erasure channels / S. Kudekar [et al.] // In Proceedings of the forty-eighth annual ACM symposium on Theory of Computing. – 2016. – P. 658-669.

Список рисунков

Список таблиц

Приложение А

Акты о внедрении результатов работы

Далее приводятся акты о внедрении и/или использовании результатов диссертационной работы, предъявленные во введении.

УТВЕРЖДАЮ
Генеральный директор АНО «НТЦ ЦК»

Качалин И.Ф.
«___» 12 2024 г.

АКТ

об использовании результатов

кандидатской диссертационной работы

«Синтез постквантовой схемы инкапсуляции сеансового ключа»

Лелюка Евгения Андреевича

Комиссия в составе:

Председатель:

Алексеев Евгений Константинович

Члены комиссии:

Чижов Иван Владимирович

Полтавская Ирина Вячеславовна

составили настоящий акт о том, что результаты диссертационной работы Лелюка Евгения Андреевича «Синтез постквантовой схемы инкапсуляции сеансового ключа», представленной на соискание ученой степени кандидата технических наук, использованы в деятельности Автономной Некоммерческой Организации «Национальный Технологический Центр Цифровой Криптографии» при выполнении научно-исследовательской работы в виде результатов анализа и исследования стойкости постквантовых криптографических механизмов.

Результаты диссертационной работы Лелюка Евгения Андреевича:

1. Разработан алгоритм исследования схемы инкапсуляции сеансового ключа (KEM) на основе кодовой криптосистемы.
2. Разработаны алгоритмы гарантированного и вероятностного декодирования D-кодов на основе кодов Рида-Маллера для применения в кодовых схемах шифрования.
3. Исследованы криптографические свойства D-кодов, в частности, получены условия разложимости квадрата Шура-Адамара D-кодов на основе кодов Рида-Маллера в прямую сумму неразложимых кодов.
4. Разработан алгоритм определения множества сильных и слабых ключей криптосистемы типа Мак-Элиса на D-кодах на основе кодов Рида-Маллера.
5. Получены экспериментальные данные по оценке вероятности ошибочной декапсуляции в схеме KEM на системе типа Мак-Элиса на D-кодах при использовании построенных вероятностных декодеров.
6. Разработаны рекомендации по выбору параметров эффективных стойких криптосистем типа Мак-Элиса на D-кодах для применения в схеме KEM.

Заключение: Использование указанных результатов позволяет: повысить диверсификацию эффективных постквантовых схем инкапсуляции сеансового ключа; оценить эффективность альтернативных схем KEM; повысить качество анализа стойкости кодовых криптосистем, построенных на модификациях кодов Рида—Маллера.

Результаты внедрялись при выполнении научно-исследовательской работы «Формирование методики и автоматизированных инструментов выбора постквантовых механизмов, основанных

на помехоустойчивом кодировании используемых при обеспечении информационной безопасности сетевого взаимодействия», шифр «Кульминация».

Председатель комиссии:

Директор по научным исследованиям, к.ф.-м.н. Алексеев Е.К.

Члены комиссии:

Старший криптограф-исследователь лаборатории исследований проблем квантово-криптографических технологий Дирекция по научным исследованиям, к.ф.-м.н. Чижов И.В.

Заместитель начальника отдела обеспечения научных исследований Дирекция по научным исследованиям Полтавская И.В.

Федеральное государственное автономное научное учреждение
"Научно-исследовательский институт
"Специализированные вычислительные устройства защиты и автоматика"

344003, г. Ростов-на-Дону, ул. Города Волос, 6 | Тел. (863) 201-28-17, факс (863) 201-28-13, e-mail: info@niisva.org

Акт
о использовании результатов практических решений,
разработанных в диссертационной работе
«Синтез постквантовой схемы инкапсуляции сеансового ключа»
Лелюка Евгения Андреевича

Комиссия в составе: председателя – заместителя директора по научной работе Гуфана К.Ю., членов комиссии: начальника центра системного программирования Селина Р.Н., заведующего лабораторией проектных разработок Короченцева Д.А. составила настоящий акт о том, что результаты диссертационного исследования Лелюка Е.А., а именно:

1) разработанные алгоритмы гарантированного и вероятностного декодирования D-кодов на основе кодов Рида-Маллера для применения в кодовых схемах шифрования;

2) данные по оценке вероятности ошибочной декапсуляции в схеме инкапсуляции сеансового ключа на системе типа Мак-Элиса на D-кодах при использовании построенных вероятностных декодеров;

3) рекомендации по выбору параметров эффективных стойких криптосистем типа Мак-Элиса на D-кодах для применения в схеме инкапсуляции сеансового ключа, использованы в ФГАНУ "Научно-исследовательский институт "Специализированные вычислительные устройства защиты и автоматика" при решении задачи выработки общего сеансового ключа для организации защищенного канала передачи информации.

Использование указанных результатов позволило организовать защищенный канал передачи информации в условиях наличия помех и пассивного наблюдателя с доступом к квантовому компьютеру достаточной мощности. При использовании указанных результатов также удалось уточнить критерии для выбора современных схем защиты информации при решении задач организации защищенного канала передачи информации.

Реализация научных результатов диссертационного исследования Лелюка Е.А. обсуждена на заседании научно-технического совета ФГАНУ "Научно-исследовательский институт "Специализированные вычислительные устройства защиты и автоматика" (протокол № 32 от 19.11.2024 г.).

Председатель комиссии: _____ (подпись) Гуфан К.Ю.

Члены комиссии: _____ (подпись) Селин Р.Н.

_____ (подпись) Короченцев Д.А.

М.П.

«19» ноября 2024 г.

УТВЕРЖДАЮ

Директор Института математики, механики и компьютерных наук им. И.И. Воровича, доктор физико-математических наук, доцент

_____ М.И. Карякин

«24» декабря 2024 г.

АКТ

внедрения результатов в учебный процесс кафедры алгебры и дискретной математики Института математики, механики и компьютерных наук им. И.И. Воровича

ФГАОУ ВО «Южный федеральный университет»

Настоящим актом подтверждается, что результаты диссертационной работы Лелюка Евгения Андреевича на тему «Синтез постквантовой схемы инкапсуляции сеансового ключа» внедрены и используются в учебном процессе кафедры алгебры и дискретной математики ИММиКН ЮФУ:

- результаты теоретических и практических исследований стойкости кодовых криптосистем типа Мак-Элиса, разработанных в диссертационной работе, используются в курсе лекций по дисциплине «Криптография» для студентов направления подготовки 01.03.02 «Прикладная математики и информатика».

Зав. Кафедры АДМ,
д.т.н., ст. н.с. Штейнберг Б.Я.

Доцент кафедры АДМ,
к.т.н. Косолапов Ю.В.

Приложение Б

Свидетельство о регистрации программы для ЭВМ

Далее приводится свидетельство о государственное регистрации программы для ЭВМ, а именно, реализации алгоритма IsDCodeStrong под названием «Программное средство определения множества сильных и слабых ключей криптосистемы на D-кодах на основе кодов Рида–Маллера».

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024691736

Программное средство определения множества сильных и слабых ключей криптосистемы на D-кодах на основе кодов Рида-Маллера

Правообладатель: ***Лелюк Евгений Андреевич (RU)***

Автор(ы): ***Лелюк Евгений Андреевич (RU)***

Заявка № **2024690988**

Дата поступления **14 декабря 2024 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ ***24 декабря 2024 г.***

Руководитель Федеральной службы

по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат 0692e7c1a6300bf54f240f670bca2026

Владелец **Зубов Юрий Сергеевич**

Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов