МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

# ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

*Научный журнал*

**2025** **№ 67**

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

**УЧРЕДИТЕЛЬ**
**Томский государственный университет**



**Адрес редакции и издателя:** 634050, г. Томск, пр. Ленина, 36
**E-mail:** pank@mail.tsu.ru

*В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.*

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Редактор-переводчик *Т. В. Бутузова*
Верстка *И. А. Панкратовой*

---

Подписано к печати 11.03.2025. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 15. Тираж 300 экз.
Заказ № 6247. Цена свободная. Дата выхода в свет 13.03.2025.

---

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

# СОДЕРЖАНИЕ

# CONTENTS

# ПАМЯТИ ЛЬВА АБРАМОВИЧА ШОЛОМОВА

31 декабря 2024 г. ушёл из жизни Лев Абрамович Шоломов, выдающийся специалист в области дискретной математики, доктор физико-математических наук, профессор, действительный член РАЕН, главный научный сотрудник Института системного анализа РАН, профессор кафедры нелинейных динамических систем и процессов управления факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова.

Лев Абрамович родился 5 марта 1936 г. в Москве, в 1954 г. окончил школу и поступил в Московский автодорожный институт, после окончания которого работал конструктором на заводе ЗИЛ.

В 1961–1965 гг. обучался на заочном отделении механико-математического факультета МГУ им. М. В. Ломоносова. С 1962 г. перешёл на работу в Институт проблем управления Академии наук.

В 1969–1972 гг. обучался в очной аспирантуре Института прикладной математики АН СССР и в 1972 г. защитил кандидатскую диссертацию под руководством О. Б. Лупанова на тему «Об информационной сложности задач, связанных с минимальной реализацией булевых функций схемами».

С 1972 г. работал в Институте проблем управления, который в 1976 г. переименован в Институт системного анализа (ИСА).

В 1989 г. Лев Абрамович защитил докторскую диссертацию на тему «Построение и анализ сложности механизмов выбора логическими методами».

В 1991 г. получил учёное звание профессора, с 2002 г. действительный член РАЕН.

Основные области научных исследований — синтез и сложность управляющих систем, теория недоопределённой информации и математическая теория выбора. Им получен ряд результатов по исследованию класса сложно реализуемых частично определённых функций. Один из первых результатов — асимптотическая нижняя оценка

сложности частично определённых булевых функций — вошёл в учебники по дискретной математике. В дальнейшем им разработаны оригинальные логические методы построения и анализа моделей последовательного и многокритериального выбора, а также получены глубокие результаты по теоретико-информационным характеристикам и кодированию недоопределённых данных.

Лев Абрамович долгое время преподавал в Национальном исследовательском технологическом университете «МИСиС», а с 2001 г. — на факультете ВМК МГУ им. М. В. Ломоносова.

Лев Абрамович — автор более 100 научных работ, в числе которых две монографии и учебные пособия. Он являлся членом Научно-методического совета по информатике Минобразования России, членом специализированного учёного совета ИСА РАН, членом редакционной коллегии журнала «Прикладная дискретная математика».

Лев Абрамович принимал участие в работе многих научных конференций и семинаров, в том числе школы-семинара SIBECRYPT. Он был простым в общении и внимательным собеседником, с интересом слушал выступления других и всегда старался понять суть проблем.

Поражал своей физической формой: в уже преклонном возрасте обгонял молодёжь на прогулке по Красноярским столбам, азартно преодолевал крутые склоны.

Светлая память о Льве Абрамовиче Шоломове навсегда останется в наших сердцах.

Редколлегия

Основные работы (mathnet.ru):

1. *Шоломов Л. А.* Основы теории дискретных логических и вычислительных устройств: учеб. пособие для студентов вузов. М.: Наука, 1980.
2. *Шоломов Л. А.* Логические методы исследования дискретных моделей выбора / Теория и методы системного анализа. М.: Наука, 1989.
3. *Шоломов Л. А.* О реализации недоопределенных булевых функций схемами из функциональных элементов // Проблемы кибернетики. 1969. № 21. С. 215–226.
4. *Sholomov L. A.* Explicit form of neutral social decision rules for basic rationlity conditions // Mathematical Social Sciences. 2000. No. 39:1. P. 81–107.
5. *Шоломов Л. А.* Элементы теории недоопределенной информации // Прикладная дискретная математика. Приложение. 2009. № 2. С. 18–42.

# МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 621.391.7 DOI 10.17223/20710410/67/1

## О ПАРАМЕТРАХ СИСТЕМЫ ШИФРОВАНИЯ ТИПА МАК-ЭЛИСА НА $D$-КОДАХ, ОСНОВАННЫХ НА ДВОИЧНЫХ КОДАХ РИДА — МАЛЛЕРА

Ю. В. Косолапов, Е. А. Лелюк

*Южный федеральный университет, г. Ростов-на-Дону, Россия*

**E-mail:** yvkosolapov@sfedu.ru, lelukevgeniy@mail.ru

Исследуются характеристики кодовой системы шифрования типа Мак-Элиса на специальной сумме тензорных произведений базовых кодов, называемой $D$-кодом. В качестве базовых выбраны двоичные коды Рида — Маллера. Ранее были найдены условия на эти $D$-коды, при выполнении которых соответствующая система шифрования устойчива к известным структурным атакам на основе произведения Шура — Адамара. Однако при использовании декодера, работающего в пределах половины кодового расстояния, система типа Мак-Элиса на $D$-кодах обеспечивает стойкость, сравнимую со стойкостью классической системы шифрования Мак-Элиса на кодах Гоппы, при существенно большем размере ключа. В настоящей работе построены два вероятностных декодера для $D$-кодов. В случае использования этих декодеров найдены параметры некоторых $D$-кодов, которые при меньшем размере ключа, чем в классической системе, обеспечивают сравнимую стойкость к атаке декодированием по информационным совокупностям. Однако наличие непренебрежимой вероятности ошибочного декодирования пока ограничивает область применения системы шифрования на $D$-кодах эфемерными механизмами защищённой передачи сеансового ключа (IND-CPA KEM).

**Ключевые слова:** *$D$-коды, схема Мак-Элиса, механизм защищённой передачи сеансового ключа.*

## ON THE PARAMETERS OF A McELIECE-TYPE CRYPTOSYSTEM ON $D$-CODES BASED ON BINARY REED — MULLER CODES

Yu. V. Kosolapov, E. A. Lelyuk

*Southern Federal University, Rostov-on-Don, Russia*

The characteristics of a McEliece-type code cryptosystem on a special sum of tensor products of base codes, called $D$-code, are investigated. Binary Reed — Muller codes were chosen as the base codes. Previously, conditions were found for these $D$-codes, under which the corresponding cryptosystem is resistant to known structural attacks based on the Schur — Hadamard product. However, when using a decoder operating within half the code distance, a McEliece-type system on $D$-codes provides security comparable to the strength of the classical McEliece cryptosystem on Goppa codes, with a significantly larger key size. In this paper, two probabilistic decoders for $D$-codes are constructed. In the case of using these decoders, parameters

of some $D$-codes have been found that provide comparable resistance to information set decoding type attacks, while having a smaller key size than in the classical system. However, the presence of a non-negligible decoding failure rate currently limits the scope of application of the $D$-code cryptosystem to ephemeral session key encapsulation mechanisms (IND-CPA KEM).

**Keywords:** *D-codes, McEliece scheme, key encapsulation mechanism.*

## Введение

Класс кодов Гоппы в настоящее время рассматривается как основной для системы шифрования типа Мак-Элиса. Это обусловлено тем, что большинство попыток заменить коды Гоппы другими кодами с целью уменьшить размер открытого ключа не увенчалось успехом из-за их слабой стойкости к структурным атакам (атакам восстановления ключа) [1]. Примечательно, что в 1978 г. Р. Мак-Элис предложил новую концепцию асимметричной системы шифрования именно на кодах Гоппы [2]. Далее эту систему будем называть оригинальной, или Original McEliece. В конкурсе NIST [4] система шифрования Мак-Элиса на кодах Гоппы носит название Classic McEliece [3]. Отличие Original McEliece от Classic McEliece заключается в правиле шифрования: в Classic McEliece шифрование выполняется по схеме, предложенной Х. Нидеррайтером в [5], так как это позволяет сократить размер открытого ключа. Заметим, что фактически схема Original McEliece являлась основой проекта NTS-KEM [6], который позже был объединён с проектом Classic McEliece по причине использования одного класса кодов Гоппы.

Отсутствие в настоящий момент эффективных структурных атак для схемы Мак-Элиса на кодах Гоппы, как представляется, только мотивирует исследователей к поиску новых способов её раскалывания [7–9]. В этой связи актуальным является не только построение принципиально новых кодовых систем шифрования, что делается в [10–12], но и поиск новых кодов для схемы Мак-Элиса. Необходимыми условиями для таких кодов являются, во-первых, стойкость к известным структурным атакам и, во-вторых, наличие эффективного декодера.

Результаты исследования стойкости системы типа Мак-Элиса на $D$-кодах на основе двоичных кодов Рида — Маллера [13, 14] и наличие эффективного декодера [15] позволяют рассматривать $D$-коды в качестве такой альтернативы. Однако стоит отметить, что высокая стойкость к атакам на шифртекст в случае использования декодеров $D$-кодов, работающих в пределах половины кодового расстояния, достигается только при больших размерах открытого ключа [14]. Это связано с небольшим кодовым расстоянием $D$-кодов. Повысить стойкость к атакам на шифртекст можно путём увеличения веса добавляемого при шифровании вектора ошибок. Это, в свою очередь, позволяет для заданного уровня стойкости уменьшить размер ключа, используя код с меньшими значениями длины и размерности. В частных случаях рассматриваемые в [14] $D$-коды являются подкодами кодов Рида — Маллера. Так как для двоичных кодов Рида — Маллера известны декодеры, которые позволяют исправлять ошибки веса, существенно большего половины кодового расстояния (см. обзор [16]), то при использовании таких декодеров существует потенциальная возможность сокращения размера ключа системы на $D$-кодах при сохранении её высокой стойкости к атакам восстановления ключа и атакам на шифртекст.

В [14] указаны параметры $D$-кодов, для которых при использовании декодеров, работающих за пределами половины кодового расстояния, может быть обеспечена

стойкость соответствующей системы шифрования типа Мак-Элиса не ниже стойкости систем на кодах Гоппы при сопоставимых размерах ключа. Стоит отметить, что сопоставимые размеры ключа достигаются при большей длине $D$-кодов относительно кодов Гоппы, то есть при уменьшении информационной скорости. Большая длина $D$-кодов, в свою очередь, ограничивает применение известных декодеров кодов Рида — Маллера для непосредственного декодирования $D$-кодов из-за их высокой вычислительной сложности и сложности по требуемой памяти. Применение декодера из [15] для $D$-кодов также ограничено высокой сложностью по памяти. Поэтому вопрос эффективного декодирования $D$-кодов большой длины остаётся актуальным.

В настоящей работе для эффективного декодирования $D$-кодов используется блочная структура кодовых слов: каждый блок декодируется декодером кода Рида — Маллера меньшей длины, а затем в случае неправильного декодирования выполняется блочное декодирование по информационным совокупностям. Такой подход позволяет существенно ускорить расшифрование, несмотря на «дорогую» операцию блочного декодирования по информационным совокупностям, а использование для кодов Рида — Маллера декодера из [17], который исправляет ошибки веса, существенно большего половины кодового расстояния, позволяет увеличить вес добавляемой ошибки и сократить размер открытого ключа при обеспечении высокой стойкости.

Работа, кроме введения и заключения, состоит из пяти пунктов. В п. 1 формально описаны схема асимметричного шифрования и механизм защищённой передачи сеансового ключа KEM на основе этой схемы. В п. 2 в компактном виде приводится описание системы шифрования типа Мак-Элиса в случае произвольного линейного кода, а также механизм защищённой передачи сеансового ключа KEM на основе системы типа Мак-Элиса. В п. 3 рассмотрено построение декодеров для $D$-кодов, применимых в схеме типа Мак-Элиса. В п. 4 содержатся результаты исследования, цель которого — поиск параметров $D$-кодов для системы типа Мак-Элиса, обеспечивающих сопоставимые с системами шифрования на кодах Гоппы стойкость и размер открытого ключа при использовании построенных декодеров. В п. 5 приводятся характеристики механизмов KEM на основе системы шифрования на $D$-кодах.

## 1. Схема асимметричного шифрования и механизм KEM

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 $\mathsf{Alg}$ — соответственно $a \leftarrow \mathsf{Alg}$; обращение к алгоритму $\mathsf{Alg}$, не принимающему параметры, — $\mathsf{Alg}(\cdot)$.

Схема асимметричной системы шифрования (PKE, Public Key Encryption) для заданного множества сообщений $\mathcal{M}$ может быть описана тройкой $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$. Здесь

1) $\mathsf{Gen}^{\mathsf{PKE}}$ — полиномиальный вероятностный алгоритм генерации ключей $pk$ и $sk$ $(pk, sk) \leftarrow \mathsf{Gen}^{\mathsf{PKE}}(\cdot)$: закрытого $sk$ и соответствующего ему открытого $pk$;
2) $\mathsf{Enc}$ — полиномиальный вероятностный алгоритм шифрования, преобразующий произвольное сообщение $\mathbf{m} \in \mathcal{M}$ в шифртекст $\mathbf{c}$: $\mathbf{c} \leftarrow \mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$, где $\mathbf{r}$ — случайное значение, которое может вырабатываться при шифровании;
3) $\mathsf{Dec}$ — 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 (One-Way-стойкостью) схемы $\mathsf{PKE}$ 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 определён как отрицательный двоичный логарифм вероятности успеха наилучшей из таких атак.

Асимметричные шифрсистемы обычно не используются для защиты непосредственно пользовательских данных, но при этом являются основой многих криптографических протоколов, в частности механизмов защищённой передачи сеансового ключа (KEM, Key Encapsulation Mechanism). 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.

Механизм защищённой передачи для фиксированного множества сеансовых ключей $\mathcal{K}$ и схемы $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ может быть описан тройкой алгоритмов $\mathsf{KEM} = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$. Здесь

1) $\mathsf{Gen}^{\mathsf{KEM}}$ — полиномиальный вероятностный алгоритм генерации открытого $pk'$ и закрытого $sk'$ ключей (строится на основе $\mathsf{Gen}^{\mathsf{PKE}}$);
2) $\mathsf{Encaps}$ — 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 $\mathsf{Enc}$);
3) $\mathsf{Decaps}$ — полиномиальный алгоритм извлечения (декапсуляции) сеансового ключа $\mathbf{K}$ из $\mathbf{c}$ (строится на основе $\mathsf{Dec}$). Алгоритм $\mathsf{Decaps}(sk', \mathbf{c})$ возвращает $\mathbf{K}$, если $\mathbf{c}$ представляет корректную инкапсуляцию ключа $\mathbf{K}$, в противном случае $\mathsf{Decaps}(sk', \mathbf{c})$ 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).

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

$$\lambda_{\mathsf{OW}} \geqslant \lambda_{\mathsf{KEM}} \geqslant m. \tag{1}$$

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

$$1 - \delta = 1 - \Pr\left[\perp \leftarrow \mathsf{Decaps}(sk', \mathbf{c}) : (pk', sk') \leftarrow \mathsf{Gen}^{\mathsf{KEM}},\ (\mathbf{K}, \mathbf{c}) \leftarrow \mathsf{Encaps}(pk')\right] \tag{2}$$

события, заключающегося в успешной декапсуляции сеансового ключа, назовём *отказоустойчивостью* KEM. Эта вероятность может влиять как на удобство использования механизма, так и на его стойкость. Неудобства, связанные с ненулевым $\delta$, заключаются в следующем. При многоразовом использовании в случае ошибки декапсуляции отправителю сеансового ключа необходимо повторно использовать алгоритм $\mathsf{Encaps}$, а получателю — алгоритм $\mathsf{Decaps}$; 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 защищённой передачи с многоразовым и одноразовым использованием проявляются во влиянии значения $\delta$ на стойкость этих механизмов. Обычно выделяют два типа стойкости механизма KEM: стойкие к

атаке на основе подобранного открытого текста (IND-CPA, INDistinguishability under Chosen Plaintext Attack) и стойкие к атаке на основе подобранного шифртекста (IND-CCA1/2, INDistinguishability under Chosen Ciphertext Attack).

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 [18]. Например, для механизма KEM BIKE [19] и его некоторых вариаций большое значение $\delta$ позволяет провести атаку с опорой на реакцию, нацеленную на восстановление закрытого ключа [20, 21]. Поэтому для многоразового использования ключей от KEM требуется обеспечение IND-CCA-стойкости уровня $\lambda_{\mathsf{KEM}}$, 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 либо неявного отказа при декапсуляции в алгоритме $\mathsf{Decaps}$ [22], хотя в механизмах из конкурса NIST используется неявный отказ.

В случае эфемерной защищённой передачи информация об ошибке декапсуляции представляется малопригодной для атакующего. Поэтому для таких механизмов достаточно IND-CPA-стойкости [23] и, согласно [24], вероятность ошибочного декодирования влияет только на отказоустойчивость информационной системы. Верхняя граница значения $\delta$ в (2) определяется тем, насколько часто в информационной системе допускается повторное выполнение KEM парой участников информационного взаимодействия. Верхнюю границу $\delta$ можно определить в виде $10^{-\gamma}$, где $\gamma \in \mathbb{N}$. Далее параметр $\gamma$ будем называть *уровнем отказоустойчивости* информационной системы, в которой используется механизм KEM, или кратко — уровнем отказоустойчивости KEM. В [24] в качестве допустимых значений $\gamma$ рассматриваются значения из множества $\{5, 6, 7\}$, при этом отмечается, что $\gamma = 5$ считается «золотым стандартом» отказоустойчивости информационных систем. В [25] в качестве допустимого рассматривается значение $\gamma = 9$. Отметим также, что в эфемерных механизмах KEM может использоваться явный отказ вместо неявного, что позволяет сократить размер закрытого ключа $sk'$ и время декапсуляции [24].

Таким образом, условие на вероятность $\delta$ в (2) для одноразового и многоразового использования ключей можно сформулировать так:

$$\delta \leqslant \begin{cases} 2^{-\lambda_{\mathsf{KEM}}}, & \text{многоразовое использование,} \\ 10^{-\gamma}, & \text{одноразовое использование,} \end{cases} \tag{3}$$

где $\lambda_{\mathsf{KEM}}$ — требуемый уровень IND-CCA-стойкости механизма KEM; $\gamma$ — требуемый уровень отказоустойчивости IND-CPA-стойкого механизма KEM.

## 2. Система шифрования типа Мак-Элиса и механизм KEM на её основе

Введём необходимые обозначения. Пусть $\mathbb{F}_q$ — поле Галуа мощности $q$. Линейным $[n, k, d]_q$-кодом $C$ будем называть подпространство пространства $\mathbb{F}_q^n$ размерности $k$ с минимальным кодовым расстоянием $d = \min\{\mathrm{wt}(\mathbf{c}) : \mathbf{c} \in C \setminus \{\mathbf{0}\}\}$, где $\mathrm{wt}(\mathbf{c})$ — вес Хэмминга (далее — просто вес) вектора $\mathbf{c}$; $\mathbf{0}$ — нулевой вектор из $\mathbb{F}_q^n$. В ряде случаев будем опускать $d$ и писать $[n, k]_q$-код, если контекст не требует явного указания этого параметра. 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

Decoder в случае, когда, например, входной вектор не может быть однозначно декодирован. Алгоритм, реализующий декодирование, также будем обозначать 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 $p_1, p_2, \dots$ будем отделять точкой с запятой и писать $\mathsf{Decoder}(\mathbf{z}; p_1, p_2, \dots)$.

Пусть $C$ — линейный $[n, k, d]_q$-код с фиксированной порождающей матрицей $G$, для которого известен эффективный декодер 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 $\mathsf{McE}(C) = (\mathsf{Gen}^{\mathsf{McE}(C)}, \mathsf{Enc}, \mathsf{Dec})$; соответствующие алгоритмы приведены на рис. 1. Отметим, что здесь и далее рассматривается схема Мак-Элиса, когда код $C$ не является секретным, поэтому матрица $G$ является частью алгоритмов $\mathsf{Gen}^{\mathsf{McE}(C)}$ и Dec. Так как в ряде случаев декодер кода $C$ может исправлять векторы ошибок веса больше, чем $\lfloor (d-1)/2 \rfloor$, то для алгоритма $\mathsf{Gen}^{\mathsf{McE}(C)}$ введён 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 $\mathsf{McE}(C)$ будем называть схемой с вероятностным шифрованием, во втором — с детерминированным шифрованием.

| | $\mathsf{Gen}^{\mathsf{McE}(C)}(t)$ : | $\mathsf{Enc}(pk, \mathbf{m}; \mathbf{e})$ : | $\mathsf{Dec}(sk, \mathbf{c})$ : |
|---|---|---|---|
| 1 | $S \leftarrow \mathrm{GL}_k(\mathbb{F}_q)$ | $\mathbf{c} = m\tilde{G} + e$ | $\mathbf{x} \leftarrow \mathsf{Decoder}(cP^{-1})$ |
| 2 | $P \leftarrow \mathcal{P}_n$ | ***вернуть*** $\mathbf{c}$ | *если* $\mathbf{x} \neq \perp$ |
| 3 | $\tilde{G} = SGP$ | | $\mathbf{m}' = \mathbf{x}\, G^{-1} S^{-1}$ |
| 4 | $pk = (\tilde{G}, t)$ | | ***вернуть*** $\mathbf{m}'$ |
| 5 | $sk = (S, P)$ | | *иначе* |
| 6 | ***вернуть*** $(pk, sk)$ | | ***вернуть*** $\perp$ |

Рис. 1. Схема асимметричного шифрования $\mathsf{McE}(C) = (\mathsf{Gen}^{\mathsf{McE}(C)}, \mathsf{Enc}, \mathsf{Dec})$

Важными характеристиками систем типа Мак-Элиса являются уровень OW-стойкости, размер открытого ключа (число байт для хранения матрицы $\tilde{G}$) и время расшифрования. В случае, когда для системы шифрования типа Мак-Элиса неизвестны эффективные атаки восстановления ключа, уровень стойкости системы принято оценивать с помощью вероятности успешного декодирования по информационным совокупностям (далее — ДИС или ISD от англ. Information Set Decoding). Этот подход к декодированию линейных кодов предложен в работе [26] и в настоящее время считается наиболее эффективной атакой на шифртекст для систем шифрования типа Мак-Элиса. Несмотря на то, что известен ряд усовершенствований этого алгоритма (см., например, [27, 28]), в настоящей работе под уровнем OW-стойкости $\lambda_{\mathsf{OW}}$ системы шифрования типа Мак-Элиса на $[n, k, d]_q$-коде понимается отрицательный двоичный логарифм вероятности успеха классического алгоритма ДИС, то есть вероятности выбора (по схеме с возвращением) информационной совокупности, свободной от ошибок:

$$\lambda_{\mathsf{OW}} = -\log_2 \left( \frac{\mathrm{C}_{n-t}^k}{\mathrm{C}_n^k} \right) = \log_2 \left( \frac{\mathrm{C}_n^k}{\mathrm{C}_{n-t}^k} \right). \tag{4}$$

В табл. 1 приведены характеристики 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 $\mathsf{Enc}$ на рис. 1; $\lambda_{\mathsf{OW}}$ — уровень стойкости, вычисленный в соответствии с (4); размер ключа в мегабайтах (Мбайт) для систем Original McEliece и Classic McEliece; $T$ — время расшифрования (в миллисекундах) информационного сообщения при использовании декодера из [3] (измерение времени выполнения операций во всех экспериментах производилось на компьютере с процессором Intel Core i7-4771 3,50 ГГц и оперативной памятью 32 Гбайт). Стоит отметить, что названия вариантов системы шифрования Original McEliece, а также параметры $[n, k]$ и $t$ выбраны такими же, как и для системы шифрования Classic McEliece из работы [3].

Значения $\lambda_{\mathsf{OW}}$ из табл. 1 в настоящей работе рассматриваются как целевые при разработке схемы шифрования.

Таблица 1

**Характеристики системы Мак-Элиса на кодах Гоппы**

| Система шифрования | $[n, k]$ | $t$ | $\lambda_{\mathsf{OW}}$ | Размер ключа, Original McEliece Мбайт | Размер ключа, Classic McEliece Мбайт | $T$ |
|---|---|---|---|---|---|---|
| mceliece348864 | $[3488, 2720]$ | 64 | 142,8 | 1,13 | 0,25 | 20 |
| mceliece460896 | $[4608, 3360]$ | 96 | 185,92 | 1,85 | 0,5 | 51 |
| mceliece6688128 | $[6688, 5024]$ | 128 | 262,28 | 4,01 | 1 | 98 |
| mceliece6960119 | $[6960, 5413]$ | 119 | 265,6 | 4,49 | 1 | 95 |
| mceliece8192128 | $[8192, 6528]$ | 128 | 302,21 | 6,38 | 1,29 | 120 |

На основе асимметричной системы шифрования $\mathsf{PKE} = (\mathsf{Gen}^{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ 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 ($\mathsf{FO}$) [18]. В частности, для всех $\lambda_{\mathsf{OW}}$ из табл. 1 система шифрования Classic McEliece обеспечивает IND-CCA-стойкость протокола KEM на основании преобразования Фудзисаки — Окамото [3]. Тройка алгоритмов преобразования $\mathsf{FO}^{\not\perp}$ [18] механизма KEM с неявным отказом (сообщение $\perp$ не возвращается) показана на рис. 2, 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.

| | $\mathsf{Gen}^{\mathsf{KEM}}(t)$ : | $\mathsf{Encaps}(pk')$ : | $\mathsf{Decaps}(sk', \mathbf{c})$ : |
|---|---|---|---|
| 1 | $s \leftarrow \mathbb{F}_q^k$ | $\mathbf{r} \leftarrow \mathbb{F}_q^k$ | $\mathbf{r}' \leftarrow \mathsf{Dec}(sk, \mathbf{c})$ |
| 2 | $(pk, sk) \leftarrow \mathsf{Gen}^{\mathsf{McE}(C)}(t)$ | $\mathbf{c} \leftarrow \mathsf{Enc}(pk, \mathbf{r}; \mathsf{G}(\mathbf{r}))$ | *если* $\mathbf{r}' \neq \perp$ |
| 3 | $pk' = pk$ | $\mathbf{K} \leftarrow \mathsf{H}(\mathbf{r} \parallel \mathbf{c})$ | $\mathbf{c}' \leftarrow \mathsf{Enc}(pk, \mathbf{r}'; \mathsf{G}(\mathbf{r}'))$ |
| 4 | $sk' = (sk, s)$ | ***вернуть*** $(\mathbf{K}, \mathbf{c})$ | *если* $\mathbf{c} = \mathbf{c}'$ |
| 5 | ***вернуть*** $(pk', sk')$ | | ***вернуть*** $\mathsf{H}(\mathbf{r}' \parallel \mathbf{c})$ |
| 6 | | | ***вернуть*** $\mathsf{H}(s \parallel \mathbf{c})$ |

Рис. 2. Механизм $\mathsf{KEM} = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$, полученный применением преобразования $\mathsf{FO}^{\not\perp}$ на основе схемы асимметричного шифрования $\mathsf{McE}(C)$ с детерминированным правилом шифрования

## 3. $D$-коды и их эффективное декодирование

В работе [14] предлагается система шифрования типа Мак-Элиса, где кодом $C$ является бинарный $D$-код. Однако в силу небольшого отношения кодового расстояния $D$-кода к длине кода для обеспечения высокого уровня стойкости $\lambda_{\mathsf{OW}}$ при использовании декодеров, работающих в пределах половины кодового расстояния, необходимо использовать открытые ключи большого размера. В частности, в [14] показано, что для обеспечения стойкости, сопоставимой со стойкостью Classic McEliece с набором параметров mceliece348864, необходимо использовать открытый ключ размера 154,85 Мбайт. Большие размеры ключей приводят не только к сопутствующим проблемам их хранения и/или передачи, но и к росту вычислительной сложности декодера и/или росту его сложности по используемой памяти. Например, конструкция мажоритарного декодера, предложенная в [15], представляет собой граф, число узлов которого быстро растёт с ростом длины кода.

В [14] отмечено, что уменьшения размера ключа при сохранении стойкости можно потенциально добиться путём применения декодеров кода Рида — Маллера, позволяющих исправлять многие шаблоны ошибок веса, превышающего половину кодового расстояния. Далее даётся определение $D$-кодов, а также строятся декодеры упомянутого типа для $D$-кодов, построенных на основе двоичных кодов Рида — Маллера.

### 3.1. Коды Рида — Маллера и их декодирование

Среди линейных кодов важным является класс двоичных $[n, k, d]_2$-кодов Рида — Маллера $\mathrm{RM}(r, m)$, где $n = 2^m$; $k = \sum\limits_{i=0}^{r} \mathrm{C}_r^i$; $d = 2^{m-r}$ [16]. В частности, известно, что эти коды достигают ёмкости канала со стираниями и симметричных каналов без памяти [29, 30]. Для кодов Рида — Маллера существуют разные подходы к декодированию [16], которые условно 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 направлены на исправление векторов ошибок большого веса, при этом они обладают ненулевой вероятностью ошибочного декодирования (DFR, Decoding Failure Rate). Примерами гарантированных декодеров являются классический декодер Рида [31] и мажоритарный декодер Мэсси [32, 33]. Одним из первых вероятностных декодеров является декодер Сидельникова — Першакова [34]; одним из последних — декодер Йе — Аббе, который, согласно [17], обладает лучшей корректирующей способностью по сравнению с декодером Сидельникова — Першакова. Отметим, что вероятностные декодеры, как правило, вычислительно сложнее, чем эффективные гарантированные.

### 3.2. $D$-коды

Подробное описание $D$-кодов, методы и алгоритмы их декодирования, а также их криптографические свойства представлены в работах [13–15, 35]. $D$-код — это специальная сумма тензорных произведений кодов, которые назовём *базовыми*. Напомним, что тензорным произведением кодов $C_1$ и $C_2$ является код $C_1 \otimes C_2$, порождающая матрица $G$ которого может быть представлена как тензорное произведение порождающих матриц кодов $C_1$ и $C_2$. Под тензорным произведением $A \otimes B$ матриц $A = (a_{i,j})$ и $B$ размера $k_1 \times n_1$ и $k_2 \times n_2$ соответственно будем понимать $(k_1 k_2 \times n_1 n_2)$-матрицу вида

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

$D$-коды, где базовыми являются коды Рида — Маллера, определяются следующим образом. Пусть $m_1, m_2 \in \mathbb{N}$; $(k_i)_{i=1}^s$ — возрастающая последовательность, $k_1 \geqslant 0$, $k_s \leqslant m_1$; $(\ell_i)_{i=1}^s$ — убывающая последовательность, $\ell_1 \leqslant m_1$, $\ell_s \geqslant 0$. Тогда $D$-код $C$ длины $n = 2^{m_1+m_2}$ и размерности $k$ представим в виде

$$C = \sum_{i=1}^{s} \left(\mathrm{RM}(k_i, m_1) \otimes \mathrm{RM}(\ell_i, m_2)\right), \tag{5}$$

где под суммой кодов понимается сумма подпространств. Далее рассмотрим только $D$-коды на основе кодов Рида — Маллера (коды вида (5)).

### 3.3. Декодирование $D$-кодов

Рассмотрим задачу декодирования $D$-кода $C$ с характеристиками $[n, k, d]_2$. Пусть $G$ — порождающая матрица кода $C$, $\mathbf{m} \in \mathbb{F}_2^k$, $\mathbf{m}G + \mathbf{e} = \mathbf{z}$, $\mathrm{wt}(\mathbf{e}) = t \in \mathbb{N}$. Вопросам построения и эффективной реализации одного декодера для $D$-кодов посвящены работы [15, 35]. В основе декодирования лежит мажоритарный подход, предложенный Дж. Мэсси [32], который, применительно к $D$-кодам, позволяет исправлять все шаблоны ошибок веса не более половины кодового расстояния: $t \leqslant \lfloor (d-1)/2 \rfloor$. Поэтому этот декодер относится к классу гарантированных. Для удобства гарантированный декодер, описанный в [15], обозначим $\mathsf{GraphDecoder}$.

Особая структура $D$-кодов и наличие вероятностных декодеров для кодов Рида — Маллера позволяют построить вероятностный декодер и для $D$-кодов. Далее строятся два вероятностных декодера: с использованием декодирования по информационным совокупностям и на основе декодирования кодов-произведений, в которых кодовое слово может быть представлено в виде матрицы. Декодер $D$-кода, когда не важен класс декодера (гарантированный или вероятностный), будем обозначать $\mathsf{DDecoder}$.

Для описания вероятностных декодеров 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 $\mathsf{DDecoder}$ зашумлённого 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 0 в противном случае. Алгоритм $\mathsf{ExitCond}$ используется как параметр алгоритма $\mathsf{DDecoder}$ 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.), четвёртым параметром алгоритма $\mathsf{ExitCond}$ является преобразование $\mathsf{op}_1$ вектора $\mathbf{m}'$ перед проверкой. Аналогичное преобразование, соответствующее параметру $\mathsf{op}_2$, может быть применено перед проверкой к вектору $\mathbf{z}$. Примеры $\mathsf{op}_1$ и $\mathsf{op}_2$ содержатся в п. 4.2.

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. 3 приведены примеры двух алгоритмов проверки правильности информационного вектора: на основе только веса ошибки и на основе значения инъективного отображения $\mathsf{G}$.

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

Рис. 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 $\mathsf{G}$

Введём обозначение для вероятностей $\xi_{\mathrm{G}}$, $\xi_{\mathrm{wt}}$ ложно положительной проверки правильности информационного вектора для алгоритмов $\mathsf{ExitCond}_{\mathsf{G}}$, $\mathsf{ExitCond}_{\mathsf{wt}}$:

$$\xi_{\mathrm{A}} = \Pr\left[\mathsf{ExitCond}_{\mathsf{A}}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1 \,|\, \mathbf{m}' \neq \mathbf{m}\right], \quad \mathsf{A} \in \{\mathsf{wt}, \mathsf{G}\}.$$

Стоит отметить, что во всех проведённых 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 $\mathsf{G}$ даёт возможность оценить эту вероятность. Действительно, $\mathsf{ExitCond}_{\mathsf{G}}(G, \mathbf{m}', \mathbf{z}, \mathsf{op}_1, \mathsf{op}_2) = 1$ при $\mathbf{m}' \neq \mathbf{m}$ означает, что

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

Отсюда получаем, что $\mathsf{G}(\mathsf{op}_1(\mathbf{m}')) - \mathsf{G}(\mathsf{op}_1(\mathbf{m})) \in C$. Если при $n - k > k$ отображение $\mathsf{G}$ 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 $\mathsf{G}$, для которого вероятностью $\xi_{\mathsf{G}}$ можно пренебречь.

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 п. 1, важную роль играет значение вероятности ошибочной декапсуляции $\delta$ (см. (2)). Поэтому при выборе конкретного алгоритма проверки следует исходить из допустимой вероятности этой ошибки. Таким условием для механизмов KEM является условие (3): для IND-CCA-стойкости уровня $\lambda_{\mathsf{KEM}}$ механизма KEM необходимо, чтобы $\xi_{\mathsf{A}} \leqslant 2^{-\lambda_{\mathsf{KEM}}}$, а для уровня отказоустойчивости $\gamma$ механизма KEM с IND-CPA-стойкостью необходимо выполнение неравенства $\xi_{\mathsf{A}} \leqslant 10^{-\gamma}$. Далее будем полагать, что алгоритм проверки выбирается, исходя из этих условий. Отметим, что проверка правильности информационного вектора может проводиться также 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). 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 на основе декодирования базового кода и ДИС*

Пусть $n_i = 2^{m_i}$, $i = 1, 2$; $C$ — код вида (5). Из представления (5) следует, что

$$C \subseteq \mathbb{F}_2^{n_1} \otimes \mathrm{RM}(\ell_1, m_2) = \underbrace{\mathrm{RM}(\ell_1, m_2) \times \cdots \times \mathrm{RM}(\ell_1, m_2)}_{n_1}, \tag{6}$$

где $A \times B$ — внешняя прямая сумма подпространств $A$ и $B$. Следовательно, кодовое слово кода $C$ представляет собой конкатенацию кодовых слов кода $\mathrm{RM}(\ell_1, m_2)$. Поэтому декодирование кода $C$ может заключаться в применении некоторого известного декодера $\mathsf{DecRow} : \{0,1\}^{2^{m_2}} \to \mathrm{RM}(\ell_1, m_2) \cup \{\perp\}$ кода $\mathrm{RM}(\ell_1, m_2)$ к каждому из $n_1$ блоков длины $n_2$ декодируемого вектора $\mathbf{z} = \mathbf{z}_1 \parallel \cdots \parallel \mathbf{z}_{n_1}$. Такой способ декодирования назовём $\mathsf{BlockDecoder}_{n_1,n_2}$:

$$\mathsf{BlockDecoder}_{n_1,n_2}(\mathbf{z}, \mathsf{DecRow}) = \mathsf{DecRow}(\mathbf{z}_1) \parallel \cdots \parallel \mathsf{DecRow}(\mathbf{z}_{n_1}).$$

Пара нижних индексов $n_1, n_2$ в названии декодера означает, что перед применением декодера $\mathsf{DecRow}$ выполняется разбиение входного вектора $\mathbf{z}$ на $n_1$ блоков длиной $n_2$ каждый.

В качестве $\mathsf{DecRow}$ в $\mathsf{BlockDecoder}_{n_1,n_2}$ могут использоваться как гарантированные, так и вероятностные алгоритмы декодирования, например декодеры Рида, Мэсси, Сидельникова — Першакова и Йе — Аббе. При этом в обоих случаях алгоритм $\mathsf{BlockDecoder}_{n_1,n_2}$ может вернуть неправильный результат, то есть некоторое количество блоков $\mathbf{z}_i$ зашумлённого кодового вектора $\mathbf{z}$ будет декодировано неправильно. Если блок был декодирован правильно, то назовём его *хорошим*, в противном случае — *плохим*. Отметим, что в худшем случае после работы алгоритма $\mathsf{BlockDecoder}_{n_1,n_2}$ неизвестно, какие блоки хорошие, а какие — плохие.

Чтобы повысить вероятность правильного декодирования, целесообразно после блочного декодирования воспользоваться декодированием по информационным совокупностям. Такой подход — блочное декодирование и применение ДИС — использован в работе [14] при разработке комбинированной атаки на шифртекст. Здесь этот подход использован для декодирования и представлен в алгоритме 1. Напомним, что информационной совокупностью линейного $[n, k]_q$-кода называется такое множество $\tau \subset \{1, \ldots, n\}$ мощности $k$, что столбцы любой порождающей матрицы кода с номерами из $\tau$ линейно независимы. Обычно при декодировании по информационным совокупностям (см., например, оригинальную работу [26]) случайным образом выполняется поиск такой информационной совокупности $\tau$, чтобы номера ненулевых координат вектора ошибки не принадлежали $\tau$. В [14] и алгоритме 1 выполняется поиск номеров хороших блоков, на основе которых может быть построена информационная совокупность.

Для описания алгоритма декодирования понадобится блочное представление порождающей матрицы $D$-кода. Из (5) вытекает, что любая порождающая $(k \times n)$-матрица $G$ этого кода представима в виде

$$G = [G_1 | G_2 | \ldots | G_{n_1}], \tag{7}$$

где строки $(k \times n_2)$-матрицы (блока) $G_i$ — кодовые слова кода $\mathrm{RM}(\ell_1, m_2)$, $i \in [\![1, n_1]\!]$. Здесь и далее запись $[\![a, b]\!]$ означает диапазон целых чисел от $a$ до $b$ включительно. Если $M$ — матрица, состоящая из $r$ столбцов, $\omega \subseteq [\![1, r]\!]$, то матрицу, состоящую из столбцов с номерами из $\omega$, будем обозначать $\Pi_\omega(M)$. Через $\tau_0$ обозначим заранее

известную фиксированную информационную совокупность кода $C$, $M_0 = \Pi_{\tau_0}(G)$. Параметрами алгоритма 1, помимо декодируемого слова $\mathbf{z}$, декодера $\mathsf{DecRow}$, являются $K$ — количество выбираемых блоков для применения ДИС, $I_{\max}$ — максимальное количество итераций ДИС (под одной итерацией понимаются шаги с 10 по 18) и $\mathsf{ExitCond}$ — условие раннего выхода из ДИС (примеры алгоритма $\mathsf{ExitCond}$ приведены на рис. 3). 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 $K$ использовано значение 0,01. Параметр $I_{\max}$ выбирается исходя из вычислительных возможностей и допустимой вероятности ошибочного декодирования.

**Алгоритм 1.** $\mathsf{ISDDecoder}$ — декодер $D$-кода с применением ДИС

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

1: $result = \perp$.
2: $\mathbf{z}' = (\mathbf{z}'_1 \parallel \ldots \parallel \mathbf{z}'_{n_1}) = \mathsf{BlockDecoder}_{n_1,n_2}(\mathbf{z}, \mathsf{DecRow})$.
3: $B = \{j \in [\![1, n_1]\!] : \mathbf{z}'_j = \perp\}$.
4: **Если** $\left(\bigcup\limits_{j \in B} [\![(j-1)n_2 + 1, jn_2]\!]\right) \bigcap \tau_0 = \varnothing$, **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 $result = \perp$:
*// 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 $\omega = \bigcup\limits_{j \in W} [\![(j-1)n_2 + 1, jn_2]\!]$.
13: $\quad\quad$ **Если** $\operatorname{rank}(G') = k$, **то**
14: $\quad\quad\quad$ найти такое $\tau \subset \omega$, что $|\tau| = k$, $\operatorname{rank}(\Pi_\tau(G)) = k$;
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** $result$.

Оценим вероятность ошибочного декодирования для алгоритма 1. В случае $i \in \mathbb{N}$ итераций ДИС эту вероятность можно вычислить по следующей формуле:

$$\mathrm{DFR}_1(i) = 1 - \left(P_{\mathrm{dec}}^{(0)} + \sum_{j=1}^{n_1} \left(1 - (1 - P_1^{(j)} P_2)^i\right) P_{\mathrm{dec}}^{(j)}\right). \tag{8}$$

Здесь $P_1^{(j)} = \mathrm{C}_{n_1-j}^K / \mathrm{C}_{n_1}^K$ — 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); $P_{\mathrm{dec}}^{(j)}$ — вероятность появления $j$ плохих блоков в зашумлённом кодовом векторе. Вероятность $P_{\mathrm{dec}}^{(j)}$ рассчитывается по формуле

$$P_{\mathrm{dec}}^{(j)} = \mathrm{C}_{n_1}^j P_{\mathrm{bad}}^j (1 - P_{\mathrm{bad}})^{n_1 - j}, \tag{9}$$

где $P_{\text{bad}}$ — вероятность того, что некоторый фиксированный блок станет плохим после применения на шаге 2 алгоритма $\mathsf{BlockDecoder}_{n_1,n_2}$. Заметим, что $P_{\text{bad}}$ зависит от кода, декодера $\mathsf{DecRow}$ и числа ошибок $t$. В настоящей работе $P_{\text{bad}}$ оценивается экспериментально. Поясним формулу (8). Значение $\text{DFR}_1(i)$ получается вычитанием из единицы вероятности правильного декодирования за $i$ итераций ДИС, которая, в свою очередь, рассчитывается как сумма двух вероятностей. Первая — вероятность $P_{\text{dec}}^{(0)}$ того, что на шаге 2 после работы алгоритма $\mathsf{BlockDecoder}_{n_1,n_2}$ будет получено правильное кодовое слово. Вторая — вероятность правильного декодирования после $i$ итераций ДИС, в которой учитываются случаи появления $j$ плохих блоков, где $j = 1, \ldots, n_1$. 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 (7), соответствующих хорошим блокам вектора $\mathbf{z}$ и составляющих матрицу ранга $k$.

Оценим вычислительную эффективность алгоритма 1. Пусть $T_{block}$ — среднее время, необходимое для выполнения алгоритма $\mathsf{BlockDecoder}_{n_1,n_2}$, а $T_{\text{ISD}}$ — среднее время, необходимое для выполнения одной итерации ДИС, тогда среднее время $T_1$, необходимое для правильного декодирования алгоритмом 1, можно вычислить по формуле

$$T_1 = T_{\text{block}} + \sum_{i=1}^{I_{\max}} i T_{\text{ISD}} \sum_{j=0}^{n_1} \left(1 - P_1^{(j)} P_2\right)^{i-1} P_1^{(j)} P_2 P_{\text{dec}}^{(j)}. \tag{10}$$

Поясним формулу (10). Для нахождения $T_1$ необходимо к времени $T_{\text{block}}$ добавить среднее время работы ДИС, которое зависит от среднего количества итераций. Поэтому во втором слагаемом в (10) 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. Благодаря сумме по $j$ учитываются все возможные варианты количества плохих блоков и вероятности их появления.

*Декодер на основе декодирования кодов-произведений*

Из вида (5) следует, что

$$C \subseteq (\mathbb{F}_2^{n_1} \otimes \text{RM}(\ell_1, m_2)) \cap (\text{RM}(k_s, m_1) \otimes \mathbb{F}_2^{n_2}),\ n_i = 2^{m_i},\ i = 1, 2.$$

Следовательно, алгоритм декодирования зашумлённого 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 — зашумлённые кодовые слова кода $\text{RM}(\ell_1, m_2)$, столбцы — зашумлённые кодовые слова кода $\text{RM}(k_s, m_1)$; 2) применить декодер по строкам, а затем к результату декодирования применить декодер по столбцам. Декодеры для строк и столбцов могут быть разными. Например, к строкам может быть применён декодер из работы [17] для исправления большого числа ошибок, а к столбцам — вычислительно более простой декодер, например мажоритарный, так как ожидается, что строчный декодер исправит «почти все» ошибки и для декодера по столбцам достаточно простого алгоритма. Декодер по строкам может вернуть символ $\perp$, в этом случае соответствующий блок может быть заменён 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, что всегда можно переставить координаты вектора $\mathbf{z}$ так, что вектор с переставленными координатами будет представлять конкатенацию $n_2$ зашумлённых кодовых слов кода $\text{RM}(k_s, m_1)$ длины $n_1$ каждый. Такая перестановка зависит только от параметров $n_1$ и $n_2$; обозначим её $\pi$. Следовательно, при декодировании вектора $\mathbf{z}$ необязательно представлять его в виде матрицы: при декодировании по строкам

к вектору $\mathbf{z}$ применяется алгоритм $\mathsf{BlockDecoder}_{n_1,n_2}$ с декодером по строкам $\mathsf{DecRow}$ в качестве второго аргумента, а при декодировании по столбцам к вектору $\pi(\mathbf{z}')$ применяется алгоритм $\mathsf{BlockDecoder}_{n_2,n_1}$, в котором вторым аргументом является декодер по столбцам $\mathsf{DecCol}$. Здесь $\mathbf{z}'$ обозначает результат после декодирования по строкам. Описанный способ декодирования приведён в алгоритме 2. В качестве $\mathsf{ExitCond}$ могут быть использованы алгоритмы, представленные на рис. 3.

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 2, можно вычислить как вероятность появления плохих блоков после работы строчного декодера в количестве, превышающем половину кодового расстояния кода $\mathrm{RM}(k_s, m_1)$. То есть

$$\mathrm{DFR}_2 = \sum_{j=\lfloor (d_2-1)/2 \rfloor + 1}^{n_1} P_{\mathrm{dec}}^{(j)}, \tag{11}$$

где $d_2$ — минимальное кодовое расстояние кода $\mathrm{RM}(k_s, m_1)$; $P_{\mathrm{dec}}^{(j)}$ имеет вид (9).

**Алгоритм 2.** $\mathsf{MatrixDecoder}$ — декодер $D$-кода без применения ДИС

**Вход:** $\mathbf{z}; \tau_0, G, \mathsf{ExitCond}, \mathsf{op}_1, \mathsf{op}_2, \mathsf{DecRow}, \mathsf{DecCol}$.

1: $result = \perp$.
2: $\mathbf{z}' = (\mathbf{z}'_1 \parallel \ldots \parallel \mathbf{z}'_{n_1}) = \mathsf{BlockDecoder}_{n_1,n_2}(\mathbf{z}, \mathsf{DecRow})$.
3: $B_r = \{j \in [\![1, n_1]\!] : \mathbf{z}'_j = \perp\}$.
4: **Если** $\big( \bigcup\limits_{j \in B_r} [\![(j-1)n_2 + 1, jn_2]\!] \big) \cap \tau_0 = \varnothing$, **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 $(\star, \ldots, \star)$ длины $n_2$;
10: $\quad \mathbf{z}'' = (\mathbf{z}''_1 \parallel \ldots \parallel \mathbf{z}''_{n_2}) = \mathsf{BlockDecoder}_{n_2,n_1}(\pi(\mathbf{z}'), \mathsf{DecCol})$;
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** $result$

Алгоритм 2, в отличие от алгоритма 1, имеет постоянное время выполнения. Оно складывается из времени работы $\mathsf{BlockDecoder}_{n_1,n_2}$ с декодером $\mathsf{DecRow}$ на шаге 2 ($T_{\mathsf{DecRow}}$) и времени работы $\mathsf{BlockDecoder}_{n_2,n_1}$ с декодером $\mathsf{DecCol}$ на шаге 10 ($T_{\mathsf{DecCol}}$), то есть

$$T_2 = T_{\mathsf{DecRow}} + T_{\mathsf{DecCol}}. \tag{12}$$

## 4. Система шифрования типа Мак-Элиса на $D$-кодах и её характеристики

### 4.1. Система шифрования типа Мак-Элиса на $D$-кодах

Схему типа Мак-Элиса на основе $D$-кода $C$ вида (5) с характеристиками $[n, k, d]_2$ и декодером $\mathsf{DDecoder} \in \{\mathsf{GraphDecoder}, \mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}$ будем обозначать $\mathsf{McE}(D, \mathsf{DDecoder})$; тройка соответствующих алгоритмов приведена на рис. 4.

| | $\mathsf{Gen}^{\mathsf{McE}(D)}(t)$ : | $\mathsf{Enc}(pk, \mathbf{m}; \mathbf{e})$ : | $\mathsf{Dec}^{\mathsf{DDecoder}}(sk, \mathbf{c})$ : |
|---|---|---|---|
| 1 | $S \leftarrow \mathrm{GL}_k(\mathbb{F}_2)$ | $\mathbf{c} = \mathbf{m}\tilde{G} + \mathbf{e}$ | $\mathbf{x} \leftarrow \mathsf{DDecoder}(\mathbf{c}P^{-1})$ |
| 2 | $P \leftarrow \mathcal{P}_n$ | ***вернуть*** $\mathbf{c}$ | *если* $\mathbf{x} \neq \perp$ |
| 3 | $pk = (SGP, t)$ | | $\mathbf{m}' = \mathbf{x}G^{-1}S^{-1}$ |
| 4 | $sk = (S, P)$ | | ***вернуть*** $\mathbf{m}$ |
| 5 | ***вернуть*** $(pk, sk)$ | | *иначе* |
| 6 | | | ***вернуть*** $\perp$ |

Рис. 4. Схема $\mathsf{McE}(D, \mathsf{DDecoder})$, $\mathsf{DDecoder} \in \{\mathsf{GraphDecoder}, \mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}$

Формально схема на рис. 4 отличается от схемы на рис. 1 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 $\mathsf{Enc}$, определяется из условия

$$t = \begin{cases} \lfloor (d-1)/2 \rfloor, & \mathsf{DDecoder} = \mathsf{GraphDecoder}, \\ t' > \lfloor (d-1)/2 \rfloor, & \mathsf{DDecoder} \in \{\mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}, \end{cases} \tag{13}$$

где $t'$ выбирается, исходя из кода, декодера, вероятности ошибочного расшифрования и требуемого уровня стойкости $\lambda_{\mathsf{OW}}$ соответствующей системы шифрования. Более подробно выбор $t'$ для некоторых систем $\mathsf{McE}(D, \mathsf{DDecoder})$ обсуждается в следующем пункте. В правиле расшифрования $\mathsf{Dec}$ на рис. 4 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 $\tau_0$, порождающая матрица $G$, условие $\mathsf{ExitCond}$, алгоритмы $\mathsf{op}_1$ и $\mathsf{op}_2$ 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 декодера, а не как его параметры.

### 4.2. Характеристики схемы $\mathsf{McE}(D, \mathsf{DDecoder})$

Обсудим выбор характеристик схемы $\mathsf{McE}(D, \mathsf{DDecoder})$, обеспечивающих заданный уровень OW-стойкости схемы, в частности целевые значения $\lambda_{\mathsf{OW}}$ из табл. 1.

В [13] найдены условия на $D$-коды, при которых ключи системы шифрования типа Мак-Элиса на этих кодах являются слабыми: построена структурная атака, позволяющая найти часть закрытого ключа. Эта атака не зависит от декодера, поэтому применима к слабым ключам всех вариантов схемы $\mathsf{McE}(D, \mathsf{DDecoder})$. В работе [14] на основе структурной атаки из [13] разработана комбинированная атака на шифртекст схемы $\mathsf{McE}(D, \mathsf{DDecoder})$. Вероятность успеха комбинированной атаки оказывается достаточно большой, что численно продемонстрировано в случае слабых ключей схемы $\mathsf{McE}(D, \mathsf{GraphDecoder})$. Несмотря на отсутствие соответствующих численных подтверждений для схем $\mathsf{McE}(D, \mathsf{ISDDecoder})$ и $\mathsf{McE}(D, \mathsf{MatrixDecoder})$, естественно предположить нецелесообразность применения слабых ключей и в этих реализациях. Поэтому параметры $\mathsf{McE}(D, \mathsf{DDecoder})$ следует определять, с одной стороны, исходя из обеспечения стойкости к структурной атаке, построенной в [13, 14], а с другой — из требования обеспечения заданного уровня OW-стойкости $\lambda_{\mathsf{OW}}$ (4).

В [14] исследованы характеристики схемы шифрования $\mathsf{McE}(D, \mathsf{GraphDecoder})$, где показано, что эта схема достигает высокого уровня стойкости ($\lambda_{\mathsf{OW}} \geqslant 128$) при большом размере открытого ключа (более 154 Мбайт). Целью настоящей работы является уменьшение размера ключа при сохранении уровня OW-стойкости за счёт использования вероятностных декодеров, в частности декодеров ISDDecoder и MatrixDecoder. В качестве DecRow в обоих алгоритмах используется декодер Йе — Аббе [17], 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 (см. рис. 3), которое используется в алгоритмах декодирования ISDDecoder и MatrixDecoder, при расшифровании в качестве алгоритма $\mathsf{op}_1$ 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 закрытого ключа $S$, а в качестве алгоритма $\mathsf{op}_2$ — алгоритм $\mathsf{mul}_P$, реализующий умножение вектора на матрицу $P$.

Из условия (13) на вес вектора ошибки в правиле шифрования Enc вытекает, что в (9) $P_{\text{bad}} \neq 0$, поэтому декодеры ISDDecoder и MatrixDecoder характеризуются ненулевой вероятностью ошибочного декодирования (см. (8) и (11) соответственно). Из (8) вытекает, что $\lim\limits_{i\to\infty} \mathrm{DFR}_1(i) = 0$, 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 (11), вероятность ошибочного декодирования зависит только от вероятности $P_{\text{bad}}$, то есть от числа добавляемых ошибок при шифровании, поэтому алгоритмически снизить вероятность ошибочного декодирования не удаётся. Однако, как отмечено выше, в отличие от ISDDecoder декодер MatrDecoder является декодером с постоянным временем декодирования. Это может быть преимуществом MatrDecoder, так как в случае, когда для криптографического протокола актуальны атаки на кодовую систему шифрования 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 (8) в алгоритме ISDDecoder не более $10^{-\gamma}$, обозначим $I_\gamma$:

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

Найденное значение $I_\gamma$ предполагается использовать в качестве $I_{\max}$ в алгоритме ISDDecoder для ограничения сверху вероятности ошибочного декодирования и, как следствие, обеспечения требуемого уровня отказоустойчивости $\gamma$.

В табл. 2 представлены параметры $D$-кодов вида (5), выбранные для использования в системе шифрования типа Мак-Элиса. Эти коды получены следующим образом. В качестве $\mathrm{RM}(\ell_1, m_2)$ из вида (6) выбраны коды $\mathrm{RM}(2,7)$, $\mathrm{RM}(3,7)$, $\mathrm{RM}(2,8)$, $\mathrm{RM}(3,8)$, $\mathrm{RM}(2,9)$, обладающие высокой корректирующей способностью при использовании де-

кодера Йе — Аббе [17]. Для каждого из этих кодов путем перебора значений параметра $m_1$ найдены $D$-коды вида (5), такие, что соответствующие им системы шифрования, во-первых, являются стойкими к комбинированной атаке из [14], а, во-вторых, уровни стойкости $\lambda_{\mathsf{OW}}$ этих систем к классической атаке декодированием по информационным совокупностям сопоставимы с уровнями стойкости соответствующих систем из табл. 1. Для начальной оценки уровня стойкости $\lambda_{\mathsf{OW}}$ по формуле (4) использовано значение числа добавляемых ошибок $t$, полученное умножением количества ошибок, исправляемых декодером Йе — Аббе для кода $\mathrm{RM}(\ell_1, m_2)$ (с вероятностью ошибочного декодирования не более $10^{-4}$), на количество блоков $2^{m_1}$. В дальнейшем оценка значения $\lambda_{\mathsf{OW}}$ уточнялась путём подбора для построенных декодеров количества добавляемых ошибок $t$ при экспериментальных вычислениях вероятности ошибочного декодирования, результаты которых содержатся в табл. 3 и 5. В итоге в табл. 2 попали $D$-коды, у которых размер ключа не больше, чем у систем соответствующего уровня OW-стойкости из табл. 1. В табл. 2 приводятся номер кода и обозначение $[(r_1^1, r_1^2), (r_2^1, r_2^2), \ldots]$ соответствующего $D$-кода:

$$[(r_1^1, r_1^2), (r_2^1, r_2^2), \ldots] = \mathrm{RM}(r_1^1, m_1) \otimes \mathrm{RM}(r_1^2, m_2) + \mathrm{RM}(r_2^1, m_1) \otimes \mathrm{RM}(r_2^2, m_2) + \ldots,$$

а также приведены параметры $m_1$, $m_2$ и $[n, k, d]$.

Таблица 2

**$D$-коды для системы типа Мак-Элиса**

| 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 |
| 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 табл. 3 приводится экспериментальная оценка корректирующей способности декодера $\mathsf{ISDDecoder}$ для кодов из табл. 2. Оценка выполнена следующим образом. Для каждого кода из табл. 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 зашумлённый кодовый вектор подавался на вход алгоритму $\mathsf{BlockDecoder}_{n_1,n_2}$. После выполнения алгоритма происходил подсчёт количества блоков зашумлённого кодового слова, которые были декодированы неправильно (подсчёт

плохих блоков). Перечисленные действия повторялись указанное в табл. 3 количество раз и на основе итогового количества плохих блоков было рассчитано значение $P_{\text{bad}}$ из (9) и $\text{DFR}_1(0) = 1-(1-P_{\text{bad}})^{2^{m_1}}$ в соответствии с (8). Значение $\text{DFR}_1(0)$ фактически означает вероятность ошибочного декодирования алгоритмом ISDDecoder до использования ДИС и может быть учтено для оценки адекватности выбранного количества ошибок для обеспечения требуемой отказоустойчивости. Для каждого кода вычислены значения количества блоков $K$, выбираемых на шаге 11 алгоритма ISDDecoder, и соответствующие им значения вероятностей $P_2$ из (8). Значения $K$ вычислены экспериментально: для каждого кода в качестве $K$ выбрано минимальное количество блоков, при котором за 100 попыток составления подматрицы из $K$ случайных блоков соответствующей порождающей матрицы хотя бы одна попытка даёт 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 (8) вычислено количество $I_9$ итераций ДИС, необходимых для обеспечения уровня отказоустойчивости $\gamma = 9$ IND-CPA-стойкого механизма KEM. Дополнительно в табл. 3 указано максимальное за наблюдаемое количество экспериментов количество плохих блоков после декодирования зашумлённого кодового вектора.

Как видно из табл. 3, для большинства кодов используемые значения количества добавляемых ошибок $t$ позволяют получить приемлемые значения вероятности $\text{DFR}_1(0)$ и, как следствие, количества итераций $I_9$. Однако для кодов 2 и 3 вероятность $\text{DFR}_1(0)$ получилась достаточно большой, что привело к значительному увеличению $I_9$. Также стоит отметить, что для кода 17, несмотря на относительно малую вероятность $\text{DFR}_1(0)$, количество итераций $I_9$ всё равно получается значительным. Это происходит из-за того, что для кода 17 найденное значение $K$ получается слишком близким к общему количеству 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 уровня отказоустойчивости $\gamma = 9$.

Таблица 3

**Оценка корректирующей способности декодера ISDDecoder**

| Номер кода | $K$ | $P_2$ | $t$ | Кол-во экспериментов | Макс. кол-во плохих блоков | $\text{DFR}_1(0)$ | $I_9$ |
|---|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* |
| 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 |

Окончание табл. 3

| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* |
|---|---|---|---|---|---|---|---|
| 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 |
| 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 |
| 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 табл. 4 приводится оценка времени декодирования с помощью алгоритма ISDDecoder кодов из табл. 2. Для каждого набора параметров экспериментов из табл. 3 ($D$-кода, количества блоков $K$ и числа ошибок $t$) приводится время декодирования в миллисекундах (мс), а именно: $T_{\text{block}}$ — среднее время выполнения алгоритма $\mathsf{BlockDecoder}_{n_1,n_2}$ на шаге 2; $T_{\text{ISD}}$ — среднее время выполнения одной итерации ДИС (шаги 10–18); $T_1$ — среднее время декодирования с помощью алгоритма ISDDecoder, вычисленное в соответствии с (10); $T_1'$ — время выполнения алгоритма для обеспече-

ния уровня отказоустойчивости $\gamma = 9$ в худшем случае (без раннего выхода из ДИС), то есть $T_1' = T_{\text{block}} + T_{\text{ISD}} I_9$.

Таблица 4

**Оценка времени декодирования алгоритмом ISDDecoder**

| Номер кода | $K$ | $t$ | $T_{\text{block}}$ | $T_{\text{ISD}}$ | $T_1'$ | $T_1$ |
|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 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 |
| 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 |

Окончание табл. 4

| *1* | *2* | *3* | *4* | *5* | *6* | *7* |
|---|---|---|---|---|---|---|
| 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 |
| 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 табл. 5 приводится оценка корректирующей способности декодера MatrixDecoder и его времени декодирования для кодов из табл. 2. В качестве алгоритмов DecRow и DecCol использован декодер Йе — Аббе [17]. В табл. 5 также содержатся оценка $\mathrm{DFR}_2$ в соответствии с (11) и оценка $T_2$ времени выполнения алгоритма (в мс) в соответствии с (12).

Таблица 5

**Оценка корректирующей способности декодера** MatrixDecoder **и его времени декодирования**

| Номер кода | $t$ | $\mathrm{DFR}_2$ | $T_2$ | Номер кода | $t$ | $\mathrm{DFR}_2$ | $T_2$ |
|---|---|---|---|---|---|---|---|
| 1 | 4000 | 1,54E–12 | 2375 | 11 | 1706 | 5,963E–20 | 7985 |
| 1 | 3749 | 1,173E–17 | 2375 | 12 | 1706 | 2,167E–9 | 127822 |
| 2 | 3392 | 0,000273 | 3172 | 13 | 7687 | 1,344E–14 | 8673 |
| 3 | 3392 | 0,000273 | 3235 | 13 | 7772 | 1,128E–13 | 8673 |
| 4 | 1744 | 0,0296 | 906 | 14 | 673 | 0,000104 | 45800 |
| 5 | 1194 | 0,119 | 9892 | 15 | 1706 | 5,963E–20 | 8001 |
| 6 | 1706 | 5,963E–20 | 7828 | 16 | 1292 | 3,802E–9 | 81084 |
| 7 | 3830 | 5,338E–8 | 2907 | 17 | 1706 | 0,000115 | 1233434 |
| 8 | 612 | 5,522E–6 | 43597 | 18 | 7279 | 6,379E–10 | 41893 |
| 9 | 612 | 5,522E–6 | 46722 | 19 | 1083 | 2,955E–14 | 80396 |
| 10 | 3237 | 4,557E–6 | 3156 | | | | |

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 счёт увеличения максимального времени декодирования. Например, для кода 1 из табл. 2 и $t = 3749$ декодеру ISDDecoder в худшем случае потребуется 26 итераций ДИС и около 2600 мс для обеспечения вероятности ошибочного декодирования $10^{-9}$; для тех же параметров декодер MatrixDecoder при меньшем времени декодирования (2300 мс) обеспечивает DFR около $10^{-17}$, что значительно меньше. Однако, например, для кодов 4 и 5 декодер MatrixDecoder не может обеспечить даже $\mathrm{DFR} = 10^{-9}$, в отличие от ISDDecoder.

В табл. 6 приводятся параметры системы шифрования типа Мак-Элиса на основе $D$-кодов из табл. 2. В частности, приводится значение размера открытого ключа в ме-

габайтах (Мбайт), а также значение уровня OW-стойкости $\lambda_{\mathsf{OW}}$ в соответствии с (4) при выбранном значении количества ошибок $t$.

Таблица 6

**Характеристики системы** $\mathsf{McE}(D, \mathsf{DDecoder})$, $\mathsf{DDecoder} \in \{\mathsf{ISDDecoder}, \mathsf{MatrixDecoder}\}$

| Номер кода | Размер ключа, Мбайт | $t$ | $\lambda_{\mathsf{OW}}$ | Номер кода | Размер ключа, Мбайт | $t$ | $\lambda_{\mathsf{OW}}$ |
|---|---|---|---|---|---|---|---|
| 1 | 0,734 | 4000 | 153,87 | 11 | 1,711 | 1706 | 143,04 |
| 1 | 0,734 | 3749 | 142,82 | 12 | 1,588 | 1706 | 132,47 |
| 2 | 0,809 | 3392 | 140,55 | 13 | 2,625 | 7687 | 262,28 |
| 3 | 0,793 | 3392 | 137,8 | 13 | 2,625 | 7772 | 265,61 |
| 4 | 0,393 | 1744 | 142,82 | 14 | 1,814 | 673 | 262,22 |
| 5 | 0,589 | 1194 | 142,81 | 15 | 2,189 | 1706 | 184,58 |
| 6 | 1,328 | 1706 | 110,3 | 16 | 4,035 | 1292 | 262,47 |
| 7 | 0,93 | 3830 | 185,96 | 17 | 3,064 | 1706 | 262,40 |
| 8 | 1,463 | 612 | 186,11 | 18 | 3,211 | 7279 | 302,23 |
| 9 | 1,541 | 612 | 197,24 | 19 | 5,441 | 1083 | 302,32 |
| 10 | 1,121 | 3237 | 185,94 | | | | |

В табл. 7 приводится сравнение характеристик системы шифрования на кодах Гоппы ($\mathsf{ME}(G)$) и системы шифрования типа Мак-Элиса на $D$-кодах. В качестве кодов Гоппы рассматриваются коды с характеристиками из табл. 1, причём размер ключа вычисляется как для случая, когда шифрование выполняется по правилу, предложенному Р. Мак-Элисом (Original McEliece), так и для шифрования по правилу, предложенному Х. Нидеррайтером Classic McEliece. Для сравнения выбраны $D$-коды из табл. 2, которые при меньшем размере ключа, чем в Original McEliece, обладают таким же уровнем OW-стойкости $\lambda_{\mathsf{OW}}$, как и соответствующая система Мак-Элиса на кодах Гоппы. Среди множества подходящих $D$-кодов выбирался код с наименьшим временем декодирования при $\mathrm{DFR} \leqslant 10^{-9}$, то есть в таблице фактически рассматривается случай $\gamma = 9$.

Таблица 7

**Сравнение характеристик систем шифрования типа Мак-Элиса**

| $\mathsf{McE}(G)$ | | | | | $\mathsf{McE}(D, \mathsf{DDecoder})$ | | | |
|---|---|---|---|---|---|---|---|---|
| Криптосистема mceliece | $t$ | $\lambda_{\mathsf{OW}}$ | Размер ключа, Original McEliece, Мбайт | Размер ключа, Classic McEliece, Мбайт | Номер кода | $t$ | $\lambda_{\mathsf{OW}}$ | Размер ключа, Мбайт |
| 348864 | 64 | 142,8 | 1,13 | 0,25 | 1 | 3749 | 142,82 | 0,73 |
| 460896 | 96 | 185,92 | 1,85 | 0,5 | 7 | 3830 | 185,96 | 0,93 |
| 6688128 | 128 | 262,28 | 4,01 | 1 | 13 | 7687 | 262,28 | 2,63 |
| 6960119 | 119 | 265,6 | 4,49 | 1 | 13 | 7772 | 265,61 | 2,63 |
| 8192128 | 128 | 302,21 | 6,38 | 1,29 | 18 | 7279 | 302,23 | 3,21 |

Как видно из табл. 7, использование $D$-кодов в системе шифрования типа Мак-Элиса позволяет сократить размер открытого ключа относительно системы Original McEliece. Отметим, что в табл. 7 имеются $D$-коды, для которых система шифрования обладает меньшим размером ключа даже при значительно большем уровне стойкости $\lambda_{\mathsf{OW}}$. 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 для уровня отказоустойчивости $\gamma = 9$ размер открытых ключей системы на $D$-кодах не столь существенно, но все же превышает размер открытых ключей системы Classic McEliece. Представляется, что размер ключей системы на $D$-кодах может быть уменьшен, если ослабить требование к отказоустойчивости, то есть при $\gamma < 9$.

Сравнивая системы шифрования Мак-Элиса на кодах Гоппы и $D$-кодах, можно заметить, что время расшифрования у последней больше при декодировании как алгоритмом ISDDecoder, так и алгоритмом MatrixDecoder. Однако в табл. 1 приводится время расшифрования при использовании оптимизированной реализации декодера из [3], в то время как в табл. 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 счёт параллельного декодирования блоков в $\mathsf{BlockDecoder}_{n_1,n_2}$ и $\mathsf{BlockDecoder}_{n_2,n_1}$.

## 5. Механизм KEM на основе $\mathsf{McE}(D, \mathsf{DDecoder})$ и его характеристики

Тройку алгоритмов механизма защищённой передачи, полученного путём применения преобразования Фудзисаки — Окамото $\mathsf{FO}^{\not\perp}$ на основе схемы шифрования $\mathsf{McE}(D, \mathsf{DDecoder})$, обозначим $\mathsf{KEM}(D, \mathsf{DDecoder}) = (\mathsf{Gen}^{\mathsf{KEM}(D)}, \mathsf{Encaps}, \mathsf{Decaps})$ (рис. 5).

| | $\mathsf{Gen}^{\mathsf{KEM}(D)}(t)$ : | $\mathsf{Encaps}(pk')$ : | $\mathsf{Decaps}(sk', \mathbf{c}, \mathsf{ExitCond})$ : |
|---|---|---|---|
| 1 | $\mathbf{s} \leftarrow \mathbb{F}_2^k$ | $\mathbf{r} \leftarrow \mathbb{F}_2^k$ | $\mathbf{r}' \leftarrow \mathsf{Dec}^{\mathsf{DDecoder}}(sk, \mathbf{c})$ |
| 2 | $(pk, sk) \leftarrow \mathsf{Gen}^{\mathsf{McE}(D)}(t)$ | $\mathbf{c} \leftarrow \mathsf{Enc}(pk, \mathbf{r}; \mathsf{G}(\mathbf{r}))$ | *если* $\mathbf{r}' \neq \perp$ |
| 3 | $pk' = pk$ | $\mathbf{K} \leftarrow \mathsf{H}(\mathbf{r} \parallel \mathbf{c})$ | *если* $\mathsf{ExitCond} = \mathsf{ExitCond}_{\mathsf{wt}}$ |
| 4 | $sk' = (sk, \mathbf{s})$ | ***вернуть*** $(\mathbf{K}, \mathbf{c})$ | $\mathbf{c}' \leftarrow \mathsf{Enc}(pk, \mathbf{r}'; \mathsf{G}(\mathbf{r}'))$ |
| 5 | ***вернуть*** $(pk', sk')$ | | *если* $\mathbf{c} = \mathbf{c}'$ |
| 6 | | | ***вернуть*** $\mathsf{H}(\mathbf{r}' \parallel \mathbf{c})$ |
| 7 | | | *иначе* |
| 8 | | | ***вернуть*** $\mathsf{H}(\mathbf{r}' \parallel \mathbf{c})$ |
| 9 | | | ***вернуть*** $\mathsf{H}(\mathbf{s} \parallel \mathbf{c})$ |

Рис. 5. Механизм $\mathsf{KEM}(D, \mathsf{DDecoder}) = (\mathsf{Gen}^{\mathsf{KEM}}, \mathsf{Encaps}, \mathsf{Decaps})$, полученный путём применения преобразования $\mathsf{FO}^{\not\perp}$ на основе схемы $\mathsf{McE}(D, \mathsf{DDecoder})$ с детерминированным правилом шифрования; $\mathsf{ExitCond} \in \{\mathsf{ExitCond}_{\mathsf{wt}}, \mathsf{ExitCond}_{\mathsf{G}}\}$

Согласно [18], при выполнении условия (3) для многоразового использования ключей механизм KEM, полученный применением преобразования Фудзисаки — Окамото $\mathsf{FO}^{\not\perp}$, обеспечивает IND-CCA-стойкость уровня $\lambda_{\mathsf{KEM}}$, если для схемы асимметричного шифрования $\mathsf{McE}(D, \mathsf{DDecoder})$ выполняется условие (1). Отсюда вытекает и IND-CPA-стойкость механизма при выполнении тех же условий.

Выше отмечалось, что выбор параметров $\mathsf{McE}(D, \mathsf{DDecoder})$ следует выполнять, исходя из обеспечения стойкости к структурной атаке, построенной в [13, 14], и из требования обеспечения заданного уровня OW-стойкости $\lambda_{\mathsf{OW}}$. При синтезе механизма $\mathsf{KEM}(D, \mathsf{DDecoder})$ на основе $\mathsf{McE}(D, \mathsf{DDecoder})$ дополнительными условиями на выбор параметров схемы $\mathsf{McE}(D, \mathsf{DDecoder})$ являются условия (1) и (3).

Схема $\mathsf{McE}(D, \mathsf{DDecoder})$ в случае применения гарантированного декодера, например GraphDecoder [15], позволяет построить $\mathsf{KEM}(D, \mathsf{GraphDecoder})$ с многоразовым использованием ключей, так как условие (3) выполняется ($\delta = 0$). Однако результа-

ты исследования [14] показали, что для обеспечения целевых значений уровней OW-стойкости, сопоставимых со значениями из табл. 1, приходится использовать ключи существенно большего размера, чем в Original McEliece.

Как показывают результаты в табл. 7, при использовании вероятностных декодеров ISDDecoder и MatrixDecoder в настоящей работе не удалось найти $D$-коды, для которых бы размер ключа системы шифрования не превышал размер ключа системы Original McEliece из табл. 1, обеспечивая при этом уровень стойкости не менее соответствующего значения $\lambda_{\mathsf{OW}}$, 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. табл. 2) для схемы $\mathsf{McE}(D, \mathsf{DDecoder})$ с построенными вероятностными декодерами не позволяют при меньшем размере ключа, чем в Original McEliece, построить механизм $\mathsf{KEM}(D, \mathsf{DDecoder})$ с соответствующим уровнем IND-CCA-стойкости $\lambda_{\mathsf{KEM}}$. Действительно, для обеспечения IND-CCA-стойкости необходимо, с одной стороны, чтобы уровень стойкости $\lambda_{\mathsf{KEM}}$ был не меньше, чем длина $m$ сеансового ключа $K$ (см. (1)), которая обычно не меньше 128, а с другой — чтобы DFR был меньше, чем $2^{-\lambda_{\mathsf{KEM}}}$ (см. (3)). Согласно табл. 3 и 5, минимальный полученный DFR для используемых параметров $D$-кодов оказался значительно больше $2^{-128}$. Однако найденные коды позволяют обеспечить уровень отказоустойчивости $\gamma = 9$ для IND-CPA-стойкого механизма KEM с одноразовым использованием ключей (см. условие (3) для одноразового использования). В частности, такие коды удалось найти за счёт 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; как следствие, этот же механизм может применяться для одноразового использования. Схема Мак-Элиса на $D$-кодах при сопоставимых размерах ключа (меньшем размере по отношению к Original McEliece и несколько большем — по отношению к Classic McEliece) в случае использования вероятностных алгоритмов декодирования 1 и 2 позволяет построить только KEM с одноразовым использованием ключей.

## Заключение

Известные асимметричные кодовые системы шифрования на основе кодов Рида — Маллера [36–38] оказались нестойкими к структурным атакам [39–44]. В [13–15] и настоящей работе предпринимается попытка реанимировать двоичные коды 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 открытого ключа). В настоящей работе удалось построить механизм KEM с меньшим ключом, чем в оригинальной системе шифрования на кодах Гоппы Original McEliece, только для передачи эфемерных (одноразовых) сеансовых криптографических ключей. Однако, учитывая интерес криптографического сообщества к KEM с одноразовым использованием ключей [24], это ограничение представляется не таким существенным. Кроме того, как было недавно показано, двоичные коды Рида — Маллера достигают ёмкости в симметричных каналах без памяти [30], поэтому ожидается дальнейшее развитие методов эффективного декодирования этих кодов, которые могут позволить не только уменьшить время расшифрования в предлагаемой системе шифрования, но и повы-

сить класс стойкости за счёт уменьшения вероятности ошибочного декодирования. Возможность исправлять ошибки большого веса также может быть использована для усиления структурной стойкости системы, например, за счёт добавления случайных столбцов к порождающей матрице кода и/или замены матрицы перестановки $P$ невырожденной матрицей специального вида.

Авторы выражают благодарность рецензенту за внимательное прочтение статьи, а также за полезные замечания и советы, которые позволили ее улучшить.

## ЛИТЕРАТУРА

1. *Weger V., Gassner N., and Rosenthal J.* A Survey on Code-Based Cryptography. arXiv: 2201.07119, 2024. 168 p.
2. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory // Deep Space Network Progress Report. 1978. No. 44. P. 114–116.
3. *Albrecht M. R., Bernstein D. J., Chou T., et al.* Classic McEliece: Conservative Code-Based Cryptography. NIST PQC Call for Proposals, 2022. Round 4 Submission. `https://classic.mceliece.org/nist/mceliece-20201010.pdf`.
4. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST, 2022. `https://csrc.nist.gov/pubs/ir/8413/upd1/final`.
5. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory // Prob. Contr. Inform. Theory. 1986. V. 15. No. 2. P. 157–166.
6. *Albrecht M., Cid C., Paterson K. G., et al.* Nts-kem // NIST PQC Round. 2019. V. 2. P. 4–13.
7. *Couvreur A., Mora R., and Tillich J.-P.* A New Approach Based on Quadratic Forms to Attack the McEliece Cryptosystem. Cryptology ePrint Archive. 2023. Paper 2023/950.
8. *Bardet M., Mora R., and Tillich J.-P.* Polynomial Time Key-Recovery Attack on High Rate Random Alternant Codes. arXiv: 2304.14757, 2023.
9. *Mora R. and Tillich J.-P.* On the dimension and structure of the square of the dual of a Goppa code // Des. Codes Cryptogr. 2023. No. 91. P. 1351–1372.
10. *Ivanov F., Kabatiansky G., Krouk E., and Rumenko N.* A new code-based cryptosystem // LNCS. 2020. V. 12087. P. 41–49.
11. *Lau T. Sh. C., Ivanov F., Ariffin M. R. K., et al.* New code-based cryptosystems via the IKKR framework // J. Inform. Security Appl. 2023. V. 76. Article 103530.
12. *Krouk E., Kabatiansky G., and Tavernier C.* McEliece-type cryptosystem based on correction of errors and erasures // 2023 XVIII Intern. Symp. REDUNDANCY. Moscow, Russia, 2023. P. 173–177.
13. *Косолапов Ю. В., Лелюк Е. А.* О структурной стойкости криптосистемы типа Мак-Элиса на сумме тензорных произведений бинарных кодов Рида — Маллера // Прикладная дискретная математика. 2022. № 57. С. 22–39.
14. *Косолапов Ю. В., Лелюк Е. А.* Криптосистема типа Мак-Элиса на $D$-кодах // Математические вопросы криптографии. 2024. Т. 15. №. 2. С. 69–90.
15. *Деундяк В. М., Лелюк Е. А.* Теоретико-графовый метод декодирования некоторых групповых MLD-кодов // Дискретн. анализ и исслед. опер. 2020. Т. 27. № 2. С. 17–42.
16. *Abbe E., Sberlo O., Shpilka A., and Ye M.* Reed — Muller codes // Foundations and Trends in Commun. Inform. Theory. 2023. V. 20. No. 1–2. P. 1–156.
17. *Ye M. and Abbe E.* Recursive projection-aggregation decoding of Reed — Muller codes // IEEE Trans. Inform. Theory. 2020. V. 66. No. 8. P. 4948–4965.
18. *Hofheinz D., Hövelmanns K., and Kiltz E.* A modular analysis of the Fujisaki — Okamoto transformation // LNCS. 2017. V. 10677. P. 341–371

19. *Arago N., Barreto P., Bettaieb S., et al.* BIKE: Bit Flipping Key Encapsulation. `https://hal.science/hal-01671903/document`. 2017.
20. *Guo Q., Johansson T., and Wagner P. S.* A key recovery reaction attack on QC-MDPC // IEEE Trans. Inform. Theory. 2018. V. 65. No. 3. P. 1845–1861.
21. *Vedenev K. V. and Kosolapov Y. V.* A reaction attack against cryptosystems based on quasigroup MDPC codes // 2023 XVIII Intern. Symp. REDUNDANCY. Moscow, Russia, 2023. P. 70–75.
22. *Joux A., Loss J., and Wagner B.* Kleptographic Attacks against Implicit Rejection. `https://eprint.iacr.org/2024/260`. 2024.
23. *Campagna M. and Crockett E.* Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS). `https://www.ietf.org/archive/id/draft-campagna-tls-bike-sike-hybrid-07.html`. 2021.
24. *Drucker N., Gueron S., and Kostic D.* A Lean BIKE KEM Design for Ephemeral Key Agreement. `https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/a-lean-bike-kem.pdf`. 2024.
25. *Baldi M., Barenghi A., Chiaraluce F., et al.* LEDAcrypt: Low-dEnsity parity-check coDebAsed cryptographic systems. `https://www.ledacrypt.org/documents/LEDAcrypt_v3.pdf`. 2019.
26. *Prange E.* The use of information sets in decoding cyclic codes // IRE Trans. Inform. Theory. 1962. V. 8. No. 5. P. 5–9.
27. *Both L. and May A.* Decoding linear codes with high error rate and its impact for LPN security // LNCS. 2018. V. 10786. P. 25–46.
28. *May A. and Ozerov I.*, On computing nearest neighbors with applications to decoding of binary linear codes // LNCS. 2015. V. 9056. P. 203–228.
29. *Kudekar S., Kumar S., Mondelli M., et al.* Reed — Muller codes achieve capacity on erasure channels // Proc. STOC'16. N.Y.: ACM, 2016. P. 658–669.
30. *Abbe E. and Sandon C.* A proof that Reed — Muller codes achieve Shannon capacity on symmetric channels // IEEE 64th Ann. Symp. FOCS. Santa Cruz, CA, USA, 2023. P. 177–193.
31. *Reed I. S.* A class of multiple-error-correcting codes and the decoding scheme // IEEE Trans. Inform. Theory. 1954. V. 4. No. 4. P. 38–492.
32. *Massey J. L.* Threshold Decoding. Cambridge, MA: MIT Press, 1963.
33. *Циммерман К.-Х.* Методы теории модулярных представлений в алгебраической теории кодирования. М.: МЦНМО, 2011.
34. *Сидельников В. М., Першаков А. С.* Декодирование кодов Рида — Маллера при большом числе ошибок // Пробл. передачи информ. 1992. Т. 28. № 3. С. 80–94.
35. *Kasami T. and Lin S.* On the construction of a class of majority-logic decodable codes // IEEE Trans. Inform. Theory. 1971. V. 17. No. 5. P. 600–610.
36. *Сидельников В. М.* Открытое шифрование на основе двоичных кодов Рида — Маллера // Дискретная математика. 1994. Т. 6. № 2. С. 3–20.
37. *Kabatiansky G. and Tavernier C.* A new code-based cryptosystem via pseudorepetition of codes // 16th Intern. Workshop Algebraic Combinat. Coding Theory. Svetlogorsk, Russia, 2018. P. 189–191.
38. *Egorova E., Kabatiansky G., Krouk E., and Tavernier C.* A new code-based public-key cryptosystem resistant to quantum computer attacks // J. Phys. Conf. Ser. 2019. V. 1163. P. 1–5.
39. *Minder L. and Shokrollahi A.* Cryptanalysis of the Sidelnikov cryptosystem // LNCS. 2007. V. 4515. P. 347–360.

40. *Бородин М. А., Чижов И. В.* Эффективная атака на криптосистему Мак-Элиса, построенную на основе кодов Рида — Маллера // Дискретная математика. 2014. Т. 26. № 1. С. 10–20.
41. *Высоцкая В. В.* Квадрат кода Рида — Маллера и классы эквивалентности секретных ключей криптосистемы Мак-Элиса — Сидельникова // Прикладная дискретная математика. Приложение. 2017. № 10. С. 66–68.
42. *Высоцкая В. В.* О структурных особенностях пространства ключей криптосистемы Мак-Элиса — Сидельникова на обобщенных кодах Рида — Соломона // Дискретная математика. 2024. Т. 36. № 4. С. 28–43.
43. *Давлетшина А. М.* Поиск эквивалентных ключей криптосистемы Мак-Элиса — Сидельникова, построенной на двоичных кодах Рида — Маллера // Прикладная дискретная математика. Приложение. 2019. № 12. С. 98–100.
44. *Чижов И. В., Бородин М. А.* Классификация произведений Адамара подкодов коразмерности 1 кодов Рида — Маллера // Дискретная математика. 2020. Т. 32. № 1. С. 115–134.

## REFERENCES

1. *Weger V., Gassner N., and Rosenthal J.* A Survey on Code-Based Cryptography. arXiv:2201.07119, 2024. 168 p.
2. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report, 1978, no. 44, pp. 114–116.
3. *Albrecht M. R., Bernstein D. J., Chou T., et al.* Classic McEliece: Conservative Code-Based Cryptography. NIST PQC Call for Proposals, 2022. Round 4 Submission. `https://classic.mceliece.org/nist/mceliece-20201010.pdf`.
4. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST, 2022. `https://csrc.nist.gov/pubs/ir/8413/upd1/final`.
5. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory. Prob. Contr. Inform. Theory, 1986, vol. 15, no. 2, pp. 157–166.
6. *Albrecht M., Cid C., Paterson K. G., et al.* Nts-kem. NIST PQC Round, 2019, vol. 2, pp. 4–13.
7. *Couvreur A., Mora R., and Tillich J.-P.* A New Approach Based on Quadratic Forms to Attack the McEliece Cryptosystem. Cryptology ePrint Archive, 2023, Paper 2023/950.
8. *Bardet M., Mora R., and Tillich J.-P.* Polynomial Time Key-Recovery Attack on High Rate Random Alternant Codes. arXiv:2304.14757, 2023.
9. *Mora R. and Tillich J.-P.* On the dimension and structure of the square of the dual of a Goppa code. Des. Codes Cryptogr., 2023, no. 91, pp. 1351–1372.
10. *Ivanov F., Kabatiansky G., Krouk E., and Rumenko N.* A new code-based cryptosystem. LNCS, 2020, vol. 12087, pp. 41–49.
11. *Lau T. Sh. C., Ivanov F., Ariffin M. R. K., et al.* New code-based cryptosystems via the IKKR framework. J. Inform. Security Appl., 2023, vol. 76, article 103530.
12. *Krouk E., Kabatiansky G., Tavernier C* McEliece-type cryptosystem based on correction of errors and erasures. 2023 XVIII Intern. Symp. REDUNDANCY, Moscow, Russia, 2023, pp. 173–177.
13. *Kosolapov Yu. V. and Lelyuk E. A.* O strukturnoy stoykosti kriptosistemy tipa Mak-Elisa na summe tenzornykh proizvedeniy binarnykh kodov Rida — Mallera [On the structural security of a McEliece-type cryptosystem based on the sum of tensor products of binary Reed — Muller codes]. Prikladnaya Diskretnaya Matematika, 2022, no. 57, pp. 22–39. (in Russian)

14. *Kosolapov Yu. V. and Lelyuk E. A.* Kriptosistema tipa Mak-Elisa na $D$-kodakh [The McEliece-type cryptosystem based on D-codes]. Matematicheskie Voprosy Kriptografii, 2024, vol. 15, no. 2, pp. 69–90. (in Russian)
15. *Deundyak V. M. and Lelyuk E. A.* A graph-theoretical method for decoding some group MLD-codes. J. Appl. Industr. Math., 2020, vol. 14, no. 2, pp. 265–280.
16. *Abbe E., Sberlo O., Shpilka A., and Ye M.* Reed — Muller Codes. Foundations and Trends in Commun. Inform. Theory, 2023, vol. 20, no. 1–2, pp. 1–156.
17. *Ye M. and Abbe E.* Recursive projection-aggregation decoding of Reed — Muller codes. IEEE Trans. Inform. Theory, 2020, vol. 66, no. 8, pp. 4948–4965.
18. *Hofheinz D., Hövelmanns K., and Kiltz E.* A modular analysis of the Fujisaki — Okamoto transformation. LNCS, 2017, vol. 10677, pp. 341–371.
19. *Arago N., Barreto P., Bettaieb S., et al.* BIKE: Bit Flipping Key Encapsulation. `https://hal.science/hal-01671903/document`, 2017.
20. *Guo Q., Johansson T., and Wagner P.S.* A key recovery reaction attack on QC-MDPC. IEEE Trans. Inform. Theory, 2018, vol. 65, no. 3, pp. 1845–1861.
21. *Vedenev K. V. and Kosolapov Y. V.* A reaction attack against cryptosystems based on quasi-group MDPC codes. 2023 XVIII Intern. Symp. REDUNDANCY, Moscow, Russia, 2023, pp. 70–75.
22. *Joux A., Loss J., and Wagner B.* Kleptographic Attacks against Implicit Rejection. `https://eprint.iacr.org/2024/260`, 2024.
23. *Campagna M. and Crockett E.* Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS). `https://www.ietf.org/archive/id/draft-campagna-tls-bike-sike-hybrid-07.html`, 2021.
24. *Drucker N., Gueron S., and Kostic D.* A Lean BIKE KEM Design for Ephemeral Key Agreement. `https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/a-lean-bike-kem.pdf`, 2024.
25. *Baldi M., Barenghi A., Chiaraluce F., et al.* LEDAcrypt: Low-dEnsity parity-check coDebAsed cryptographic systems. `https://www.ledacrypt.org/documents/LEDAcrypt_v3.pdf`, 2019.
26. *Prange E.* The use of information sets in decoding cyclic codes. IRE Trans. Inform. Theory, 1962, vol. 8, no. 5, pp. 5–9.
27. *Both L. and May A.* Decoding linear codes with high error rate and its impact for LPN security. LNCS, 2018, vol. 10786, 2018, pp. 25–46.
28. *May A. and Ozerov I.*, On computing nearest neighbors with applications to decoding of binary linear codes. LNCS, 2015, vol. 9056, pp. 203–228.
29. *Kudekar S., Kumar S., Mondelli M., et al.* Reed — Muller codes achieve capacity on erasure channels. Proc. STOC'16, N.Y., ACM, 2016, pp. 658–669.
30. *Abbe E. and Sandon C.* A proof that Reed — Muller codes achieve Shannon capacity on symmetric channels. IEEE 64th Ann. Symp. FOCS, Santa Cruz, CA, USA, 2023, pp. 177–193.
31. *Reed I. S.* A class of multiple-error-correcting codes and the decoding scheme. IEEE Trans. Inform. Theory, 1954, vol. 4, no. 4, pp. 38–492.
32. *Massey J. L.* Threshold Decoding. Cambridge, MA, MIT Press, 1963.
33. *Tsimmerman K.-Kh.* Metody teorii modulyarnykh predstavleniy v algebraicheskoy teorii kodirovaniya [Methods of the Modular Representations Theory in Algebraic Coding Theory]. Moscow, MCCME, 2011. (in Russian)
34. *Sidel'nikov V. M. and Pershakov A. S.* Decoding of Reed — Muller codes with a large number of errors. Problems Inform. Transmission, 1992, vol. 28, no. 3, pp. 269–281.

35. *Kasami T. and Lin S.* On the construction of a class of majority-logic decodable codes. IEEE Trans. Inform. Theory, 1971, vol. 17, no. 5, pp. 600–610.
36. *Sidel'nikov V. M.* Open coding based on Reed — Muller binary codes. Discrete Math. Appl., 1994, vol. 4, no. 3, pp. 191–207.
37. *Kabatiansky G. and Tavernier C.* A new code-based cryptosystem via pseudorepetition of codes. 16th Intern. Workshop Algebraic Combinat. Coding Theory, Svetlogorsk, Russia, 2018, pp. 189–191.
38. *Egorova E., Kabatiansky G., Krouk E., and Tavernier C.* A new code-based public-key cryptosystem resistant to quantum computer attacks. J. Phys. Conf. Ser., 2019, vol. 1163, pp. 1–5.
39. *Minder L. and Shokrollahi A.* Cryptanalysis of the Sidelnikov cryptosystem. LNCS, 2007, vol. 4515, pp. 347–360.
40. *Borodin M. A. and Chizhov I. V.* Effective attack on the McEliece cryptosystem based on Reed — Muller codes. Discrete Math. Appl., 2014, vol. 24, no. 5, pp. 273–280.
41. *Vysotskaya V. V* Kvadrat koda Rida-Mallera i klassy jekvivalentnosti sekretnyh kljuchej kriptosistemy Mak-Jelisa-Sidel'nikova [Reed-Muller code square and equivalence classes of secret keys of the McEliece-Sidelnikov cryptosystem]. Prikladnaja diskretnaja matematika. Prilozhenie, 2017, no. 10. pp. 66–68. (in Russian)
42. *Vysotskaya V. V.* O strukturnykh osobennostyakh prostranstva klyuchey kriptosistemy Mak-Elisa — Sidel'nikova na obobshchennykh kodakh Rida — Solomona [On the structural features of the key space of the McEliece — Sidelnikov cryptosystem based on generalized Reed — Solomon codes]. Diskretnaya Matematika, 2024, vol. 36, iss. 4, pp. 28–43. (in Russian)
43. *Davletshina A. M.* Poisk ekvivalentnykh klyuchey kriptosistemy Mak-Elisa — Sidel'nikova, postroennoy na dvoichnykh kodakh Rida — Mallera [Search for equivalent keys of the McEliece — Sidelnikov cryptosystem built on the Reed — Muller binary codes]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2019, no. 12, pp. 98–100. (in Russian)
44. *Chizhov I. V. and Borodin M. A.* Classification of Hadamard products of one-codimensional subcodes of Reed — Muller codes. Discrete Math. Appl., 2022, vol. 32, no. 5, pp. 297–311.

УДК 519.7 DOI 10.17223/20710410/67/2

# СОВРЕМЕННЫЕ ПАРАДИГМЫ ПОСТРОЕНИЯ СХЕМ ЦИФРОВОЙ ПОДПИСИ НА РЕШЁТКАХ

А. Г. Леевик*,**, Е. С. Малыгина*,***, Е. М. Мельничук*, Д. А. Набоков*

*ООО «КуАпп», г. Москва, Россия
**Университет ИТМО, г. Санкт-Петербург, Россия
***Международный научно-образовательный математический центр НГУ, Новосибирский государственный университет, г. Новосибирск, Россия

**E-mail:** anton.leevik@gmail.com, emalygina@qapp.tech, emelnichuk@qapp.tech, dnabokov@qapp.tech

Ввиду возникновения угрозы нарушения стойкости криптографических алгоритмов с помощью квантового вычислителя большую популярность обрело направление постквантовой криптографии — нового раздела, включающего алгоритмы и протоколы, эффективно противостоящие атакам с помощью квантового компьютера. Теория решёток на сегодняшний день является перспективным направлением постквантовой криптографии. Одними из первых криптосистем на решетках были GGH и NTRU. В их основе лежит задача о поиске ближайшего вектора решётки, а отличие схем заключается в построении решёток. На основе криптосистем GGH и NTRU была предложена схема цифровой подписи NTRUSign, взявшая лучшее у них. Другой подход к построению подписи появился в 2008 г., в его основе лежит парадигма Hash-and-Sign, а подпись для сообщения вырабатывается с помощью лазейки. Годом позже В. Любашевским был предложен ещё один подход к построению подписи на решетках. Он заключается в использовании преобразования Фиата — Шамира, однако в силу специфики решёток алгоритм формирования подписи выдаёт корректную подпись с некоторой вероятностью, поскольку в целях безопасности используется выборка отбраковки. В данной работе представлен обзор существующих парадигм построения схем цифровых подписей на решётках, а также криптографических схем, построенных на рассматриваемых парадигмах. Проведён сравнительный анализ схем, определены преимущества и недостатки каждого из подходов, определены наилучшие условия их применения.

**Ключевые слова:** *постквантовая криптография, теория решёток, цифровая подпись на базе решёток.*

# CURRENT PARADIGMS FOR CONSTRUCTION OF LATTICE-BASED DIGITAL SIGNATURE SCHEMES

A. G. Leevik*,**, E. S. Malygina*,***, E. M. Melnichuk*, D. A. Nabokov*

*QApp, Moscow, Russia
**ITMO University, Saint Petersburg, Russia
***Mathematical Center in Akademgorodok, Novosibirsk State University, Novosibirsk, Russia

With the advent of quantum computing, research into post-quantum cryptography has gained significant attention. This is a novel branch of cryptography that utilizes

algorithms and protocols designed to withstand attacks from quantum computers. Lattice theory represents a promising area within post-quantum cryptographic research. Two early examples of lattice-based cryptosystems are the GGH and NTRU schemes. These schemes are based on the challenge of finding the closest vector in a lattice and differ primarily in the type of lattice used. The NTRUSign protocol was developed by combining the strengths of both schemes. In 2008, another approach to lattice signatures was introduced by a group of authors. It is based on the hash-and-sign paradigm, in which a signature for a message is generated using a trapdoor. A year later, V. Lyubashevsky proposed another method for constructing lattice-based signatures that utilizes the Fiat — Shamir transform. However, due to the nature of the underlying lattice structure, the algorithm for signature generation produces a correct signature only with a certain probability. This is due to the use of a rejection sampling for security purposes. This paper presents an overview of existing lattice-based signature construction approaches and cryptographic schemes that are based on these approaches. A comparative analysis was conducted on these schemes, identifying the advantages and disadvantages of each method. Based on the results, optimal conditions for the application of each approach have been determined.

**Keywords:** *post-quantum cryptography, lattice theory, lattice-based signature.*

## Введение

С появлением квантовых компьютеров и разработкой алгоритма Шора многие современные криптографические схемы окажутся нестойкими [1], а именно схемы, в основе которых лежат задачи факторизации или дискретного логарифмирования. Эта угроза способствовала развитию нового криптографического направления — постквантовой криптографии, послужившего толчком к созданию криптографических алгоритмов, противостоящих атакам с использованием квантовых компьютеров. Среди многих постквантовых направлений криптография на базе решёток является одной из наиболее перспективных. Криптографические примитивы, в основе которых лежат решётки, могут быть использованы для создания цифровых подписей и шифрования с открытым ключом. Трудные задачи на решётках, например задача нахождения кратчайшего вектора (SVP) [2] или задача поиска ближайшего вектора (CVP) [3], считаются стойкими к квантовым атакам. Соответственно можно говорить о перспективе замены существующих схем цифровой подписи и шифрования, которые будут подвержены атакам с использованием квантовых компьютеров, с точки зрения стойкости и практичности. С практической точки зрения некоторые схемы шифрования с открытым ключом и схемы цифровой подписи, основанные на трудных решёточных задачах, в настоящее время более практичны, чем, например, традиционные схемы RSA [4].

Теоретическое направление, касающееся доказуемой стойкости схем, в основе которых лежат решётки, возникло в 1996 г. вместе с редукцией Айтая от наихудшего случая к среднему [5], что, в свою очередь, привело к рассмотрению устойчивых к коллизиям хеш-функций, которые так же трудно взломать, как решить некоторые задачи, касающиеся евклидовых решёток, в наихудшем случае. Задача Айтая в среднем случае теперь называется задачей о нахождении малых целочисленных решений (SIS). Ещё одним крупным достижением в этой области стало представление О. Регевом задачи обучения с ошибками (LWE) [6]. Отметим, что LWE в среднем сложна (к ней сводятся стандартные задачи на решётках в худшем случае) и достаточно гибка для использования в криптографии.

Основной недостаток криптографических схем, основанных на задачах LWE и SIS, заключается в их ограниченной эффективности. Ключ обычно содержит случайную матрицу над кольцом $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ для небольшого $q$, размер которого линеен относительно параметра стойкости. При этом размерность векторного пространства решётки должна составлять, по меньшей мере, квадрат относительно параметра стойкости. В 2002 г. Д. Миччианчо адаптировал задачу SIS к структурированным матрицам с сохранением редукции от наихудшего случая к среднему [7]. Наихудший случай — это сужение стандартной задачи на решётках к задаче на конкретном семействе циклических решёток. Структура матриц Миччианчо допускает интерпретацию в терминах арифметики кольца $\mathbb{Z}_q[x]/(x^n - 1)$, где $n$ — размерность в наихудшем случае, а $q$ — малое простое число. Конструкция Миччианчо приводит к семейству хеш-функций, стойких к вычислению прообраза, с квазилинейной сложностью относительно параметра $n$. Наибольший выигрыш обусловлен использованием дискретного преобразования Фурье для умножения многочленов. В работах [8, 9] предложено заменить кольцо на $\mathbb{Z}_q[x]/\Phi$, где многочлен $\Phi$ является неприводимым над рациональными числами и имеет малые коэффициенты (например, $\Phi = x^n + 1$, где $n$ — степень двойки). Полученная в результате хеш-функция оказалась стойкой к нахождению коллизий относительно предполагаемой сложности модифицированной задачи усреднения, которую в настоящее время часто называют задачей нахождения малых целочисленных решений над кольцом (R-SIS). В 2009 г. В. Любашевский [10] представил эффективную цифровую подпись, доказуемо такую же надёжную, как R-SIS (в модели случайного оракула). Чуть позже в [11] был предложен вариант задачи LWE, рассматриваемый в кольцах и названный R-LWE, большая гибкость которого обеспечивает более естественные и эффективные криптографические конструкции.

На сегодняшний день криптография на основе решёток стала доступной в качестве будущей альтернативы теоретико-числовой криптографии. Следует отметить, что использование специального класса решёток, например идеальных, даёт существенное ускорение и уменьшение размеров ключей для большинства криптопротоколов, а сложные решёточные задачи, адаптированные под такой класс решёток, остаются по-прежнему трудными [11, 12].

Настоящая работа посвящена обзору существующих парадигм, на базе которых строятся цифровые подписи на решётках, а также обзору самых первых цифровых подписей и одних из последних, являющихся наиболее привлекательными с точки зрения параметров и стойкости, построенных на этих парадигмах.

Структура работы следующая. В п. 1 приводятся основные обозначения и базовые определения, связанные с задачами на решетках. Пункты 2–4 посвящены описанию основных парадигм, на основе которых строятся цифровые подписи на решётках, а именно: рассматриваются парадигмы GGH/NTRUSign, Hash-and-Sign и Фиата — Шамира и для каждой парадигмы наиболее значимые цифровые подписи. В п. 5 обсуждаются выбор параметров и оценка стойкости для решёточных схем.

## 1. Предварительные сведения

### 1.1. Обозначения

Матрицы обозначаются заглавными буквами жирным шрифтом (например, $\mathbf{A}$), векторы — строчными буквами жирным шрифтом (например, $\mathbf{v}$). Векторы являются вектор-столбцами.

Для целого (обычно простого) $q$ пусть $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ — кольцо целых чисел по модулю $q$, где элементы представлены числами из диапазона $\{-\lfloor (q-1)/2 \rfloor, \ldots, \lfloor (q-1)/2 \rfloor\}$.

Пусть $\mathcal{R} = \mathbb{Z}[x]/(\varphi(x))$ — кольцо многочленов для произвольного целого $n$ по модулю многочлена $\varphi(x)$, где $\varphi(x)$ известен из контекста; $\mathcal{R}_{\mathbb{R}} = \mathbb{R}[x]/(\varphi(x))$ — кольцо многочленов над действительными числами; $\mathcal{R}_{\mathbb{Q}} = \mathbb{Q}[x]/(\varphi(x))$ — кольцо многочленов над рациональными числами; $\mathcal{R}_q = \mathbb{Z}_q[x]/(\varphi(x))$. Для $a \in \mathbb{R}$ символ $\lfloor a \rceil$ обозначает округление до ближайшего целого, причём $\lfloor 1/2 \rceil = 1$. Символ естественным образом расширяется для матриц, векторов и многочленов, округляя их каждый коэффициент.

Для целого $\eta$ обозначим $S_\eta$ — множество многочленов степени меньше $n$ (где $n$ определено из контекста) с коэффициентами из $[-\eta, \eta] \cap \mathbb{Z}$. Для заданного вектора многочленов $\mathbf{y} = \left( \sum\limits_{0 \leqslant i < n} y_i x^i, \ldots, \sum\limits_{0 \leqslant i < n} y_{nk-n+i} x^i \right)^{\mathrm{T}} \in \mathcal{R}^k$ определим его $\ell_2$-норму как $\|\mathbf{y}\|_2 = \|(y_0, \ldots, y_{nk-1})^{\mathrm{T}}\|_2$. Аналогичным образом определяются $\ell_1$- и $\ell_\infty$-нормы. Гипершаром с центром в $\mathbf{c} \in \mathcal{R}^m$ радиуса $r > 0$ и размерности $m > 0$ называется множество $\mathscr{B}_{\mathcal{R},m}(r, \mathbf{c}) = \left\{ \mathbf{x} \in \mathcal{R}_{\mathbb{R}}^m : \|\mathbf{x} - \mathbf{c}\|_2 \leqslant r \right\}$.

Для распределения $U$ под записью $a \leftarrow U$ будем понимать, что элемент $a$ принимает значение в соответствии с распределением $U$. В записи $a \leftarrow \mathcal{I}$ элемент $a$ принимает случайное значение из множества $\mathcal{I}$.

**Определение 1.** Схема цифровой подписи состоит из трёх полиномиальных алгоритмов $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, где вероятностный алгоритм $\mathsf{Gen}(1^\lambda)$ вырабатывает пару ключей $(\mathsf{sk}, \mathsf{vk})$ для параметра стойкости $\lambda$ (который равен отрицательному двоичному логарифму вероятности успеха наилучшей известной атаки на схему); вероятностный алгоритм $\mathsf{Sign}(\mathsf{sk}, \mu)$ возвращает подпись $\sigma$ для сообщения $\mu$; детерминированный алгоритм $\mathsf{Verify}(\mathsf{vk}, \mu, \sigma)$ возвращает бит $b$.

Цифровая подпись является $\gamma$-корректной, если для любой пары ключей $(\mathsf{sk}, \mathsf{vk})$ и сообщения $\mu$ выполняется

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mu, \mathsf{Sign}(\mathsf{sk}, \mu)) = 1] \geqslant \gamma, \tag{1}$$

где вероятностное пространство задаёт алгоритм $\mathsf{Sign}$. Будем говорить, что подпись корректна в (Q)ROM модели, если неравенство (1) выполняется, когда вероятность также зависит от случайности в (квантовом) случайном оракуле, моделирующим хеш-функцию, используемую в схеме.

Дадим определение двум основным моделям безопасности для схем цифровой подписи, а именно: $\mathsf{UFCMA}$ (existential UnForgeability under Chosen Message Attacks) и $\mathsf{UFNMA}$ (existential UnForgeability under No-Message Attacks).

**Определение 2** ($\mathsf{UFCMA}$-модель). Пусть $T, \delta \geqslant 0$, тогда схема цифровой подписи $\mathrm{sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ является $(T, \delta)$-$\mathsf{UFCMA}$ стойкой в QROM-модели, если для любого квантового злоумышленника $\mathcal{A}$ с границей на время выполнения $\leqslant T$, которому предоставляется (классический) доступ к подписывающему оракулу и (квантовый) доступ к случайному оракулу $H$, выполняется

$$\mathsf{Adv}_{\mathsf{sig}}^{\mathsf{UFCMA}}(\mathcal{A}) = \Pr_{(\mathsf{vk},\mathsf{sk})}[\mathsf{Verify}(\mathsf{vk}, \mu^*, \sigma^*) = 1 \mid (\mu^*, \sigma^*) \leftarrow \mathcal{A}^{H, \mathsf{Sign}}(\mathsf{vk})] \leqslant \delta,$$

где вероятностное пространство задаётся случайными $\mathcal{A}$ и $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Злоумышленник не может вызвать оракул $\mathsf{Sign}$ для сообщения $\mu^*$.

Модель $\mathsf{UFNMA}$ определяется схожим образом, за исключением того, что злоумышленнику не разрешается вызывать подписывающий оракул для сообщений.

### 1.2. Трудные задачи на решётках

Поскольку в основе алгоритмов схемы цифровой подписи лежит дискретное гауссово распределение, дадим его определение.

**Определение 3.** Ненормализованное гауссово распределение со стандартным отклонением $\sigma \in \mathbb{R}$ и центром $c \in \mathbb{R}^n$, вычисленное в точке $x \in \mathbb{R}^n$, определяется как

$$\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right).$$

В случае, когда $\mathbf{c} = \mathbf{0}$, распределение обозначают $\rho_\sigma(\mathbf{x})$.

Дискретное гауссово распределение над $\mathbb{Z}^m$ с центром в точке $\mathbf{0}$ определяется как

$$D_\sigma^m(\mathbf{x}) = \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(\mathbb{Z}^m)},$$

где $\rho_\sigma(\mathbb{Z}^m) = \sum\limits_{\mathbf{x}\in\mathbb{Z}^m} \rho_\sigma(\mathbf{x})$.

Определим MLWE- и MSIS-гипотезы для алгебраических решёток, используемых в схемах цифровой подписи.

**Определение 4** (Decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$). Для положительных целых $q, k, \ell, \eta$ и размерности $n$ кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$, равно

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,\ell,\eta}(\mathcal{A}) = \big|\mathsf{Pr}[b=1|\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{b})] - \\ -\mathsf{Pr}[b=1|\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; (\mathbf{s}_1,\mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}_1+\mathbf{s}_2)]\big|.\end{aligned}$$

Запись «$s \leftarrow S$» означает, что величина $s$ выбрана в соответствии со случайным равномерным распределением на множестве $S$.

**Определение 5** (Search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$). Для положительных целых $q, k, \ell$, положительного действительного $\beta$ и размерности $n$ кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$, равно

$$\mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,\beta}(\mathcal{A}) = \mathsf{Pr}\left[0 < \|\mathbf{y}\|_2 < \beta \wedge (\mathbf{A}|\mathbf{Id})\cdot\mathbf{y} = 0 \bmod q | \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})\right].$$

Аналогичным образом можно определить LWE- и SIS-гипотезы. В этом случае кольцо $\mathcal{R}_q$ имеет размерность $n = 1$, то есть векторы и матрицы рассматриваются над $\mathbb{Z}_q$. Для гипотез Ring-LWE и Ring-SIS имеем $k = \ell = 1$.

Определим также менее стандартную задачу SelfTargetMSIS, которая, однако, часто используется при доказательстве стойкости схем цифровой подписи, основанных на алгебраических решётках.

**Определение 6** ($\mathsf{SelfTargetMSIS}_{H,n,q,k,\ell,\beta}$). Пусть $H : \{0,1\}^* \to \mathcal{R}_2$ — криптографическая хеш-функция. Для положительных целых $q, k, \ell$, положительного действительного $\beta$ и размерности $n$ кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Search-$\mathsf{SelfTargetMSIS}_{H,n,q,k,\ell,\beta}$, равно

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{H,n,q,k,\ell,\beta}(\mathcal{A}) = \\ &= \mathsf{Pr}\left[\begin{array}{c} 0 \leqslant \|\mathbf{y}\|_2 \leqslant \beta \\ \wedge H(\mu\|(\mathbf{Id}|\mathbf{A})\cdot\mathbf{y}) = c \end{array} \;\middle|\; \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \left(\mathbf{y} = \begin{pmatrix}\mathbf{r}\\ c\end{pmatrix}, \mu\right) \leftarrow \mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A})\right].\end{aligned}$$

Задачи MSIS и SelfTargetMSIS могут быть определены и для $\ell_\infty$-нормы.

## 2. Парадигма GGH/NTRUSign

Криптосистемы GGH [13] и NTRUEncrypt [14] — одни из первых, в основе которых лежат сложные задачи на решётках, в частности задача об аппроксимации ближайшего вектора. Разница между этими схемами заключается в том, что последнюю можно рассматривать как спецификацию первой. Криптосистема GGH включает схему цифровой подписи, что легло в основу NTRUSign [15], которая сохраняет почти весь дизайн GGH, но с применением NTRU-решёток, используемых в NTRUEncrypt. Предшественник схемы NTRUSign, NSS [16], был взломан К. Джентри и др. [17, 18]. NTRUSign постигла та же участь, атаки представлены в [19], где показано восстановление секретного ключа с экспериментами на 400 подписях. Поскольку модификация NTRUSign с учётом контрмер и версия с возмущениями также были взломаны [20], интерес к этой схеме был потерян из-за непрактичности её применения. Однако недавние исследования [21] дают некоторые надежды относительно будущего этой схемы.

### 2.1. Схема цифровой подписи NTRUSign

В основе схемы цифровой подписи лежит задача об аппроксимации ближайшего вектора в NTRU-решётке (APPR-CVP).

**Определение 7** (Задача APPR-CVP). Пусть $\mathcal{R} = \mathbb{Z}[x]/(\varphi(x))$, где $\varphi(x) = x^n - 1$; $\mathscr{L}$ — $\mathcal{R}$-модуль ранга $r$; $\mathbf{m} \in \mathcal{R}_{\mathbb{R}}^r$ — произвольный вектор. Вектор $\mathbf{v} \in \mathscr{L}$ называется решением задачи APPR-CVP, если $\|\mathbf{v} - \mathbf{x}\| < \mathcal{N}$ для некоторого малого $\mathcal{N} \in \mathbb{R}$.

На множестве $\mathcal{R} = \mathbb{Z}[x]/(x^n - 1)$ будем использовать операции сложения и свёрточного умножение (свёртку). Под свёрткой двух многочленов $f$ и $g$ будем понимать вычисление коэффициента при $x^k$ в $f * g$:

$$(f * g)_k = \sum_{i+j=k \,(\mathrm{mod}\, N)} f_i \cdot g_j, \quad 0 \leqslant k < N,$$

где $f_i$ и $g_j$ — коэффициенты $f$ и $g$ соответственно для $i, j = 1, \ldots, N$.

Рассмотрим $q \in \mathbb{Z}$, $h \in \mathcal{R}$ и обозначим $M_{h,q} = \{(u, v) \in \mathcal{R}^2 : v = u * h \mod q\}$. Это множество является решёткой размерности $2N$. Если $f$ и $g$ соответствуют бинарным многочленам из $\mathcal{R}$ и $d_f$, $d_g$ — количество отличных от нуля коэффициентов $f$ и $g$ соответственно, то нормы $\|f\|$, $\|g\|$ и $\|(f, g)\|$ определяются следующим образом:

$$\|f\| = \sqrt{d_f \left(1 - \frac{d_f}{N}\right)}, \quad \|g\| = \sqrt{d_g \left(1 - \frac{d_g}{N}\right)}, \quad \|(f, g)\| = \sqrt{\|f\|^2 + \|g\|^2}.$$

Генерация ключей. Зададим целые $N, q, B \geqslant 0$ и вектор $t$. Далее сгенерируем секретный и открытый базисы решётки. Для этого положим $i = B$ и пока $i \geqslant 0$:

- выберем случайным образом бинарные многочлены $f, g \in \mathcal{R}$ так, чтобы $d_f$ и $d_g$ были отличны от нуля;
- найдём малые $F, G \in \mathcal{R}$ (т. е. $\|F\|, \|G\| = \mathcal{O}(\sqrt{N})$), такие, что $f * G - F * g = q$;
- если $t$ имеет стандартную форму, то положим $f_i = f$ и $f'_i = F$. Если $t$ имеет траспонированную форму, то положим $f_i = f$ и $f'_i = g$. Также положим $h_i = f_i^{-1} * f'_i \mod q$ и $i = i - 1$.

Открытым ключом являются параметры $N, q, d_f, d_g, B$ и $h = h_0 = f_0^{-1} * f'_0 \mod q$, а секретным — множество $\{f_i, f'_i, h_i : i = 0, \ldots, B\}$.

Формирование подписи. Для подписания необходима хеш-функция $H : \mathcal{D} \to \mathcal{R}$ для некоторого множества $\mathcal{D}$. В действительности $H = H_2 \circ H_1$, где $H_1$ — стандартная безопасная хеш-функция, получающая на выходе $\beta$ бит $H_1(\mathcal{D})$. Функция $H_2 : \mathbb{Z}_2^\beta \to \mathbb{Z}_q^N$

задаёт дайджест сообщения $m = H_2(H_1(\mathcal{D}))$. Также нам потребуется функция нормы $\|\cdot\| : \mathcal{R}^2 \to \mathbb{R}$ и граница нормы $\mathcal{N} \in \mathbb{R}$. Для $(s,t) \in \mathcal{R}^2$ определим величину $\|(s \bmod q, r \bmod q)\|$, являющуюся минимальным значением среди $\|(s + k_1 q, r + k_2 q)\|$ для $k_1, k_2 \in \mathcal{R}$.

Процесс подписания сообщения $M \in \mathcal{D}$ с использованием секретного ключа $\{f_i, f_i', h_i : i = 0, \ldots, B\}$ выглядит следующим образом:

1) Положить $r = 0$.
2) Положить $s = 0$ и $i = B$. Представить $r$ в виде битовой строки, вычислить $m_0 = H(M\|r)$ и положить $m = m_0$.
3) Пока $i \geqslant 1$:
   — вычислить $x = \lfloor -(1/q)m * f_i' \rceil$, $y = \lfloor -(1/q)m * f_i \rceil$, $s_i = x * f_i + y * f_i'$;
   — вычислить $m = s_i * (h_i - h_{i-1}) \bmod q$;
   — положить $s = s + s_i$, $i = i - 1$.
4) Вычислить $x = \lfloor -(1/q)m * f_0' \rceil$, $y = \lfloor -(1/q)m * f_0 \rceil$, $s_0 = x * f_0 + y * f_0'$ и положить $s = s + s_0$.
5) Вычислить $b = \|(s, (s * h - m_0) \bmod q)\|$. Если $b \geqslant \mathcal{N}$, то положить $r = r + 1$ и перейти на шаг 2.

Подписью является тройка $(M, r, s)$.

Проверка. На вход подаются подпись $(M, r, s)$ и открытый ключ $h$. Величина $r$ кодируется в битовую строку и вычисляются значения $m = H(M\|r)$ и $b = \|(s, (s * h - m) \bmod q)\|$. Если $b < \mathcal{N}$, то подпись верна, в противном случае подпись отклоняется.

Безопасность. Восстановление ключа из публичной информации эквивалентно нахождению малых векторов в NTRU-решётке, что, в свою очередь, приводит к сложной задаче, на которой основана NTRUEncrypt. В транспонированной решётке ситуация для злоумышленника ещё сложнее, поскольку целевые малые векторы рассматриваются приближённо к гауссовой эвристике и, как следствие, на практике их сложно найти с помощью редукции решётки. Для подписи хорошо использовать редуцированный базис, однако нахождение такого базиса — трудная задача для больших $N$.

Рекомендованными параметрами для NTRUSign являются

$$(N, q, d_f, d_g, B, t, \mathcal{N}) = (251, 128, 73, 71, 1, \text{“transpose”}, 310);$$

время работы алгоритмов подписи указано в табл. 1.

Таблица 1

**Время, затраченное на генерацию ключей, подпись и проверку NTRUSign**

| Схема | Генерация ключей, мс | Подпись, мс | Проверка, мс |
|---|---|---|---|
| NTRUSign-251 | 180 000 | 500 | 300 |

Первые работы, касающиеся того, что схемы цифровой подписи GGH и NTRUSign могут быть нестойкими, представили К. Джентри и М. Шидло [17, 18], которые заметили, что при каждой подписи происходит утечка некоторой информации о секретном ключе. Однако, как продемонстрировали П. Нгуен и О. Регев несколькими годами позже [19], эта утечка информации действительно приводит к атаке на схему. Точнее, они показали, что при наличии достаточного количества пар «сообщение — подпись» можно восстановить закрытый ключ. Более того, их атака достаточно эффективна и была

реализована и применена к большинству вариантов параметров в GGH и NTRUSign; тем самым установлено, что эти схемы цифровой подписи не являются стойкими на практике.

Идея атаки довольно проста. Основное наблюдение состоит в том, что разность $(m-s)$, полученная из пары «сообщение — подпись» $(m,s)$, распределена равномерно в $\{\mathbf{Bx} : \mathbf{x} \in [-1/2, 1/2]^n\}$, где $\mathbf{B}$ — базис решётки. Следовательно, при достаточном количестве таких пар возникает следующая алгоритмическая задача, называемая задачей скрытого параллелепипеда: при множестве случайных точек, равномерно распределённых по неизвестному $n$-мерному параллелепипеду, необходимо восстановить этот параллелепипед или его приближение. Эффективное решение этой задачи составляет атаку.

Наилучшими мерами противодействия атаке являются методы возмущения [22, 23]. Они изменяют процесс генерации подписи таким образом, что скрытый параллелепипед заменяется значительно более сложным телом, что, по-видимому, предотвращает атаки описанного типа. Основной недостаток возмущений заключается в том, что они замедляют генерацию подписи и увеличивают размер секретного ключа. Однако NTRUSign с возмущениями не имеет никакого доказательства безопасности.

### 2.2. Модификации NTRUSign

Разработка схем цифровой подписи на решётках тесно связана с редукцией вектора по модулю фундаментального параллелепипеда секретного базиса [15]. Использование такого подхода привело к утечке секретной информации, а именно формы параллелепипеда [19]. NTRUSign — чрезвычайно эффективная схема, как следствие, интерес к разработке мер противодействия атакам возрастает, однако сами меры не имеют особого успеха [20].

*Схема цифровой подписи № 1*

В [21] представлен альтернативный подход к устранению утечки NTRUSign. Вместо модификаций используемых решёток и алгоритмов авторы рассматривают классическую негерметичную схему цифровой подписи NTRUSign и скрывают её с помощью гауссова шума, используя методы по аналогии со схемой цифровой подписи Любашевского.

Секретный ключ $\mathbf{S}$ представляется в виде матрицы из множества $\{-d, \ldots, d\}^{m\times k}$. Сообщение хешируется в вектор $\mathbf{c} \in \{-1, 0, 1\}^k$, такой, что $\|\mathbf{c}\|_1 \leqslant \kappa$, а подпись состоит из элемента $\mathbf{Sc}$, сдвинутого на маску $\mathbf{y} \leftarrow D_\sigma^m$, где $D_\sigma^m$ является дискретным гауссовым распределением размерности $m$ со стандартным отклонением $\sigma$.

Вместо того чтобы скрыть $\mathbf{Sc}$, авторы получают негерметичную схему цифровой подписи из NTRUSign, а затем используют эту подпись в качестве секретной и скрывают её с помощью корректно подобранного $\mathbf{y}$. Важный нюанс заключается в выборе размерности $m$ и стандартного отклонения $\sigma$: если $\sigma$ выбрано слишком маленьким, то секрет не будет скрыт должным образом, как следствие, представленная модификация не будет стойкой относительно атаки на NTRUSign. Если $\sigma$ слишком велико, то схема потеряет практическую эффективность.

В отличие от других доказуемо стойких схем цифровой подписи, например таких, как GPV [24] или [25], исходная схема цифровой подписи NTRU не претерпевает изменений, за исключением более консервативного выбора общедоступных параметров, и тем самым сохраняет присущий ей размер и вычислительную эффективность. Однако маскировка подписи обходится дорого. В табл. 2 приведены размеры предлагаемой подписи и ключей для различных уровней стойкости.

Подпись скрывается с помощью шумоподавления в предположении, что не существует никаких структурных атак на открытый ключ NTRU, поскольку за последнее десятилетие не опубликовано никаких известных структурных атак на NTRU-решётки и что такие сложные задачи, как SIS, решаются не проще, чем в случайных решётках.

Таблица 2

**Размеры ключей и подписи модифицированной версии NTRUSign**

| | Откр. ключ (биты) | Секр. ключ (биты) | Подпись (биты) |
|---|---|---|---|
| Уровень стойкости 100 | | | |
| Размер | 10 700 | 6 900 | 1 400 |
| Скорость | 12 700 | 6 900 | 1 400 |
| Уровень стойкости 112 | | | |
| Размер | 12 300 | 7 700 | 1 550 |
| Скорость | 14 600 | 7 700 | 1 550 |
| Уровень стойкости 128 | | | |
| Размер | 14 500 | 8 700 | 1 750 |
| Скорость | 17 100 | 8 700 | 1 750 |
| Уровень стойкости 160 | | | |
| Размер | 16 800 | 9 900 | 2 000 |
| Скорость | 19 800 | 9 900 | 2 000 |

В силу гибридности схемы можно выделить три типа потенциальных атак:

1. Атака на NTRU-решётку заключается в попытке найти секретный ключ $(f, g)$ только из открытого ключа $h = g * f^{-1}$. Однако эту задачу невозможно решить, поскольку отсутствует теоретическое доказательство неразрешимости этой атаки.
2. Попытка выявить исходную NTRU-подпись внутри построенной (скрытой) подписи, чтобы затем перейти к атаке по аналогии с [20]. Эту задачу можно решить с помощью методов, описанных в [26].
3. Если злоумышленнику удастся создать подделку за полиномиальное время, то эту подделку можно использовать для решения задачи SIS.

Стоит отметить, что некоторые атаки на NTRUSign могут снизить уровень стойкости схемы, а также то, что методы редукции решётки обречены на провал либо из-за того, что короткий вектор оказывается слишком длинным, либо из-за увеличения вычислительной сложности [20, 27–29].

*Схема цифровой подписи № 2*

Д. Штеле и Р. Штейнфельд получили доказуемо стойкую схему, очень близкую к NTRUSign (с теоретической точки зрения) [30]. Они модифицировали схему NTRUSign, чтобы сделать её доказуемо стойкой для использования в стандартной модели (соответственно в модели случайного оракула) относительно предполагаемой квантовой (соответственно классической) сложности стандартных задач на решётках в наихудшем случае, ограниченных семейством решеток в круговых полях. Авторы показали, как расширить секретный ключ схемы NTRUEncrypt до секретного ключа подписи. Стойкость схемы следует из уже доказанной сложности задач R-SIS и R-LWE.

Для модифицированной схемы NTRUSign была применена техника, описанная в [15]. Отметим, что заявленный подход является эвристическим. Например, мы выбираем секретный ключ для зашифрования и отклоняем выборку до тех пор, пока не будут выполняться некоторые свойства (в первую очередь, взаимная простота двух секретных многочленов над $\mathbb{Z}[x]/(x^n - 1)$). Однако как влияет использование такого

подхода на стойкость, детально не исследовано. Авторы показывают, что в модифицированной схеме вероятность отклонения достаточно далека от 1, если использовать дзета-функцию Дедекинда в круговых полях. Кроме того, стойкость схемы следует из сложности задачи R-SIS даже при использовании дополнительной отбраковки.

Модифицированная схема NTRUSign устойчива к атакам на классическом компьютере в модели случайного оракула (предполагающей наихудшую сложность стандартных решёточных задач для идеальных решёток). Из-за использования случайного оракула отсюда непосредственно не следует, остаётся ли это доказательство значимым в случае квантовых атак. Как указано в [31], следует быть крайне осторожным, используя случайный оракул в квантовой модели.

## 3. Парадигма Hash-and-Sign

Схемы цифровой подписи, основанные на парадигме Hash-and-Sign, берут свое начало из работы У. Диффи и М. Хеллмана [32]. Концепция построения следует критерию, согласно которому сообщение $M$ должно быть хешировано перед тем, как быть подписанным. Таким образом, для подписания сообщения сначала необходимо вычислить его хеш, чтобы получить величину $h = H(M)$, которая должна находиться в диапазоне функции лазейки $f$ (пример такой функции — функция RSA). Затем результат хеширования подписывается $\sigma = f^{-1}(h)$, а алгоритм проверки проверяет условие $f(\sigma) = H(M)$, чтобы выяснить, является ли $(\sigma, M)$ допустимой парой.

Такая теория стала основой для моделирования хеш-функций с помощью случайного оракула [33]. Если $f$ является перестановочной лазейкой, то схема экзистенциально устойчива к атаке на выбранное сообщение. Приложение решёток к такому типу схем на основе парадигмы Hash-and-Sign заключается в том, что короткий базис решётки мог бы обеспечить такую функцию лазейки. Это породило первое предложение — схему GPV цифровой подписи, основанную на сложных задачах на решётках [24]. Основными в схеме являются:

— построение функций лазейки со свойством, что каждое выходное значение имеет несколько прообразов;
— использование гауссовой выборки;
— использование модульных решёток.

Более поздняя схема Миччианчо и Пайкерта [34] также использует парадигму Hash-and-Sign, но с более эффективной лазейкой, нежели в GPV. Улучшения в алгоритме генерации ключей были внесены Дж. Олвеном и К. Пайкертом [35].

### 3.1. Схема цифровой подписи GPV

GPV — схема цифровой подписи на решётках, стойкость которой основана на сложности решения задачи SIS. В алгоритмах схемы цифровой подписи используется функция лазейки, дискретное гауссово распределение и модульные решётки.

Зададим целые значения $n$, $q$ (полиномиально зависящее от $n$), $m = \Omega(n \log q)$ и $s \geqslant \geqslant 2\sqrt{n \log q}$ — стандартное отклонение. В оригинальной работе [24] в качестве функции лазейки предложено использовать короткий базис решётки.

**Определение 8.** Для матрицы $\mathbf{A} \in \mathbb{Z}^{n \times m}$, определяющей решётку

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = 0 \pmod q\} \supseteq q\mathbb{Z}^m,$$

матрица $\mathbf{T} \in \mathbb{Z}^{n \times m}$ является лазейкой, если:

— $\mathbf{AT} = 0 \pmod q$;
— $\mathbf{T}$ имеет полный ранг над $\mathbb{Z}$;

— каждый столбец $\mathbf{t}_i \in \mathbb{Z}_q^m$ матрицы $\mathbf{T}$ является коротким.

Генерация ключей. На этапе генерации по заданным параметрам создаётся пара $(\mathsf{vk}, \mathsf{sk})$, где

— открытый ключ $\mathsf{vk}$ составляют описанная выше матрица $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ и такая функция $f_{\mathbf{A}} : \mathbb{Z}^m \to \mathbb{Z}^n$, что для $\mathbf{x} \in \mathbb{Z}^m$

$$\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \pmod{q};$$

— секретный ключ $\mathsf{sk} = \mathbf{T}$, являющийся коротким базисом решётки $\Lambda^{\perp}(A)$, выступает в роли лазейки для решения задачи $\mathsf{SIS}$ и используется для выбора $\mathbf{x}'$, такого, что $\mathbf{x}' = f_{\mathbf{A}}^{-1}(\mathbf{u})$ с вероятностью $e^{-\|\mathbf{x}'\|^2/\sigma^2}$.

Фактически алгоритм выработки открытого/секретного ключей заключается в генерации односторонней функции с лазейкой. Для этого используется алгоритм $\mathsf{TrapGen}(1^n)$, возвращающий пару $(\mathbf{A}, \mathbf{T})$, где $\mathbf{A}$ описывает одностороннюю функцию $f_{\mathbf{A}}$, а $\mathbf{T}$ — лазейка.

Формирование подписи. Для подписания сообщения необходима хеш-функция $H : \{0,1\}^\star \to \mathbb{Z}_q^n$. Перед подписанием сообщение $M$ хешируется:

$$\mathbf{y} = H(M) \in \mathbb{Z}_q^n.$$

Так как хеш-значение — это вектор, его можно записать в виде $\mathbf{y} = \mathbf{A}\mathbf{x}$. Далее используем короткий базис $\mathbf{T}$ и функцию $f_{\mathbf{A}}^{-1}$, чтобы получить прообраз, соответствующий полученному хеш-значению:

$$\mathbf{x}' = f_{\mathbf{A}}^{-1}(\mathbf{u}).$$

Для этого используется рандомизированный вариант алгоритма “nearest plane” [24] — алгоритм 1 ($\mathsf{SampleD}$).

**Алгоритм 1.** SampleD

**Вход:** $\mathbf{T}, \mathbf{s}, \mathbf{c}$.
**Выход:** Вектор $\mathbf{v}_0$.

$\mathbf{v}_n := \mathbf{0}$.
$\mathbf{c}_n := \mathbf{c}$.
**Для** $i = n, \ldots, 1$:
  $c'_i := \langle \mathbf{c}_i, \widetilde{\mathbf{t}_i} \rangle / \langle \widetilde{\mathbf{t}_i}, \widetilde{\mathbf{t}_i} \rangle \in \mathbb{R}$;
  $s'_i := s / \|\widetilde{\mathbf{t}_i}\|_2 > 0$;
  $z_i \sim D_{\mathbb{Z}, s'_i, c'_i}$;
  $\mathbf{c}_{i-1} := c_i - z_i \mathbf{b}_i$;
  $\mathbf{v}_{i-1} := \mathbf{v}_i + z_i \mathbf{b}_i$.
Вернуть $\mathbf{x}' := \mathbf{v}_0$.

Полученный таким образом вектор $\mathbf{x}'$ является подписью $\sigma_M$ сообщения $M$.

Проверка. Для проверки подписи достаточно вычислить хеш-значение $\mathbf{y} = H(M)$, убедиться, что подпись является коротким вектором, то есть $\|\sigma_M\|_2 \leqslant 6n \log q$, и проверить, что выполняется равенство $\mathbf{A} \cdot \sigma_M = \mathbf{y}$.

Безопасность GPV. Стойкость схемы GPV основывается на сложности задачи $\mathsf{SIS}$. В оригинальной работе [24] невозможность подделки подписи доказывается в модели EUF-CMA. Изложим суть доказательства.

Пусть алгоритм $\mathcal{A}$ позволяет подделать подпись с вероятностью $e = e(n)$. Построим алгоритм $\mathcal{B}$, решающий задачу $\mathsf{SIS}$ с вероятностью $p = e(n)(1 - 2^{-\Omega(n)})$, используя для этого алгоритм $\mathcal{A}$ и симулятор $\mathcal{S}$. Принимая на вход матрицу $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathcal{B}$ должен найти ненулевой вектор $\mathbf{z} \in \mathbb{Z}_q^m$, такой, что $||\mathbf{z}||_2 < \beta$ и $\mathbf{Az} = 0 \pmod q$. Поскольку алгоритм $\mathcal{A}$ требует взаимодействия с оракулом подписи GPV и случайным оракулом, алгоритм $\mathcal{B}$ использует симулятор $\mathcal{S}$, чтобы симулировать ответы на запросы $\mathcal{A}$, играя роль «реального» алгоритма. Для корректной работы алгоритма необходимо, чтобы распределение ответов симулятора $\mathcal{S}$ и реальных оракула подписи и случайного оракула было неотличимо.

В рамках модели безопасности алгоритм $\mathcal{A}$ в ходе работы может осуществлять два типа запросов: на хеш (то есть обращение к случайному оракулу) и на подпись (обращение к оракулу подписи). Симулятор $\mathcal{S}$ в зависимости от запроса поступает следующим образом:

— при обращении к случайному оракулу: на каждый запрос $H(M)$, если до этого такого запроса не было, выбирается $\mathbf{x}_M \leftarrow D_{\mathbb{Z}^m,\sigma}$, вычисляется $\mathbf{u}_M = \mathbf{Ax}_M \bmod q$ и возвращается $H(M) = \mathbf{u}_M$. При этом тройка элементов $(M, \mathbf{u}_M, \mathbf{x}_M)$ сохраняется в памяти для будущих запросов;

— при обращении к оракулу подписи: для каждого запроса на подпись $\mathsf{Sign}(M)$ мы предполагаем с высокой вероятностью, что предварительно был сделан запрос на подпись $H(M)$. Поэтому мы можем извлечь соответствующую тройку $(M, \mathbf{u}_M, \mathbf{x}_M)$ и вернуть значение $\mathbf{x}_M$ в качестве подписи.

После коммуникации с симулятором $\mathcal{S}$ алгоритм $\mathcal{A}$ с вероятностью успеха $e$ возвращает подделку $(M^\star, \mathbf{x}^\star)$ для сообщения $M^\star$, для которого не было сделано запросов на подпись. После получения подделки алгоритм $\mathcal{B}$ делает запрос к оракулу подписи симулятора $\mathcal{S}$ для сообщения $M^\star$ и получает значение $\mathbf{x}_{M^\star}$. Таким образом, получаем равенство

$$H(M^\star) = \mathbf{Ax}^\star = \mathbf{Ax}_{M^\star}.$$

При этом так как $\Pr[\mathbf{x}^\star = \mathbf{x}_{M^\star}] \leqslant 2^{-\Omega(n)}$ [36], то $(\mathbf{x}^\star - \mathbf{x}_{M^\star})$ — решение задачи $\mathsf{SIS}$ с вероятностью $1 - 2^{-\Omega(n)}$.

### 3.2. Схема цифровой подписи Falcon

Falcon — схема цифровой подписи, использующая парадигму Hash-and-Sign на базе NTRU-решёток. Falcon использует преимущества NTRUSign и GPV, за счёт чего удаётся сделать размеры ключей и подписи компактными, сохраняя доказуемую стойкость схемы.

В схеме Falcon в качестве модуля используется число $q = 12\,289$. Зададим целое $n = 2^k$ и круговой многочлен $\phi(x) = x^n + 1$, задающий кольцо $\mathcal{R} = \mathbb{Z}[x]/(\phi(x))$. При этом $\phi$ полностью раскладывается в $\mathbb{Z}_q[x]$. Определим границу $\lfloor \beta^2 \rfloor > 0$ и стандартное отклонение $\sigma$. Кроме того, зададим промежуток $[\sigma_{\min}, \sigma_{\max}]$, в котором будут варьироваться все значения $\sigma'$ при выборке целого числа $z \sim D_{\mathbb{Z},\sigma',\mu}$ с помощью подпроцедуры $\mathsf{SamplerZ}$ алгоритма $\mathsf{ffSampling}$. Байтовую длину подписи обозначим через $\mathsf{sbytelen}$.

Генерация ключей (алгоритм 2). Процесс выработки ключевой пары в схеме цифровой подписи Falcon состоит из двух частей:

1) Решение NTRU-уравнения: ищутся многочлены $f, g, F, G \in \mathcal{R}$, удовлетворяющие NTRU-уравнению

$$fG - gF = q \pmod \phi.$$

2) Построение бинарного дерева Falcon $T$, которое удовлетворяет следующим условиям:
   — значением корня $T$.value дерева $T$ высоты $k$ является многочлен $l \in \mathcal{R}_{\mathbb{Q}}$, где $n = 2^k$;
   — левый и правый потомки дерева $T$, обозначаемые как $T$.left и $T$.right, являются бинарными деревьями Falcon высоты $k - 1$.

Для построения дерева $T$ используется $\mathbf{LDL}^\star$-разложение матриц, где $\mathbf{L} \in \mathcal{R}_{\mathbb{Q}}^{n \times n}$ — нижняя треугольная матрица с единичными элементами на диагонали; $\mathbf{L}^\star$ — матрица, эрмитово сопряжённая матрице $\mathbf{L}$; $\mathbf{D} \in \mathcal{R}_{\mathbb{Q}}^{n \times n}$ — диагональная матрица.

Непосредственно нам понадобится $\mathbf{LDL}^\star$-разложение матрицы $\mathbf{G} = \mathbf{BB}^\star$, где

$$\mathbf{B} = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix};$$

$\mathbf{B}^\star$ — матрица, эрмитово сопряжённая матрице $\mathbf{B}$. В алгоритме генерации ключей схемы цифровой подписи Falcon для построения $T$ используется функция ffLDL$^\star$, которая принимает на вход матрицу $\mathbf{G}$.

Алгоритм ffLDL$^\star$ построения дерева $T$ работает рекурсивно и состоит из двух частей:

1) на первом этапе вычисляется $\mathbf{LDL}^\star$-разложение матрицы $\mathbf{G} = \mathbf{BB}^\star$, которая подаётся на вход в алгоритм в FFT-представлении, то есть каждый элемент матрицы $\mathbf{G}$ представлен FFT-формой:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ L_{21} & 1 \end{bmatrix} \cdot \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ L_{21}^\star & 1 \end{bmatrix},$$

   где $L_{ij}, D_{ij} \in \mathcal{R}_{\mathbb{Q}}$. Значение $L_{21}$ сохраняется в корне дерева $T$.value;
2) на втором этапе определяются значения левого и правого потомков корня $T$.value из элементов $D_{11}, D_{22} \in \mathbb{Q}[x]/(x^n + 1)$ матрицы $\mathbf{D}$. Равенство $D_{ii} = d_{i1}(x^2) + x d_{i2}(x^2)$, $i \in 1, 2$, позволяет задать функцию

$$\mathsf{splitfft}(D_{ii}) = (d_{i1}, d_{i2}) \in \left(\mathbb{Q}[x]/(x^{n/2} + 1)\right)^2.$$

   Из полученных элементов $d_{ij}$ строятся две новые матрицы:

$$\mathbf{G}_{\text{left}} = \begin{bmatrix} d_{11} & d_{12} \\ d_{12}^\star & d_{11} \end{bmatrix}, \quad \mathbf{G}_{\text{right}} = \begin{bmatrix} d_{21} & d_{22} \\ d_{22}^\star & d_{21} \end{bmatrix}.$$

   Для матрицы $\mathbf{G}_{\text{left}}$ вызывается алгоритм ffLDL$^\star$, чтобы определить значение $T$.left и всех его потомков по рекурсии. Аналогично матрица $\mathbf{G}_{\text{right}}$ используется для вычисления значения $T$.right и всех его потомков.

Заметим, что алгоритм ffLDL$^\star$ строит ненормализованное дерево $T$. Нормализация относительно $\sigma$ происходит на шагах 6–7 алгоритма генерации ключей.

Формирование подписи (алгоритм 3). Для подписания вычисляется хеш-значение $c \in \mathcal{R}_q$ от сообщения $\mathbf{m}$, дополненного случайно сгенерированной солью $\mathbf{r} \in \{0,1\}^{320}$. В схеме Falcon для этого используется функция HashToPoint, принимающая на вход $\mathbf{r}||\mathbf{m}$, а также значения $q$ и $n$. Следующим шагом с помощью алгоритма семплирования ffSampling [37], реализующего выборку с использованием лазейки “fast Fourier nearest plane” [38], находим значения $s_1, s_2 \in \mathcal{R}$, такие, что $s_1 + s_2 h = c \pmod q$. “Fast Fourier

**Алгоритм 2.** Выработка ключей Falcon

**Вход:** $\phi \in \mathbb{Z}[x]$, $q$.
**Выход:** $(sk, vk)$.
1: $f, g, F, G := \mathsf{NTRUGen}(\phi, q)$.
2: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$.
3: $\hat{\mathbf{B}} := \begin{bmatrix} \mathsf{FFT}(g) & -\mathsf{FFT}(f) \\ \mathsf{FFT}(G) & -\mathsf{FFT}(F) \end{bmatrix}$.
4: $\mathbf{G} := \hat{\mathbf{B}} \times \hat{\mathbf{B}}^{\star}$.
5: $T := \mathsf{ffLDL}^{\star}(\mathbf{G})$.
6: **Для** $leaf \in T$:
7: $\quad leaf.\text{value} := \sigma / \sqrt{leaf.\text{value}}$
8: $sk := (\hat{\mathbf{B}}, \mathbf{T})$.
9: $h := gf^{-1} \bmod q$.
10: $pk := h$.

nearest plane" представляет собой рекурсивный вариант алгоритма семплирования, предложенного П. Клейном [39] и использованного в оригинальной схеме цифровой подписи GPV [24].

Однако, в отличие от алгоритма выборки Клейна, алгоритм, предложенный в схеме цифровой подписи Falcon в качестве лазейки, вместо одной матрицы $\mathbf{L}$ использует дерево $T$, состоящее из набора таких матриц.

На последнем шаге алгоритма формирования подписи $s_2$ сжимается в битовую строку $\mathbf{s}$ посредством функции $\mathsf{Compress}$ и подписью становится пара $(\mathbf{r}, \mathbf{s})$.

**Алгоритм 3.** Формирование подписи Falcon

**Вход:** сообщение $\mathbf{m}$, закрытый ключ $\mathsf{sk}$, ограничение $\lfloor \beta^2 \rfloor$.
**Выход:** подпись $\omega$.
1: $\mathbf{r} \xleftarrow{\$} \{0,1\}^{320}$.
2: $c := \mathsf{HashToPoint}(\mathbf{r}||\mathbf{m}, q, n)$.
3: $\mathbf{t} := \left( -\frac{1}{q}\mathsf{FFT}(c) \cdot \mathsf{FFT}(F), \frac{1}{q}\mathsf{FFT}(c) \cdot \mathsf{FFT}(f) \right)$.
4: **Выполнить**
5: $\quad$ **Выполнить**
6: $\quad\quad \mathbf{z} := \mathsf{fftSampling}(\mathbf{t}, \mathbf{T})$,
7: $\quad\quad \mathbf{s} := (\mathbf{t} - \mathbf{z})\hat{\mathbf{B}}$
8: $\quad$ **до тех пор пока** $||\mathbf{s}||_2^2 > \lfloor \beta^2 \rfloor$;
9: $\quad (s_1, s_2) := \mathsf{invFFT}(\mathbf{s})$,
10: $\quad \mathbf{s} := \mathsf{Compress}(s_2, 8 \cdot \mathsf{sbytelen} - 328)$
11: **до тех пор пока** $(\mathbf{s} = \perp)$.
12: **Вернуть** $\omega := (\mathbf{r}, \mathbf{s})$.

Проверка (алгоритм 4). Чтобы проверить подпись $\omega = (\mathbf{r}, \mathbf{s})$, проверяющая сторона начинает с вычисления хеш-значения $c \in \mathcal{R}_q$, используя $\mathbf{r}||\mathbf{m}$ и алгоритм $\mathsf{HashtoPoint}$. После этого к $\mathbf{s}$ применяется функция $\mathsf{Decompress}$ для получения значения $s_2 \in \mathcal{R}_q$,

что, в свою очередь, позволяет вычислить $s_1 = c - s_2 h \mod q$. На последнем этапе проверяется условие $||(s_1, s_2)||_2 \leqslant \lfloor \beta^2 \rfloor$.

Безопасность. Как уже отмечалось, стойкость схемы GPV, лежащей в основе цифровой подписи Falcon, базируется на сложности задачи SIS в модели ROM [24]. В схеме Falcon используются NTRU-решётки, поэтому доказательство стойкости, представленное для схемы GPV, адаптируется для NTRU-решёток. Фактически злоумышленнику $\mathcal{A}$ для подделки подписи некоторого сообщения $M^\star$ при заданных $y \in \mathcal{R}$ и матрицы $\mathbf{A} = \begin{bmatrix} 1 & h \end{bmatrix}$, являющейся открытым ключом, необходимо найти такие короткие $s_1, s_2 \in \mathcal{R}$, что выполняется

$$\mathbf{As} = \begin{bmatrix} 1 & h \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = s_1 + s_2 h = y \pmod q.$$

**Алгоритм 4.** Проверка подписи Falcon

**Вход:** сообщение $\mathbf{m}$, подпись $\omega = (\mathbf{r}, \mathbf{s})$, открытый ключ pk, ограничение $\lfloor \beta^2 \rfloor$.
**Выход:** принять или отклонить подпись.

1: $c := \mathsf{HashToPoint}(\mathbf{r}||\mathbf{m}, q, n)$.
2: $s_2 := \mathsf{Decompress}(\mathbf{s}, 8 \cdot \mathsf{sbytelen} - 328)$.
3: **Если** $s_2 = \perp$, **то**
4: отклонить подпись.
5: $s_1 := c - s_2 h \mod q$.
6: **Если** $||(s_1, s_2)||_2^2 \leqslant \lfloor \beta^2 \rfloor$, **то**
7: принять подпись,
8: **иначе**
9: отклонить подпись.

Помимо подделки, одно из направлений возможных атак на схему цифровой подписи Falcon связано с восстановлением секретного ключа. Для проведения данной атаки применяются алгоритмы редукции решёток, в частности алгоритм DBKZ [40], а определение параметров, при которых схема соответствует уровням стойкости, осуществляется с помощью методологии coreSVP hardness (см. далее п. 5). Параметры для обеспечения стойкости схемы Falcon для разных уровней представлены в табл. 3.

Таблица 3

**Параметры для цифровой подписи Falcon**

| Схема | Уровень стойкости (NIST) | $n$ | $q$ | $\sigma$ | $\sigma_{\min}$ | $\sigma_{\max}$ | $\lfloor \beta^2 \rfloor$ |
|---|---|---|---|---|---|---|---|
| Falcon-512 | I | 512 | 12289 | 165,736617183 | 1,277833697 | 1,8205 | 34034726 |
| Falcon-1024 | V | 1024 | 12289 | 168,388571447 | 1,298280334 | 1,8205 | 70265242 |

## 4. Парадигма Фиата — Шамира

Альтернативный способ построения схемы цифровой подписи заключается в том, чтобы сначала построить некоторый протокол аутентификации сторон, а затем преобразовать его в схему цифровой подписи с помощью преобразования Фиата — Шамира [41, 42]. Одной из известных схем классической криптографии, построенной на таком подходе, является схема Шнорра [43]. Перенос схемы Шнорра на аппарат теории решёток, а также её адаптация к сложным задачам на решётках были осуществлены впервые В. Любашевским в работах [10, 26, 44].

В [10] за основу взята задача о коротком целочисленном решении SIS. Первоначально протокол аутентификации сторон строится на основе решётки, в которой вызов (challenge) рассматривается как многочлен некоторого кольца $\mathcal{R}$. Стойкость протокола основана на сложности нахождения аппроксимированного кратчайшего вектора в стандартной модели, а также в модели случайного оракула.

Затем протокол аутентификации сторон преобразуется в схему цифровой подписи путём использования хеш-функции $h$ из стойкого к коллизиям семейства хеш-функций $\mathcal{H}$ для выработки вызова на стороне подписывающего. В схеме цифровой подписи также оптимизируются параметры протокола. Стойкость схемы цифровой подписи зависит от сложности поиска коллизий в определённых семействах хеш-функций. Злоумышленник, способный подделать подпись, может затем использовать это для поиска коллизии хеш-функции, выбранной случайным образом из $\mathcal{H}$. Следовательно, подделка подписи и нахождение коллизии случайно выбранной $h \in \mathcal{H}$ эквивалентно нахождению коротких векторов в решётке над кольцом $\mathcal{R}$.

Разница со схемой Шнорра заключается в том, что для сохранения сложности задачи SIS секретные векторы необходимо брать малыми по норме, из чего следует, что и нормы векторов подписи также малы. Из этого можно сделать вывод, что, в отличие от схемы Шнорра, где распределение значения подписи равновероятно по всему $\mathbb{Z}_q$, вектор подписи при переносе на решётки не распределён равномерно по всему пространству $\mathbb{Z}_q^n$, а подчиняется некоему закону распределения $\mathcal{Q}$. Для обеспечения стойкости схемы цифровой подписи требуется, чтобы распределение выходных векторов не зависело от секретных значений, поэтому при выработке вектора подписи из исходного распределения $\mathcal{Q}$ с некоторой вероятностью он отклоняется и процесс выработки начинается сначала. Отказ происходит таким образом, чтобы распределение выходных векторов подписи не зависело от секретного значения. Такая парадигма получила название «парадигма Фиата — Шамира с прерываниями», а алгоритм выработки элементов из заданного распределения называется алгоритмом выработки отбраковки.

В [26] В. Любашевским предложены модификации его ранних работ. Наиболее значительным изменением является адаптация задачи Ring-SIS к задаче Ring-LWE, что существенно уменьшает размеры подписи и ключей, тем самым повышая её эффективность. Второе улучшение касается алгоритма формирования подписи, который включает асимптотически более короткие подписи. Здесь требуется более сложная выборка отбраковки, чтобы подпись была независима от секретного ключа, а также выборка из нормального распределения, где необходимы высокоточные вычисления. В работе доказано, что схема является устойчивой относительно известных атак, её стойкость основана на сложности нахождения коротких векторов в решётке в наихудшем случае.

На парадигме Фиата — Шамира с прерываниями построена схема цифровой подписи Dilithium [45], которая выбрана Национальным институтом стандартов и технологий США в качестве нового стандарта цифровой подписи [46]. В новом конкурсе на выбор дополнительной схемы цифровой подписи также представлены схемы на решётках, построенные на данной парадигме, среди них выделяется схема HAETAE [47], в которой авторы смогли уменьшить размер подписи по сравнению с Dilithium на 30 %.

### 4.1. Схема цифровой подписи Любашевского

Опишем общий подход к построению схемы цифровой подписи, а также рассмотрим различные варианты, предложенные самим Любашевским либо другими авторами.

Пусть $q$ — простое число, задающее поле вычетов $\mathbb{Z}_q$. Несмотря на то, что некоторые варианты схем Любашевского построены в кольцах многочленов $\mathcal{R}_q = \mathbb{Z}_q[x]/(p(x))$,

где $p(x)$ — многочлен степени $n$, существует биективное отображение $\sigma : \mathcal{R}_q \to \mathbb{Z}_q^n$, таким образом, будем рассматривать построение схемы на целочисленных решётках как общий случай. Пусть $k, l, m$ — размерности матриц и векторов, зависящие от уровня стойкости схемы; $\mathcal{Q}, \mathcal{C}$ — распределения маскирующего вектора $\mathbf{y}$ и секретного ключа $\mathbf{s}$ соответственно; $\mathcal{P}$ — целевое распределение вектора $\mathbf{z}$ на выходе алгоритма.

Генерация ключей. Первоначально выбирается открытая матрица $\mathbf{A} \in \mathbb{Z}_q^{k \times l}$, являющаяся общей для всех пользователей системы (может быть вычислена предварительно). Секретным ключом является матрица $\mathbf{S} \leftarrow \mathcal{C}^{l \times m}$; открытым — значение $\mathbf{T} = \mathbf{AS} \bmod q \in \mathbb{Z}_q^{k \times m}$.

Формирование подписи. Сначала вырабатывается маскирующий вектор $\mathbf{y}$ из распределения $\mathcal{Q}^l$. Далее для сообщения $\mu$ подписывающий вычисляет хеш, используя стойкую к коллизиям хеш-функцию $H$, получает значение $\mathbf{c} = H(\mathbf{Ay} \bmod q, \mu) \in \mathbb{Z}_q^m$ и вычисляет потенциальный вектор подписи $\mathbf{z} = \mathbf{Sc} + \mathbf{y} \in \mathbb{Z}_q^l$. Поскольку вектор $\mathbf{y}$ выбран из распределения $\mathcal{Q}$, а значение $\mathbf{S}$ не меняется при генерации новой подписи, можно заметить, что распределение вектора $\mathbf{z}$ равно распределению $\mathcal{Q}$, сдвинутому на $\mathbf{Sc}$ (будем обозначать как $\mathcal{Q}_{+\mathbf{Sc}}$). Чтобы выходное распределение вектора $\mathbf{z}$ было равно $\mathcal{P}$, применим процедуру выборки отбраковки [48], в результате которой в качестве подписи возвратим значение $(\mathbf{z}, \mathbf{c})$.

Выборка отбраковки. Пусть даны две функции плотности распределения вероятностей $p_s$ и $p_t$; выборка отбраковки — это способ получения элементов из $p_t$ при наличии доступа к элементам из $p_s$. Для $p_s$ и $p_t$ существует такое значение $M \in \mathbb{R}$, что для любого $x$ имеет место $p_t(x) \leqslant M \cdot p_s(x)$. Тогда если значение $z$ получено из $p_s$ и принято с вероятностью $\dfrac{p_t(z)}{M \cdot p_s(z)}$, то распределение принятого $z$ в точности равно $p_t$, а среднее количество прерываний для генерации $z$ равно $M$. В рамках алгоритма формирования подписи вектор $\mathbf{z}$ принимается с вероятностью $\dfrac{\mathcal{P}(\mathbf{z})}{M \cdot \mathcal{Q}_{+\mathbf{Sc}}(\mathbf{z})}$.

Проверка. На входе пара $(\mathbf{z}, \mathbf{c})$ и открытый ключ $\mathbf{T}$. Сначала проверяется, что вектор $\mathbf{z}$ имеет малую норму, то есть $\|\mathbf{z}\| \leqslant \beta$ (в зависимости от спецификации схемы может использоваться как $\ell_2$-, так и $\ell_\infty$-норма) для $\beta$, выбранного в зависимости от распределения $\mathcal{P}$. Если эта проверка пройдена, то проверяется условие $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$. Если равенство выполнено, то подпись принимается.

Безопасность. Стойкость схемы Любашевского основывается на сложности задач SIS и LWE. Защищённость подписи от подделки докажем в модели EUF-CMA. В рамках данной модели злоумышленник $\mathcal{A}$ может отправлять $q_s$ запросов на подпись, а также $q_h$ запросов на хеширование выбранных им сообщений. Злоумышленник выигрывает, если на выходе он выдаёт корректную подделку подписи для сообщения, для которого не отправлял запроса на подпись. В рамках теоретического доказательства приведём алгоритм симуляции подписи, а также алгоритм сведения задачи подделки подписи к задаче SIS.

Пусть существует алгоритм $\mathcal{B}$, на вход которого подаётся матрица $\mathbf{A}$. На выходе $\mathcal{B}$ должен вернуть короткий вектор $\mathbf{v}$, такой, что $\mathbf{Av} = \mathbf{0}$. Для решения задачи SIS $\mathcal{B}$ использует алгоритм $\mathcal{A}$, решающий задачу подделки подписи. $\mathcal{B}$ выбирает параметры $q, k, l, m$ для схемы цифровой подписи и из экземпляра задачи SIS генерирует открытый и секретный ключи: для этого $\mathcal{B}$ выбирает матрицу $\mathbf{S} \leftarrow \mathcal{C}^{l \times m}$ и вычисляет $\mathbf{T} = \mathbf{AS} \bmod q$. Алгоритм $\mathcal{A}$ получает на вход от $\mathcal{B}$ матрицы $\mathbf{A}$ и $\mathbf{T}$ и на выходе выдаёт подделку подписи $(\mathbf{z}, \mathbf{c})$. В ходе работы $\mathcal{A}$ может делать два типа запросов: на хеш и на подпись. Для обработки запросов от $\mathcal{A}$ алгоритм $\mathcal{B}$ использует симулятор $\mathcal{S}$,

который симулирует схему цифровой подписи таким образом, что при создании подписи не используется секретный ключ. На вход симулятору подаются матрицы $\mathbf{A}, \mathbf{T}$, а алгоритм $\mathcal{A}$ отправляет $\mathcal{S}$ свои запросы. При обработке запросов на подпись и на хеш $\mathcal{S}$ использует случайный оракул $H$, который заменяет хеш-функцию (алгоритм 5).

**Алгоритм 5.** Симуляция подписи

**Вход:** $\mathbf{A}, \mathbf{T}, \mu, M, \kappa$.
**Выход:** $(\mathbf{z}, \mathbf{c})$.

1: $\mathbf{c} \xleftarrow{\$} \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}, \|\mathbf{v}\|_1 \leqslant \kappa\}$.
2: $\mathbf{z} \xleftarrow{\$} \mathcal{P}$.
3: С вероятностью $1/M$:
4: Установить $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
5: Вернуть $(\mathbf{z}, \mathbf{c})$.

Важным аспектом при доказательстве является неразличимость между выходами алгоритмов $\mathcal{S}$ и реальной схемы цифровой подписи. Данное условие является обязательным, так как в противном случае $\mathcal{A}$ отличит выход симулированной от действительной схем и закончит работу, если получит симулированную подпись от $\mathcal{S}$. Неразличимость распределений выходов таких алгоритмов определяется с помощью разных статистических методов, например статистической разностью (в оригинальной работе [26]) и дивергенцией (дивергенция Реньи используется в [47]).

В [26] доказывается неразличимость распределений выходов реальной схемы и симулятора $\mathcal{S}$ при использовании в качестве $\mathcal{P}, \mathcal{Q}$ дискретных гауссовых распределений. Далее доказательство происходит следующим образом: пусть $\mathcal{A}$ с некоторой вероятностью вернул подделку подписи $(\mathbf{z}, \mathbf{c})$ для сообщения $\mu$. Допустим, что $\mathcal{A}$ не обращался с запросом на хеш для сообщения $\mathbf{w} = \mathbf{Az} - \mathbf{Tc}$, тогда вероятность того, что он сможет вычислить $\mathbf{c} = H(\mathbf{w}, \mu)$, равна $1/|D_H|$, где $D_H$ — область значений $H$. Эта вероятность пренебрежимо мала, поэтому будем считать, что $\mathcal{A}$ делал запрос на хеш. Поскольку мы знаем нужный запрос, то для него и последующих запросов заново выработаем новые ответы и запустим алгоритм $\mathcal{A}$ снова. Тогда по лемме разветвления [49] получаем другую подделку подписи $(\mathbf{z}', \mathbf{c}')$ для сообщения $\mu$. Поскольку $\mathbf{c}, \mathbf{c}'$ являются результатами выхода $H(\mathbf{Aw}, \mu)$, а $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$ и $\mathbf{c}' = H(\mathbf{Az}' - \mathbf{Tc}', \mu)$, то $\mathbf{Az} - \mathbf{Tc} = \mathbf{Az}' - \mathbf{Tc}'$. В итоге получим

$$\mathbf{A}(\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc}) = \mathbf{0}.$$

Вектор $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc}$ является малым по норме, поэтому осталось доказать, что $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc} \neq \mathbf{0}$. Пусть $i$ — это позиция, в которой различаются векторы $\mathbf{c}$ и $\mathbf{c}'$. По лемме из [26] с вероятностью $1 - 2^{-100}$ существует секретный ключ $\mathbf{S}'$, отличающийся от $\mathbf{S}$ только в $i$-м столбце, и $\mathbf{AS} = \mathbf{AS}'$. Тогда если $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc} = \mathbf{0}$, то существует такой $\mathbf{S}'$, что $\mathbf{z} - \mathbf{z}' + \mathbf{S}'\mathbf{c}' - \mathbf{S}'\mathbf{c} \neq \mathbf{0}$. Поскольку $\mathcal{A}$ неизвестен секретный ключ $\mathbf{S}$, мы получим решение SIS с вероятностью $1/2$. Таким образом, задача подделки подписи не легче, чем задача SIS.

Другой задачей для злоумышленника является восстановление ключа. Он может это сделать двумя способами: 1) либо решить уравнение $\mathbf{AS} = \mathbf{T} \pmod q$, 2) либо найти секретный ключ из значений $(\mathbf{z}, \mathbf{c})$, где $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$. Первый вариант является сложным, так как требует решения задачи ISIS (Inhomogeneous SIS), которая не легче SIS, второй вариант также сложный, так как требует решения задачи Search-LWE.

Выбор распределений $\mathcal{P}$ и $\mathcal{Q}$. Этот выбор важен с точки зрения компактности получаемой подписи, а также сложности программной реализации схемы. Так, например, в выбранной в качестве стандарта схеме цифровой подписи Dilithium [45] используется равномерное распределение в гиперкубе. Благодаря этому достигается простота программной реализации схемы цифровой подписи и стойкость к атакам по побочным каналам. В работах [26, 50] используется дискретное гауссово распределение, из-за чего размеры подписи удаётся уменьшить, однако ввиду сложности реализации гауссова распределения это приводит к раскрытию секретной информации при использовании атак по побочным каналам [51–53]. Кроме того, гауссово распределение не ограничено и требуется делать дополнительные проверки, что подпись корректна. Новым подходом стало использование равномерного распределения в $n$-мерном гипершаре для генерации подписи [54]. В [54] показано, что ожидаемый размер подписи, полученной при выборке из гипершара, равен ожидаемому размеру подписи, полученной из дискретного гауссова распределения, однако первая выборка реализуется проще, чем вторая. С применением результатов [54] была разработана и предложена для рассмотрения на новом этапе конкурса NIST схема цифровой подписи HAETAE [47]. Благодаря использованию равномерного распределения из гипершара, в HAETAE удалось уменьшить размер подписи на 29–39 % и размер открытого ключа на 20–25 % в зависимости от уровня стойкости.

### 4.2. Схема цифровой подписи Dilithium

Схема цифровой подписи Dilithium выбрана Национальным институтом стандартов и технологий США в качестве одного из новых стандартов цифровой подписи [46]. Схема базируется на работах В. Любашевского, но использует модульные решётки. Таким образом, стойкость схемы основывается на модульных вариантах задач SIS и LWE [12, 55]. В схеме Dilithium для выработки секретных и маскирующих векторов используется равномерное распределение в гиперкубе. Такой выбор распределения обусловлен простотой реализации и защитой от атак по побочным каналам. Опишем упрощённую (менее эффективную) версию, являющуюся несколько изменённой версией схемы из [56].

Генерация ключей (алгоритм 6). Алгоритм генерирует матрицу $\mathbf{A}$ размера $k \times \ell$, являющуюся открытой; каждый её элемент является многочленом в кольце $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, где $q = 2^{23} - 2^{13} + 1$, $n = 256$. После этого случайным образом выбираются секретные векторы $\mathbf{s}_1$ и $\mathbf{s}_2$. Каждая компонента этих векторов является элементом из множества $S_\eta$, то есть элементом $\mathcal{R}_q$ малой нормы, не превышающей $\eta$. В качестве нормы элемента $\mathcal{R}_q$ вычисляется его бесконечная норма $\|a\|_\infty = \max\limits_i |a_i|$, где $a \in \mathcal{R}_q$, то есть каждый коэффициент выбираемого многочлена лежит в $\{-\eta, \ldots, \eta\}$. Множество $S_\eta^k$ называется $k$-мерным гиперкубом, а векторы выбираются из него случайно и равномерно. Вторая часть открытого ключа равна $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. Все алгебраические операции выполняются над кольцом $\mathcal{R}_q$ (алгоритм 6).

**Алгоритм 6.** Выработка ключей Dilithium

1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$.
2: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$.
3: $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
4: **Вернуть** $(\mathbf{A}, \mathbf{t})$ — открытый ключ, $(\mathbf{s}_1, \mathbf{s}_2)$ — секретный ключ.

Формирование подписи (алгоритм 7). Подписывающий генерирует маскирующий вектор $\mathbf{y}$, состоящий из многочленов из $\mathcal{R}_q$ с коэффициентами меньше $\gamma_1$. Параметр $\gamma_1$ достаточно велик, чтобы конечная подпись не раскрывала секретный ключ (т. е. алгоритм формирования подписи основан на нулевом разглашении), но в то же время достаточно мал, чтобы подпись было сложно подделать. Затем подписывающий вычисляет $\mathbf{Ay}$ и берёт в качестве $\mathbf{w}_1$ старшие биты компонентов вычисленного вектора. В частности, каждая компонента $w$ в $\mathbf{Ay}$ может быть записана в каноническом виде $w = w_1 \cdot 2\gamma_2 + w_0$, где $|w_0| \leqslant \gamma_2$. Таким образом, $\mathbf{w}_1$ — это вектор, содержащий все значения $w_1$. Вызов $c$ задаётся как хеш сообщения $M$ и вектора $\mathbf{w}_1$ и представляет собой многочлен из $\mathcal{R}_q$ с элементами $\pm 1$ в количестве $\tau$ и остальными элементами, равными нулю. Причина такого распределения заключается в том, что $c$ имеет малую норму и является элементом кольца размера $\log_2 \binom{256}{\tau} + \tau$, находящегося в диапазоне от 128 до 256. Сама подпись вычисляется как $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$.

Для избежания зависимости $\mathbf{z}$ от секретного ключа используется выборка отбраковки. Параметр $\beta$ задан как максимально возможный коэффициент $c\mathbf{s}_i$. Поскольку количество $\pm 1$ в $c$ равно $\tau$, а максимальный коэффициент в $\mathbf{s}_i$ равен $\eta$, очевидно, что $\beta \leqslant \tau \cdot \eta$. Если какой-либо коэффициент $\mathbf{z}$ больше, чем $\gamma_1 - \beta$, то подпись отклоняется и процедура генерации подписи запускается сначала. Кроме того, если младшие биты какого-либо коэффициента вектора $\mathbf{Az} - c\mathbf{t}$ больше, чем $\gamma_2 - \beta$, процедура генерации подписи перезапускается. Первая проверка необходима для обеспечения стойкости, вторая — не только для стойкости, но и для корректности.

**Алгоритм 7.** Формирование подписи Dilithium

**Вход:** Сообщение $M$, открытый ключ $(\mathbf{A}, \mathbf{t})$, секретный ключ $(\mathbf{s}_1, \mathbf{s}_2)$.
**Выход:** Подпись $\sigma = (\mathbf{z}, c)$ сообщения $M$.

1: $\mathbf{z} := \perp$.
2: **Пока** $\mathbf{z} = \perp$:
3: $\quad \mathbf{y} \leftarrow S^{\ell}_{\gamma_1 - 1}$;
4: $\quad \mathbf{w}_1 := \mathsf{HighBits}(\mathbf{Ay}, 2\gamma_2)$;
5: $\quad c(\in B_\tau) := H(M \| \mathbf{w}_1)$;
6: $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$.
7: $\quad$ **Если** $\|\mathbf{z}\|_\infty \geqslant \gamma_1 - \beta$ или $\|\mathsf{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geqslant \gamma_2 - \beta$, **то** $\mathbf{z} := \perp$.
8: **Вернуть** $\sigma = (\mathbf{z}, c)$.

Проверка (алгоритм 8). Проверяющий сначала вычисляет $\mathbf{w}'_1$ в качестве старших битов вектора $\mathbf{Az} - c\mathbf{t}$, а затем принимает подпись, если все коэффициенты $\mathbf{z}$ меньше $(\gamma_1 - \beta)$ и если $c$ является хешем сообщения $M$ и вектора $\mathbf{w}'_1$. Отметим, что $\mathbf{Az} - c\mathbf{t} = \mathbf{Ay} - c\mathbf{s}_2$. Кроме того, $\mathsf{HighBits}(\mathbf{Ay}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)$. Это верно потому, что действительная подпись содержит $\|\mathsf{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$. Поскольку мы знаем, что коэффициенты $c\mathbf{s}_2$ меньше $\beta$, добавления $c\mathbf{s}_2$ недостаточно, чтобы вызвать какие-либо переносы путём увеличения любого коэффициента младшего порядка до величины не менее $\gamma_2$.

**Алгоритм 8.** Проверка подписи Dilithium

**Вход:** Сообщение $M$, открытый ключ $(\mathbf{A}, \mathbf{t})$, подпись $\sigma = (\mathbf{z}, c)$.
**Выход:** Принять или отклонить подпись.
1: $\mathbf{w}'_1 := \mathsf{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$.
2: **Если** $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ и $c = H(M\|\mathbf{w}'_1)$, **то**
принять подпись.

Безопасность. Наследуя идею доказательства из [26], стойкость схемы Dilithium доказывается похожим образом в модели случайного оракула, основываясь на сложности двух задач. Первая задача — это задача распознавания LWE над кольцами многочленов, в рамках которой требуется различить экземпляр задачи LWE $(\mathbf{A}, \mathbf{t} = \mathbf{As_1} + \mathbf{s_2})$ и пару $(\mathbf{A}, \mathbf{u})$, где $\mathbf{u}$ выбран случайно. Второй является задача SelfTargetMSIS [57], в рамках которой требуется найти короткий вектор $\begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}$ и сообщение $\mu$, удовлетворяющие равенству

$$H\left(\mu \| [\mathbf{A}|\mathbf{t}|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}\right) = c.$$

Для модели случайного оракула можно также произвести сведение, используя лемму о разветвлении [49], от задачи MSIS к задаче SelfTargetMSIS, сводя тем самым стойкость схемы Dilithium к сложности задач MSIS и MLWE. Однако в модели QROM мы не можем воспользоваться данной леммой, поэтому сложность подделки подписи основана на задачах MLWE и SelfTargetMSIS. Обоснование сложности задачи и соотвественно стойкости схемы опирается на следующие причины:

1) на данный момент не существует схем цифровой подписи, построенных с помощью преобразования Фиата — Шамира из сигма-протокола, которые являются стойкими в модели ROM и нестойкими в модели QROM;
2) есть возможность выбрать такой набор параметров, при котором задача SelfTargetMSIS станет информационно-теоретически сложной и стойкость схемы будет основана только на задаче MLWE. Но при таком наборе параметров размеры подписи и открытого ключа увеличиваются в несколько раз [57].

Однако в недавней работе [58] показано, что если сигма-протокол коллапсирует и имеет свойство special soundness, то схема цифровой подписи, полученная с помощью преобразования Фиата — Шамира, будет стойкой в модели QROM. Свойство special soundness сигма-протокола Dilithium обеспечивается сложностью задачи MSIS, а в работах [58, 59] показано, что сигма-протокол коллапсирует. Таким образом, можно считать, что безопасность схемы для модели QROM обоснована.

### 4.3. Схема цифровой подписи HAETAE

HAETAE — схема постквантовой цифровой подписи на решётках, представленная на дополнительном конкурсе NIST. HAETAE также построена на парадигме Фиата — Шамира с прерываниями [10, 26], а её стойкость основана на сложности решения модульных версий задач SIS и LWE [12, 55]. Несмотря на то, что HAETAE частично похожа на схему CRYSTALS-Dilithium, она имеет следующие основные отличия:

1) вместо унимодального распределения для генерации выборки отбраковки HAETAE использует бимодальное распределение, как в схеме цифровой подписи BLISS [50];

2) выборки случайных векторов и векторов отбраковки осуществляются с помощью использования многомерного гипершара, а не гиперкуба.

Опишем некоторые предварительные сведения, необходимые для понимания работы схемы HAETAE.

Улучшение компактности. Выбор распределений для выборки случайных векторов и векторов отбраковки существенно влияет на размер подписи [54]. Dilithium использует дискретные равномерные распределения в гиперкубах, что упрощает реализацию схемы. Однако такие распределения далеки от оптимальных с точки зрения результирующих размеров подписей. В HAETAE существует компромисс: немного теряя в простоте реализации, получаются более компактные подписи.

Равномерное распределение на гипершаре. Гауссово распределение превосходит равномерное распределение на гипершаре с точки зрения компактности подписи [54]. Однако у гауссова распределения есть два недостатка:

— шаг отбраковки включает вычисление некоторой трансцендентной функции на входных данных, которая зависит от секретного ключа. Это является громоздким для реализации и чувствительным к атакам по побочным каналам [52];
— поскольку итоговая подпись связана с гауссовым распределением, существует ненулевая вероятность, что подпись будет слишком большой и не пройдёт проверку.

Как следствие, в качестве альтернативного выбора в [54] представлены равномерные распределения на гипершарах, приводящие к более компактным подписям, нежели гауссовы распределения и равномерные распределения на гиперкубе. Таким образом, шаг отбраковки заключается всего лишь в проверке того, достаточно ли малы евклидовы нормы рассматриваемых векторов.

Бимодальное распределение. Модификация схемы цифровой подписи Любашевского [10, 26] представлена в [50] и использует бимодальное распределение при генерации самой подписи. Таким образом, подпись имеет вид $\mathbf{y}+(-1)^b\mathbf{Sc}$, где вектор $\mathbf{y}$ выбран из фиксированного распределения, а величина $b \in \{0,1\}$ выбрана равномерно. Подпись отклоняется для заданного целевого распределения, не зависящего от секрета. Чтобы убедиться, что проверка прошла успешно, все вычисления выполняются по модулю $2q$, а генерация ключа приводит к равенству $\mathbf{AS} = q\mathbf{Id}$. Оказывается, что эта модификация может привести к более компактным подписям, чем унимодальный случай.

Выбор кольца и модуля. Оригинальный дизайн схемы цифровой подписи BLISS [50] основан на задачах Ring-LWE и Ring-SIS, а алгоритм генерации ключей — на соотношениях полиномов по аналогии с NTRU. Для обеспечения бóльшей гибкости без потери эффективности реализации в HAETAE рассматриваются модульные решётки (как в Dilithium) с фиксированным кольцом многочленов $\mathcal{R} = \mathbb{Z}[x]/(x^{256}+1)$ для всех уровней стойкости.

В качестве модуля $q$ выбирается простое число, такое, что операции в кольце $\mathcal{R}$ могут быть эффективно реализованы, а алгоритм декомпозиции битов работает правильно. Для операций в кольце $\mathcal{R}$ используется теоретико-числовое преобразование (NTT), а мультипликативная группа $\mathbb{Z}_q^*$ имеет элемент порядка 512, что эквивалентно условию $q \equiv 1 \pmod{512}$. В HAETAE $q = 64513$.

Генерация ключей. Ключевой парой схемы HAETAE является $(\mathbf{A}, \mathbf{s})$, где $\mathbf{A} \in \mathcal{R}_p^{k\times(k+l)}$ — открытый ключ; $\mathbf{s} \in \mathcal{R}_p^{k+l}$ — секретный, причём $\mathbf{As} = -\mathbf{As} \pmod p$. Для генерации такой пары полагают $p = 2q$ и стремятся к выполнению условия $\mathbf{As} = q\mathbf{j} \pmod{2q}$, где $\mathbf{j} = (1,0,0,\ldots,0)^{\mathrm{T}}$.

Матрица $\mathbf{A}$ и вектор $\mathbf{s}$ получаются следующим образом: генерируется MLWE-выборка $\mathbf{b}-\mathbf{a} = \mathbf{A}_0\mathbf{s}_0+\mathbf{e}_0 \bmod q$, где $\mathbf{A}_0 \leftarrow U(\mathcal{R}_q^{k\times(l-1)})$; $\mathbf{a} \leftarrow U(\mathcal{R}_q^k)$; $(\mathbf{s}_0, \mathbf{e}_0) \leftarrow U(S_\eta^{l-1}\times S_\eta^k)$. Для любого $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_0$ определим $\mathbf{A} = (2(\mathbf{a} - \mathbf{b}_1) + q\mathbf{j}|2\mathbf{A}_0|2\mathbf{I}_k)$ и $\mathbf{s} = (1|\mathbf{s}_0|\mathbf{e}_0 - \mathbf{b}_0)$.

Для разложения $nk$-мерного вектора $\mathbf{b}$ с координатами в $[0, q-1]$ выбираем $\mathbf{b}_0$ с координатами в $\{-1, 0, 1\}$ таким образом, что если координата $\mathbf{b}$ нечётная, то она округляется до ближайшего значения, кратного 4. Далее можем записать $\mathbf{b} = \mathbf{b}_0 + 2\mathbf{b}_1$, где $\mathbf{b}_1$ кодируется с использованием $\lceil \log_2 q - 1 \rceil$ бит на координату и $\mathbf{b}_0 = (-1)^{\lceil \mathbf{b}/2 \rceil \bmod 2}\mathbf{b} \bmod 2$.

Критическим шагом в алгоритме генерации ключей является ограничение нормы $\|\mathbf{s}c\|_2$, где $\mathbf{s}$ генерируется, как и ранее, а $c \in \mathcal{R}$ — многочлен с коэффициентами в $\{0, 1\}$ и имеет $\leqslant \tau$ ненулевых коэффициентов для некоторого $\tau$. Чем ниже эта граница, тем меньше подпись, что, в свою очередь, приводит к усложнению подделки.

Формирование и проверка подписи (алгоритмы 9 и 10). Подпись сообщения $M$ состоит из двух компонент: $c = H(\mathbf{A}\lfloor \mathbf{y} \rceil \bmod 2q, M)$ и $\mathbf{z} = \lfloor \mathbf{y} \rceil \pm c\mathbf{s}$. Иногда вектор $\mathbf{z}$ отклоняется и процесс подписания выполняется снова. Отметим, что $\mathbf{Az} = \mathbf{A}\lfloor \mathbf{y} \rceil + qc\mathbf{j} \bmod 2q$ не зависит от знака, выбранного для $c\mathbf{s}$. На этапе верификации подписи проверяется согласованность пары $(\mathbf{z}, c)$ и малый размер нормы вектора $\mathbf{z}$.

**Алгоритм 9.** Формирование подписи

**Вход:** Секретный ключ $(\mathbf{A}, \mathbf{s})$, сообщение $M$.
**Выход:** $\sigma = (\lfloor \mathbf{z} \rceil, c)$.
1: $\mathbf{y} \leftarrow U(\mathscr{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$.
2: $\mathbf{w} \leftarrow \mathbf{A}\lfloor \mathbf{y} \rceil$.
3: $c := H(\mathbf{w}, M) \in \mathcal{R}_2$.
4: $\mathbf{z} := y + (-1)^b c\mathbf{s}$ для $b \leftarrow U(\{0, 1\})$.
5: **Если** $\|\mathbf{z}\|_2 > B'$, **то**
начать сначала,
6: **иначе**
7: **Если** $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, **то**
начать сначала с вероятностью $1/2$.
8: **Вернуть** $\sigma = (\lfloor \mathbf{z} \rceil, c)$.

**Алгоритм 10.** Проверка подписи

**Вход:** Открытый ключ $\mathbf{A}$, сообщение $M$, подпись $\sigma = (\tilde{\mathbf{z}}, c)$.
**Выход:** $\sigma = (\lfloor \mathbf{z} \rceil, c)$.
1: $\tilde{\mathbf{w}} := \mathbf{Az} - qc\mathbf{j} \bmod 2q$.
2: **Вернуть** $c = H(\tilde{\mathbf{w}}, M)$ и $\|\tilde{\mathbf{z}}\| < B + \sqrt{n(k+\ell)}/2$.

Безопасность. Стойкость схемы HAETAE основывается на сложности задач MLWE и BimodalSelfTargetMSIS [47]. Суть задачи BimodalSelfTargetMSIS состоит в поиске короткого вектора $\mathbf{y}$, значения $c$ и сообщения $\mu$, таких, что $H((\mathbf{Ay} - qc\mathbf{j}) \bmod 2q, \mu) = c$, где матрица $\mathbf{A} = (2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_0 \mid 2\mathbf{I}_k) \bmod 2q$. В [47] представлено сведение задачи MSIS к задаче BimodalSelfTargetMSIS, подобно сведению MSIS к SelfTargetMSIS [45]. Доказательство использует идеи из [60] и сводит стойкость в модели UFCMA к стойкости в модели UFNMA. Для этого требуется, чтобы соответствующая схеме цифровой подписи протокол аутентификации сторон обладал высокой мин-энтропией обязательств

(commitment min-entropy), а также свойством нулевого разглашения честного проверяющего (honest-verifier zero-knowledge), что доказывается для протокола аутентификации сторон, построенного на основе HAETAE.

## 5. Оценка стойкости и подбор параметров

Обоснование стойкости криптографических схем состоит из двух частей: теоретического доказательства стойкости схемы и её практического криптоанализа. Теоретическое доказательство необходимо для обоснования сложности взлома схемы. Так, в ходе доказательства производится сведение задачи взлома к вычислительно сложной математической задаче. Практический анализ стойкости схемы представляет собой оценку её стойкости к ряду самых успешных существующих на данный момент атак, исходя из которой подбираются параметры, обеспечивающие заданный уровень безопасности. Так как стойкость рассмотренных схем цифровой подписи основывается на сложности задач SIS и LWE в различных формулировках, одним из главных видов атак на такие схемы являются атаки, направленные на поиск коротких векторов в решётке.

На сегодняшний день лучшим алгоритмом редукции базиса и нахождения коротких векторов является BKZ [61]. Алгоритм BKZ на вход принимает базис решётки, а также параметр $b$ (размер блока) и на выходе возвращает редуцированный базис. В рамках работы алгоритма происходит обращение к оракулу, решающему задачу SVP для решётки размерности $b$. Количество обращений к оракулу полиномиально, поэтому сложность алгоритма BKZ зависит от сложности алгоритма, решающего SVP и используемого оракулом. Отсюда следует, что в рамках оценки стойкости схемы рассматривается сложность не всего алгоритма BKZ, а только одного вызова алгоритма, решающего SVP. Очевидно, что при увеличении значения $b$ увеличивается время работы алгоритма и уменьшается предполагаемая норма получаемого кратчайшего вектора на его выходе. Таким образом, для оценки стойкости схемы необходимо сначала определить размер блока $b$, при котором можно будет схему взломать, а далее оценить сложность алгоритма решения SVP для такой размерности, эта сложность и является уровнем стойкости схемы. Такой подход к оценке практической стойкости криптографических схем на решётках предложен в работе [62] и называется методологией coreSVP hardness.

В настоящее время сложность лучшего классического алгоритма [63] решения задачи SVP равна $\approx 2^{c_C \cdot b}$, где $c_C = \log_2 \sqrt{3/2} \approx 0{,}292$. Лучший квантовый алгоритм [64] решает SVP за время $\approx 2^{c_Q \cdot b}$, где $c_Q = \log_2 \sqrt{13/9} \approx 0{,}265$. Также считается, что даже с улучшениями вряд ли сложность квантового алгоритма решения SVP сможет быть меньше $\sqrt{4/3}^{b+o(b)} \approx 2^{0{,}2075\, b}$.

Теперь опишем подход, с помощью которого реализуется атака на экземпляр задачи LWE $(\mathbf{A}, \mathbf{t})$, где $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$; $\mathbf{A} \in \mathbb{Z}^{n \times m}$; $\mathbf{s} \in \mathbb{Z}^m$; $\mathbf{t}, \mathbf{e} \in \mathbb{Z}^n$. Для LWE существуют два основных вида атак — это атаки на основную решётку и дуальную к ней. Для реализации первой атаки построим следующую решётку:

$$\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} : (\mathbf{A}| - \mathbf{I}_m| - \mathbf{t})\mathbf{x} = \mathbf{0} \pmod q\}.$$

Значение $d = m + n + 1$ является размерностью решётки, $q^m$ — её объёмом. Для данной решётки кратчайшим вектором является вектор $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$. Для случайных решёток $\mathcal{L}$ существует гауссова эвристика — это некоторое предположение, полученное эвристическим методом, которое примерно оценивает $\lambda_1(\mathcal{L})$ в случайных решётках,

и данная эвристика равна $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d}{2\pi e}} \cdot \mathrm{Vol}(\Lambda)^{1/d}$, где $d$ — размерность решётки. Поскольку для решётки $\Lambda$ значение $\lambda_1(\Lambda) = \|\mathbf{v}\|$ значительно меньше, чем эвристика случайной решётки, которой равен второй минимум $\lambda_2(\Lambda)$, перед нами лежит экземпляр задачи unique-SVP. Для решения задачи и оценки сложности алгоритма BKZ опишем некоторые предположения и определения, на которых строится оценка.

**Предположение** о геометрической прогрессии [65]. Длины векторов Грама — Шмидта после редукции базиса решётки удовлетворяют следующему соотношению:

$$\|\mathbf{b}_i^*\| = \alpha^{i-1} \cdot \|\mathbf{b}_1\|, \quad 0 < \alpha < 1.$$

**Определение 9** (корневой эрмитовый фактор [66]). Корневой эрмитовый фактор базиса $\mathbf{B}$ решётки $\mathcal{L}$ определяется как

$$\delta(\mathbf{B}) = \left( \frac{\|\mathbf{b}_1\|}{\mathrm{Vol}(\mathcal{L})^{1/d}} \right)^{1/d}.$$

Для BKZ-$\beta$ редуцированного базиса случайной решётки, согласно гауссовой эвристике, выполняется следующее соотношение [66]:

$$\lim_{d\to\infty} \delta(\mathbf{B}) = \left( \frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2(\beta-1))}.$$

Пусть $\delta_0 = \left( \frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2(\beta-1))}$. Выразим значение $\|\mathbf{b}_1\|$ как $\|\mathbf{b}_1\| = \delta_0^d \cdot \mathrm{Vol}(\mathcal{L})^{1/d}$. Объединяя предположение о геометрической прогрессии, значение корневого эрмитового фактора и выражение $\mathrm{Vol}(\mathcal{L}) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$, получаем значение $\alpha = \delta_0^{-2d/(d-1)} \approx \delta_0^{-2}$. Таким образом, имеем следующую зависимость: $\|\mathbf{b}_i^*\| = \delta_0^{2-2i} \cdot \mathrm{Vol}(\mathcal{L})^{1/d}$.

Для редуцированного базиса мы считаем, что выполняется предположение о геометрической прогрессии, поэтому можем предсказать предполагаемое значение $\|\mathbf{b}_i^*\|$ после запуска на базисной матрице алгоритма BKZ с параметром блока $\beta$. Так как значение длины проекций векторов уменьшается к последним векторам матрицы, возьмём последний блок $(\mathbf{b}_{d-\beta}, \ldots, \mathbf{b}_d)$: в данном блоке оракул SVP найдёт вектор с проекцией, примерно равной $\|\mathbf{b}_{d-\beta+1}^*\| \approx \delta_0^{2\beta-d} \cdot \mathrm{Vol}(\Lambda)^{1/d}$. Однако если проекция искомого вектора $\mathbf{v}$ к первым $d-\beta$ векторам окажется меньше, то оракул непременно её найдёт. Таким образом, для успешного нахождения вектора $\mathbf{v}$ требуется, чтобы

$$\|\mathbf{v}^*\| \leqslant \delta_0^{2\beta-d} \cdot \mathrm{Vol}(\Lambda)^{1/d} = \delta_0^{2\beta-d} \cdot q^{m/d}. \tag{2}$$

Найдя наименьшее значение $\beta$, такое, чтобы выполнялось (2), можем оценить сложность алгоритма SVP: $2^{0{,}292\,\beta}$ и $2^{0{,}265\,\beta}$ для классического и квантового вычислителя соответственно.

Дуальная атака на LWE [62] состоит из поиска короткого вектора в дуальной решётке $\Lambda' = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^n : \mathbf{A}^T\mathbf{x} = \mathbf{y} \pmod q\}$. Найдя короткий вектор $(\mathbf{w}, \mathbf{v}) \in \Lambda'$ длины $l$, можем вычислить значение $\langle \mathbf{w}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{s} \rangle + \langle \mathbf{w}, \mathbf{e} \rangle$, которое тоже является малым значением и распределено согласно гауссовому распределению со стандартным отклонением $l\sigma$, если пара $(\mathbf{A}, \mathbf{b})$ является экземпляром задачи LWE, в противном случае распределение равномерное. Расстояние между двумя такими распределениями ограничено значением $\varepsilon = 4\exp(-2\pi^2\tau^2)$, где $\tau = l\sigma/q$. Нахождение такого вектора

даёт $\varepsilon$-преимущество для решения задачи Decision-LWE. Пусть $l = \|\mathbf{b}_1\|$ после применения BKZ. Зная, что $\Lambda'$ имеет размерность $d = m + n$ и объём $q^n$, получаем значение $l = \delta^{d-1} q^{n/d}$. Таким образом, для получения преимущества $\varepsilon$ требуется запустить алгоритм BKZ с блоком $\beta$, таким, что

$$-2\pi^2\tau^2 \geqslant \ln(\varepsilon/4).$$

Методология coreSVP hardness используется для анализа безопасности и подбора параметров для большинства существующих схем цифровой подписи на решетках, в частности, с её помощью подобраны параметры для схем Dilithium, HAETAE и Falcon. В рамках оценки стойкости анализировались две атаки: на подделку подписи и на восстановление секретного ключа. Для каждой атаки подобрано значение блока $\beta$, для которого соблюдается уровень стойкости, определённый Национальным институтом стандартов и технологий США. Значения блоков и соответствующие им уровни стойкости схем в битах как против классического, так и против квантового злоумышленника представлены в табл. 4.

Таблица 4

**Значения блоков BKZ и оценки стойкости схем цифровой подписи Falcon, Dilithium, HAETAE**

| Схема | Восстановление ключа | | | Подделка подписи | | |
|---|---|---|---|---|---|---|
| | $\beta$ | Классическая сложность | Квантовая сложность | $\beta$ | Классическая сложность | Квантовая сложность |
| Уровень стойкости I (NIST) | | | | | | |
| Falcon | 458 | 133 | 121 | 411 | 120 | 108 |
| Уровень стойкости II (NIST) | | | | | | |
| Dilithium | 423 | 123 | 112 | 423 | 123 | 112 |
| HAETAE | 428 | 125 | 109 | 409 | 119 | 105 |
| Уровень стойкости III (NIST) | | | | | | |
| Dilithium | 638 | 186 | 169 | 624 | 182 | 165 |
| HAETAE | 810 | 236 | 208 | 617 | 180 | 158 |
| Уровень стойкости V (NIST) | | | | | | |
| Falcon | 936 | 273 | 248 | 952 | 277 | 252 |
| Dilithium | 909 | 265 | 241 | 863 | 252 | 229 |
| HAETAE | 988 | 288 | 253 | 878 | 256 | 225 |

Производительность схем, а также размеры открытого и секретного ключей и подписи также зависят от требуемого уровня стойкости и для схем Dilithium, Falcon и HAETAE представлены в табл. 5. Сравнивая схемы по разным показателям эффективности, нельзя определённо сказать, какая из них лучшая, так как каждая имеет свои преимущества и недостатки.

Таблица 5

**Показатели эффективности схем цифровой подписи Falcon, Dilithium, HAETAE**

| Схема | Открытый ключ (байты) | Секретный ключ (байты) | Подпись (байты) | Формирование подписи (циклы) | Проверка подписи (циклы) |
|---|---|---|---|---|---|
| Уровень стойкости I (NIST) | | | | | |
| Falcon | 897 | 1 281 | 666 | 1 009 764 | 81 036 |
| Уровень стойкости II (NIST) | | | | | |
| Dilithium | 1 312 | 2 528 | 2 420 | 333 013 | 118 412 |
| HAETAE | 992 | 1 376 | 1 463 | 6 253 166 | 387 594 |
| Уровень стойкости III (NIST) | | | | | |
| Dilithium | 1 952 | 4 000 | 3 293 | 529 106 | 179 424 |
| HAETAE | 1 472 | 2 080 | 2 337 | 9 472 724 | 718 010 |
| Уровень стойкости V (NIST) | | | | | |
| Falcon | 1 793 | 2 305 | 1 280 | 2 053 080 | 160 596 |
| Dilithium | 2 592 | 4 864 | 4 595 | 642 192 | 279 936 |
| HAETAE | 2 080 | 2 720 | 2 908 | 8 989 980 | 913 378 |

## Заключение

Рассмотрев существующие подходы к построению схем цифровой подписи на решётках, а также сравнив количественные показатели флагманских схем, приходим к выводу, что лучшего подхода на данный момент не существует, а каждый из существующих обладает своими преимуществами и недостатками. Парадигма GGH/NTRUSign сейчас отошла на второй план ввиду того, что главная схема NTRUSign оказалась небезопасной [19], а новые модификации проигрывают в эффективности с точки зрения размеров ключей и подписи другим схемам на решётках.

Схемы цифровой подписи, построенные на двух других парадигмах, оказались более успешными, и две из них — Falcon и Dilithium — выбраны в качестве нового стандарта цифровой подписи США [46]. Анализируя оба подхода, нельзя прийти к однозначному мнению, что один из них лучше, чем другой, а сравнивая показатели эффективности схем — представителей данных парадигм, можно сделать вывод, что выбор схемы зависит от условий её использования. Так, схема цифровой подписи Falcon обладает достаточно малым размером подписи и малым временем её проверки, однако время создания подписи в несколько раз выше, чем у схемы Dilithium. Важным аспектом является также факт, что при формировании подписи производятся операции с числами с плавающей точкой, данное свойство не позволяет использовать Falcon на системах, не поддерживающих такие операции, а также ведёт к утечке информации по побочным каналам; при проверке подписи операции с числами с плавающей точкой не производятся. Использование схемы Falcon вместе с математическим сопроцессором сильно ускоряет алгоритм формирования подписи: общее время работы алгоритмов создания подписи и её проверки становится меньше, чем у схемы Dilithium [67], однако авторы в [67] замечают, что при использовании математического сопроцессора операции с числами с плавающей точкой всё равно не являются константными по времени, что может потенциально привести к атакам по побочным каналам. Исходя из особенностей схемы Falcon, рекомендуется использовать её в случае, когда клиенту требуется только хранить и проверять подпись, в то время как формирование подписи можно производить на производительном сервере [67–69]; также рекомендуется делать это в оффлайн-режиме [70] в целях недопущения атак по побочным каналам.

Схема цифровой подписи Dilithium является более универсальной, по сравнению со схемой Falcon она обладает рядом преимуществ: во-первых, Dilithium не использует

арифметику с плавающей точкой и дизайн схемы позволяет применять маскирование и выполнение операций за константное время [45, 69], что обеспечивает определённый уровень защиты от атак по побочным каналам; во-вторых, схему Dilithium легче программно реализовать [45, 69] безопасным образом, то есть обеспечив защиту от атак по побочным каналам, а также легче отслеживать ошибки реализации. Главное преимущество Dilithium — это баланс между размерами ключей и подписей и временем работы алгоритмов, что позволяет схеме быть универсальной. В Falcon, а также во многих других постквантовых схемах, в отличие от Dilithium, упор делается на один из показателей в ущерб другим, что сужает круг ситуаций, в которых целесообразно использовать данную схему. Сейчас схема Dilithium начинает активно внедряться в различные информационные системы в качестве альтернативы классическим схемам цифровой подписи, так, например, исследователи компании Google предлагают использовать Dilithium и SPHINCS+ в качестве основных схем цифровой подписи [71].

Схема HAETAE похожа на Dilithium, однако в силу нововведений вырабатывает подписи меньших размеров, чем Dilithium. Программная реализация схемы требует значительных доработок в целях повышения скорости выполнения алгоритмов, однако без учёта производительности данная схема выглядит перспективно и может в будущем заменить Dilithium.

## ЛИТЕРАТУРА


1. *Shor P. W.* Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer // SIAM Rev. 1995. No. 41. P. 303–332.
2. *Ajtai M., Kumar R., and Sivakumar D.* A sieve algorithm for the shortest lattice vector problem // Proc. STOC'01. Hersonissos, Greece, 2001. P. 601–610.
3. *Dinur I., Kindler G., and Safra S.* Approximating CVP to within almost-polynomial factors is NP-hard // Combinatorica. 2003. V. 23. P. 205–243.
4. *Roy S. S., Vercauteren F., Mentens N., et al.* Compact ring-LWE cryptoprocessor // LNCS. 2014. V. 8731. P. 371–391.
5. *Ajtai M.* Generating hard instances of lattice problems // Proc. STOC'96. Philadelphia, Pennsylvania, USA, 1996. P. 99–108.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography // Proc. STOC'05. Baltimore, MD, USA. 2005. P. 84–93.
7. *Micciancio D.* Generalized compact knapsacks, cyclic lattices, and efficient one-way functions // Comput. Complexity. 2007. No. 16(4). P. 365–411.
8. *Peikert C. and Rosen A.* Lattices that admit logarithmic worst-case to average-case connection factors // Proc. STOC'07. San Diego, California, USA. 2007. P. 478–487.
9. *Lyubashevsky V. and Micciancio D.* Generalized compact knapsacks are collision resistant // LNCS. 2006. V. 4052. P. 144–155.
10. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures // LNCS. 2009. V. 5912. P. 598–616.
11. *Lyubashevsky V., Peikert C., and Regev O.* On ideal lattices and learning with errors over rings // J. ACM. 2010. V. 60. P. 43:1–43:35.
12. *Langlois A. and Stehlé D.* Worst-case to average-case reductions for module lattices // Des. Codes Cryptography. 2015. V. 75. P. 565–599.
13. *Goldreich O., Goldwasser S., and Halevi S.* Public-key cryptosystems from lattice reduction problems // LNCS. 1997. V. 1294. P. 112–131.
14. *Hoffstein J., Pipher J., and Silverman J. H.* NTRU: A ring-based public key cryptosystem // LNCS. 1998. V. 1423. P. 267–288.

15. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSign: Digital signatures using the NTRU lattice // LNCS. 2003. V. 2612. P. 122–140.
16. *Hoffstein J., Pipher J., and Silverman J. H.* NSS: An NTRU lattice-based signature scheme // LNCS. 2001. V. 2045. P. 211–228.
17. *Gentry C., Jonsson J., Stern J., and Szydlo M.* Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001 // LNCS. 2001. V. 2248. P. 1–20.
18. *Gentry C. and Szydlo M.* Cryptanalysis of the revised NTRU signature scheme // LNCS. 2002. V. 2332. P. 299–320.
19. *Nguyen P. Q. and Regev O.* Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures // J. Cryptology. 2009. No. 22(2). P. 139–160.
20. *Ducas L. and Nguyen P. Q.* Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures // LNCS. 2012. V. 7658. P. 433–450.
21. *Melchor C. A., Boyen X., Deneuville J.-C., and Gaborit P.* Sealing the leak on classical NTRU signatures // LNCS. 2014. V. 8772. P. 1–21.
22. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSIGN: Digital signatures using the NTRU lattice // LNCS. 2003. V. 2612. P. 122–140.
23. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. Cryptology ePrint Archive. Paper 2005/274. `https://eprint.iacr.org/2005/274`.
24. *Gentry C., Peikert C., and Vaikuntanathan V.* Trapdoors for hard lattices and new cryptographic constructions // Proc. STOC'08. Victoria, British Columbia, Canada, 2008. P. 197–206.
25. *Stehlé D. and Steinfeld R.* Making NTRU as secure as worst-case problems over ideal lattices // LNCS. 2011. V. 6632. P. 27–47.
26. *Lyubashevsky V.* Lattice signatures without trapdoors // LNCS. 2012. V. 7237. P. 738–755.
27. *Gama N. and Nguyen P. Q.* Predicting lattice reduction // LNCS. 2008. V. 4965. P. 31–51.
28. *Howgrave-Graham N.* A hybrid lattice-reduction and meet-in-the-middle attack against NTRU // LNCS. 2007. V. 4622. P. 150–169.
29. *May A. and Silverman J. H.* Dimension reduction methods for convolution modular lattices // LNCS. 2001. V. 2146. P. 110–125.
30. *Stehlé D. and Steinfeld R.* Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. Cryptology ePrint Archive. Paper 2013/004. `https://eprint.iacr.org/2013/004`.
31. *Boneh D., Dagdelen Ö., Fischlin M., et al.* Random oracles in a quantum world // LNCS. 2011. V. 7073. P. 41–69.
32. *Diffie W. and Hellman M. E.* New directions in cryptography // IEEE Trans. Inform. Theory. 1976. No. 22(6). P. 644–654.
33. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols // Proc. CCS'93. Fairfax, Virginia, USA, 1993. P. 62–73.
34. *Micciancio D. and Peikert C.* Trapdoors for lattices: Simpler, tighter, faster, smaller // LNCS. 2012. V. 7237. P. 700–718.
35. *Alwen J. and Peikert C.* Generating shorter bases for hard random lattices // Theory Comput. Syst. 2011. No. 48(3). P. 535–553.
36. *Peikert C. and Rosen A.* Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices // LNCS. 2006. V. 3876. P. 145–166.
37. *Fouque P.-A., Hoffstein J., Kirchner P., et al.* Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Specification v1.2. `https://falcon-sign.info/falcon.pdf`. 2020.

38. *Ducas L. and Prest T.* Fast Fourier orthogonalization // Proc. ISSAC'16. Waterloo, ON, Canada, 2016. P. 191–198.
39. *Klein P. N.* Finding the closest lattice vector when it's unusually close // Proc. SODA'00. San Francisco, California, USA, 2000. P. 937–941.
40. *Micciancio D. and Walter M.* Practical, predictable lattice basis reduction // LNCS. 2016. V. 9665. P. 820–849.
41. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems // LNCS. 1987. V. 263. P. 186–194.
42. *Abdalla M., An J. H., Bellare M., and Namprempre C.* From identification to signatures via the Fiat — Shamir transform: Minimizing assumptions for security and forward-security // LNCS. 2002. V. 2332. P. 418–433.
43. *Schnorr C. P.* Efficient signature generation by smart cards // J. Cryptology. 1991. V. 4. No. 3. P. 161–174.
44. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures // LNCS. 2009. V. 5912. P. 598–616.
45. *Ducas L., Kiltz E., Lepoint T., et al.* CRYSTALS-Dilithium: A lattice-based digital signature scheme // IACR Trans. Cryptographic Hardware and Embedded Systems. 2018. No. 1. P. 238–268.
46. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Interagency/Internal Report (NISTIR). 2022. Report 8413. https://csrc.nist.gov/pubs/ir/8413/upd1/final.
47. *Cheon J. H., Choe H., Devevey J., et al.* HAETAE: Shorter Lattice-Based Fiat — Shamir Signatures. Cryptology ePrint Archive. Paper 2023/624. https://eprint.iacr.org/2023/624.
48. *Devroye L.* General principles in random variate generation // Non-Uniform Random Variate Generation. N.Y.: Springer, 1986. P. 27–82.
49. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma // Proc. CCS'06. Alexandria, Virginia, USA, 2006. P. 390–399.
50. *Ducas L., Durmus A., Lepoint T., and Lyubashevsky V.* Lattice signatures and bimodal gaussians // LNCS. 2013. V. 8042. P. 40–56.
51. *Groot Bruinderink L., Hülsing A., Lange T., and Yarom Y.* Flush, Gauss, and reload — a cache attack on the BLISS lattice-based signature scheme // LNCS. 2016. V. 9813. P. 323–345.
52. *Espitau T., Fouque P. A., Gérard B., and Tibouchi M.* Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers // Proc. CCS'17. Dallas, Texas, USA, 2017. P. 1857–1874.
53. *Pessl P., Bruinderink L. G., and Yarom Y.* To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures // Proc. CCS'17. Dallas, Texas, USA, 2017. P. 1843–1855.
54. *Devevey J., Fawzi O., Passelègue A., and Stehlé D.* On rejection sampling in Lyubashevsky's signature scheme // LNCS. 2022. V. 13794. P. 34–64.
55. *Brakerski Z., Gentry C., and Vaikuntanathan V.* (Leveled) fully homomorphic encryption without bootstrapping // Proc. ITCS'12. Cambridge, Massachusetts, 2012. P. 309–325.
56. *Bai S. and Galbraith S. D.* An improved compression technique for signatures based on learning with errors // LNCS. 2014. V. 8366. P. 28–47.
57. *Kiltz E., Lyubashevsky V., and Schaffner C.* A concrete treatment of Fiat — Shamir signatures in the quantum random-oracle model // LNCS. 2018. V. 10822. P. 552–586.

58. *Don J., Fehr S., Majenz C., and Schaffner C.* Security of the Fiat — Shamir transformation in the quantum random-oracle model // LNCS. 2019. V. 11693. P. 356–383.
59. *Liu Q. and Zhandry M.* Revisiting post-quantum Fiat — Shamir // LNCS. 2019. V. 11693. P. 326–355.
60. *Devevey J., Fallahpour P., Passelègue A., and Stehlé D.* A detailed analysis of Fiat — Shamir with aborts // LNCS. 2023. V. 14085. P. 327–357.
61. *Schnorr C. P. and Euchner M.* Lattice basis reduction: Improved practical algorithms and solving subset sum problems // Math. Programming. 1994. V. 66. P. 181–199.
62. *Alkim E., Ducas L., Pöppelmann T., and Schwabe P.* Post-quantum key exchange: A new hope // Proc. SEC'16. Austin, TX, USA, 2016. P. 327–343.
63. *Becker A., Ducas L., Gama N., and Laarhoven T.* New directions in nearest neighbor searching with applications to lattice sieving // Proc. SODA'16. Arlington, Virginia, 2016. P. 10–24.
64. *Laarhoven T., Mosca M., and Van De Pol J.* Finding shortest lattice vectors faster using quantum search // Des. Codes Cryptography. 2015. V. 77. P. 375–400.
65. *Schnorr C. P.* Lattice reduction by random sampling and birthday methods // LNCS. V. 2607. P. 145–156.
66. *Chen Y.* Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Thése de doctorat. Paris, 2013. (in French)
67. *Howe J. and Westerbaan B.* Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7 // LNCS. 2023. V. 14064. P. 442–462.
68. *Vidaković M. and Miličević K.* Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments // Algorithms. 2023. V. 16. No. 11:518.
69. *Lyubashevsky V.* CRYSTALS-Dilithium Round 3 Presentation. `https://csrc.nist.gov/presentations/2021/crystals-dilithium-round-3-presentation`. 2021.
70. *Westerbaan B.* NIST's Pleasant Post-Quantum Surprise. `https://blog.cloudflare.com/nist-post-quantum-surprise/`. 2022.
71. *Schmieg S., Kolbl S., and Endignoux G.* Google's Threat Model for Post-Quantum Cryptography. `https://bughunters.google.com/blog/5108747984306176/google-s-threat-model-for-post-quantum-cryptography`. 2024.

## REFERENCES

1. *Shor P. W.* Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev., 1995, no. 41, pp. 303–332.
2. *Ajtai M., Kumar R., and Sivakumar D.* A sieve algorithm for the shortest lattice vector problem. Proc. STOC'01, Hersonissos, Greece, 2001, pp. 601–610.
3. *Dinur I., Kindler G., and Safra S.* Approximating CVP to within almost-polynomial factors is NP-hard. Combinatorica, 2003, vol. 23, pp. 205–243.
4. *Roy S. S., Vercauteren F., Mentens N., et al.* Compact ring-LWE cryptoprocessor. LNCS, 2014, vol. 8731, pp. 371–391.
5. *Ajtai M.* Generating hard instances of lattice problems. Proc. STOC'96, Philadelphia, Pennsylvania, USA, 1996, pp. 99–108.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography. Proc. STOC'05, Baltimore, MD, USA, 2005, pp. 84–93.
7. *Micciancio D.* Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput. Complexity, 2007, no. 16(4), pp. 365–411.

8. *Peikert C. and Rosen A.* Lattices that admit logarithmic worst-case to average-case connection factors. Proc. STOC'07, San Diego, California, USA, 2007, pp. 478–487.
9. *Lyubashevsky V. and Micciancio D.* Generalized compact knapsacks are collision resistant. LNCS, 2006, vol. 4052, pp. 144–155.
10. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures. LNCS, 2009, vol. 5912, pp. 598–616.
11. *Lyubashevsky V., Peikert C., and Regev O.* On ideal lattices and learning with errors over rings. J. ACM, 2010, vol. 60, pp. 43:1–43:35.
12. *Langlois A. and Stehlé D.* Worst-case to average-case reductions for module lattices. Des. Codes Cryptography, 2015, vol. 75, pp. 565–599.
13. *Goldreich O., Goldwasser S., and Halevi S.* Public-key cryptosystems from lattice reduction problems. LNCS, 1997, vol. 1294, pp. 112–131.
14. *Hoffstein J., Pipher J., and Silverman J. H.* NTRU: A ring-based public key cryptosystem. LNCS, 1998, vol. 1423, pp. 267–288.
15. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSign: Digital signatures using the NTRU lattice. LNCS, 2003, vol. 2612, pp. 122–140.
16. *Hoffstein J., Pipher J., and Silverman J. H.* NSS: An NTRU lattice-based signature scheme. LNCS, 2001, vol. 2045, pp. 211–228.
17. *Gentry C., Jonsson J., Stern J., and Szydlo M.* Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001. LNCS, 2001, vol. 2248, pp. 1–20.
18. *Gentry C. and Szydlo M.* Cryptanalysis of the revised NTRU signature scheme. LNCS, 2002, vol. 2332, pp. 299–320.
19. *Nguyen P. Q. and Regev O.* Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology, 2009, no. 22(2), pp. 139–160.
20. *Ducas L. and Nguyen P. Q.* Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. LNCS, 2012, vol. 7658, pp. 433–450.
21. *Melchor C. A., Boyen X., Deneuville J.-C., and Gaborit P.* Sealing the leak on classical NTRU signatures. LNCS, 2014, vol. 8772, pp. 1–21.
22. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSIGN: Digital signatures using the NTRU lattice. LNCS, 2003, vol. 2612, pp. 122–140.
23. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. Cryptology ePrint Archive, paper 2005/274. `https://eprint.iacr.org/2005/274`.
24. *Gentry C., Peikert C., and Vaikuntanathan V.* Trapdoors for hard lattices and new cryptographic constructions. Proc. STOC'08, Victoria, British Columbia, Canada, 2008, pp. 197–206.
25. *Stehlé D. and Steinfeld R.* Making NTRU as secure as worst-case problems over ideal lattices. LNCS, 2011, vol. 6632, pp. 27–47.
26. *Lyubashevsky V.* Lattice signatures without trapdoors. LNCS, 2012, vol. 7237, pp. 738–755.
27. *Gama N. and Nguyen P. Q.* Predicting lattice reduction. LNCS, 2008, vol. 4965, pp. 31–51.
28. *Howgrave-Graham N.* A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. LNCS, 2007, vol. 4622, pp. 150–169.
29. *May A. and Silverman J. H.* Dimension reduction methods for convolution modular lattices. LNCS, 2001, vol. 2146, pp. 110–125.
30. *Stehlé D. and Steinfeld R.* Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. Cryptology ePrint Archive, paper 2013/004. `https://eprint.iacr.org/2013/004`.

31. *Boneh D., Dagdelen Ö., Fischlin M., et al.* Random oracles in a quantum world. LNCS, 2011, vol. 7073, pp. 41–69.
32. *Diffie W. and Hellman M. E.* New directions in cryptography. IEEE Trans. Inform. Theory, 1976, no. 22(6), pp. 644–654.
33. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols. Proc. CCS'93, Fairfax, Virginia, USA, 1993, pp. 62–73.
34. *Micciancio D. and Peikert C.* Trapdoors for lattices: Simpler, tighter, faster, smaller. LNCS, 2012, vol. 7237, pp. 700–718.
35. *Alwen J. and Peikert C.* Generating shorter bases for hard random lattices. Theory Comput. Syst., 2011, no. 48(3), pp. 535–553.
36. *Peikert C. and Rosen A.* Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. LNCS, 2006, vol. 3876, pp. 145–166.
37. *Fouque P.-A., Hoffstein J., Kirchner P., et al.* Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Specification v1.2. `https://falcon-sign.info/falcon.pdf`, 2020.
38. *Ducas L. and Prest T.* Fast Fourier orthogonalization. Proc. ISSAC'16, Waterloo, ON, Canada, 2016, pp. 191–198.
39. *Klein P. N.* Finding the closest lattice vector when it's unusually close. Proc. SODA'00, San Francisco, California, USA, 2000, pp. 937–941.
40. *Micciancio D. and Walter M.* Practical, predictable lattice basis reduction. LNCS, 2016, vol. 9665, pp. 820–849.
41. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems. LNCS, 1987, vol. 263, pp. 186–194.
42. *Abdalla M., An J. H., Bellare M., and Namprempre C.* From identification to signatures via the Fiat — Shamir transform: Minimizing assumptions for security and forward-security. LNCS, 2002, vol. 2332, pp. 418–433.
43. *Schnorr C. P.* Efficient signature generation by smart cards. J. Cryptology, 1991, vol. 4, no. 3, pp. 161–174.
44. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures. LNCS, 2009, vol. 5912, pp. 598–616.
45. *Ducas L., Kiltz E., Lepoint T., et al.* CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptographic Hardware and Embedded Systems, 2018, no. 1, pp. 238–268.
46. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Interagency/Internal Report (NISTIR), 2022, Report 8413. `https://csrc.nist.gov/pubs/ir/8413/upd1/final`.
47. *Cheon J. H., Choe H., Devevey J., et al.* HAETAE: Shorter Lattice-Based Fiat — Shamir Signatures. Cryptology ePrint Archive, paper 2023/624, `https://eprint.iacr.org/2023/624`.
48. *Devroye L.* General principles in random variate generation. Non-Uniform Random Variate Generation, N.Y., Springer, 1986, pp. 27–82.
49. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma. Proc. CCS'06, Alexandria, Virginia, USA, 2006, pp. 390–399.
50. *Ducas L., Durmus A., Lepoint T., and Lyubashevsky V.* Lattice signatures and bimodal gaussians. LNCS, 2013, vol. 8042, pp. 40–56.
51. *Groot Bruinderink L., Hülsing A., Lange T., and Yarom Y.* Flush, Gauss, and reload — a cache attack on the BLISS lattice-based signature scheme. LNCS, 2016, vol. 9813, pp. 323–345.

52. *Espitau T., Fouque P. A., Gérard B., and Tibouchi M.* Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. Proc. CCS'17, Dallas, Texas, USA, 2017, pp. 1857–1874.
53. *Pessl P., Bruinderink L. G., and Yarom Y.* To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. Proc. CCS'17, Dallas, Texas, USA, 2017, pp. 1843–1855.
54. *Devevey J., Fawzi O., Passelègue A., and Stehlé D.* On rejection sampling in Lyubashevsky's signature scheme. LNCS, 2022, vol. 13794, pp. 34–64.
55. *Brakerski Z., Gentry C., and Vaikuntanathan V.* (Leveled) fully homomorphic encryption without bootstrapping. Proc. ITCS'12, Cambridge, Massachusetts, 2012, pp. 309–325.
56. *Bai S. and Galbraith S. D.* An improved compression technique for signatures based on learning with errors. LNCS, 2014, vol. 8366, pp. 28–47.
57. *Kiltz E., Lyubashevsky V., and Schaffner C.* A concrete treatment of Fiat — Shamir signatures in the quantum random-oracle model. LNCS, 2018, vol. 10822, pp. 552–586.
58. *Don J., Fehr S., Majenz C., and Schaffner C.* Security of the Fiat — Shamir transformation in the quantum random-oracle model. LNCS, 2019, vol. 11693, pp. 356–383.
59. *Liu Q. and Zhandry M.* Revisiting post-quantum Fiat — Shamir. LNCS, 2019, vol. 11693, pp. 326–355.
60. *Devevey J., Fallahpour P., Passelègue A., and Stehlé D.* A detailed analysis of Fiat — Shamir with aborts. LNCS, 2023, vol. 14085, pp. 327–357.
61. *Schnorr C. P. and Euchner M.* Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Programming, 1994, vol. 66, pp. 181–199.
62. *Alkim E., Ducas L., Pöppelmann T., and Schwabe P.* Post-quantum key exchange: A new hope. Proc. SEC'16, Austin, TX, USA, 2016, pp. 327–343.
63. *Becker A., Ducas L., Gama N., and Laarhoven T.* New directions in nearest neighbor searching with applications to lattice sieving. Proc. SODA'16, Arlington, Virginia, 2016, pp. 10–24.
64. *Laarhoven T., Mosca M., and Van De Pol J.* Finding shortest lattice vectors faster using quantum search. Des. Codes Cryptography, 2015, vol. 77, pp. 375–400.
65. *Schnorr C. P.* Lattice reduction by random sampling and birthday methods. LNCS, vol. 2607, pp. 145–156.
66. *Chen Y.* Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Thése de doctorat. Paris, 2013. (in French)
67. *Howe J. and Westerbaan B.* Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7. LNCS, 2023, vol. 14064, pp. 442–462.
68. *Vidaković M. and Miličević K.* Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments. Algorithms, 2023, vol. 16, no. 11:518.
69. *Lyubashevsky V.* CRYSTALS-Dilithium Round 3 Presentation. `https://csrc.nist.gov/presentations/2021/crystals-dilithium-round-3-presentation`, 2021.
70. *Westerbaan B.* NIST's Pleasant Post-Quantum Surprise. `https://blog.cloudflare.com/nist-post-quantum-surprise/`, 2022.
71. *Schmieg S., Kolbl S., and Endignoux G.* Google's Threat Model for Post-Quantum Cryptography. `https://bughunters.google.com/blog/5108747984306176/google-s-threat-model-for-post-quantum-cryptography`, 2024.

УДК 519.7 DOI 10.17223/20710410/67/3

# О ВОЗМОЖНОСТИ МОДИФИКАЦИИ АЛГОРИТМА КБ-256 С ТОЧКИ ЗРЕНИЯ ПОИСКА НЕВОЗМОЖНЫХ ПЕРЕХОДОВ РАЗНОСТЕЙ БЛОКОВ

А. Б. Чухно

*НИУ ВШЭ, г. Москва, Россия*

**E-mail:** achuhno@hse.ru

Наличие у блочного алгоритма шифрования невозможных переходов разностей блоков может приводить к эффективным методам восстановления секретного ключа. Для алгоритма КБ-256 ранее было найдено большое количество невозможных переходов разностей блоков. В данной работе рассматривается вопрос об изменении функции обратной связи с целью уменьшения числа итераций, на которые их можно распространить. Показано, что изменение количества суммируемых подблоков в функции обратной связи не сможет уменьшить максимальное число итераций, на которое распространяется невозможный переход разностей блоков. Для выделенного типа обобщённых сетей Фейстеля предложен общий подход к поиску разностей с вероятностью 1.

**Ключевые слова:** *КБ-256, невозможные переходы разностей блоков, циркулянтная матрица, схема Фейстеля.*

# ON THE POSSIBILITY OF MODIFYING THE KB-256 ALGORITHM FROM THE SEARCHING FOR IMPOSSIBLE DIFFERENTIALS VIEW POINT

A. B. Chuhno

*Higher School of Economics, Moscow, Russia*

The presence of impossible differentials in a block cipher algorithm can lead to efficient methods for recovering the secret key. A large number of impossible differentials have been found for the KB-256 algorithm. This paper considers the modification of the feedback function to reduce the number of iterations to which they can be extended. A general approach to finding differences with probability 1 is proposed. It is shown that changing the number of summable sub-blocks in the feedback function will not reduce the maximum number of iterations to which an infeasible differential can be extended.

## Введение

Анализ криптографических алгоритмов с использованием невозможных переходов разностей блоков предложен независимо друг от друга Э. Бихамом, А. Бирюковым и А. Шамиром [1] и Л. Кнудсеном [2]. Под невозможным переходом разностей блоков при фиксированном ключе понимается невозможность заданного значения разности выходных блоков при входных блоках с фиксированной разностью.

Наличие невозможных переходов разностей блоков позволяет восстанавливать последовательность итерационных ключей. При переборе итерационных ключей проверяются разности входных блоков и соответствующие им разности выходных блоков — если появился невозможный переход разностей блоков, то ключ отбрасывается.

Алгоритм блочного шифрования КБ в нескольких вариантах предложен впервые в работе [3], в [4] проведено сравнение быстродействия 32 итераций алгоритма КБ с алгоритмом Магма. Далее под алгоритмом КБ-256 понимается вариант алгоритма с входным блоком шифрования размером 256 бит. В работе [5] для алгоритма КБ-256 найдено большое количество невозможных переходов разностей блоков.

Таким образом, алгоритм КБ-256 имеет изъян. Возникает вопрос, можно ли изменить устройство итерационной функции, чтобы уменьшить число итераций, на которые распространяется невозможный переход разностей блоков?

## 1. Основные понятия

Обозначим: $V_n = \{0,1\}^n$.

Пусть имеется алгоритм блочного шифрования $F : V_n \times V_m \to V_n$, устроенный по итеративному принципу

$$F(x,K) = G_{K_1} \circ G_{K_2} \circ \ldots \circ G_{K_t}(x) = G(G(\ldots G(x,K_1)\ldots,K_{t-1}),K_t),$$

где $x \in V_n$ — блок открытого текста; $K \in V_m$ — ключ шифрования. Последовательность $K_1, K_2, \ldots, K_t$, $K_j \in V_h$, $j = 1,\ldots,t$ — итерационные ключи, полученные из ключа $K$.

Также обозначим через $F_j(x,K)$ — применение первых $j$ итераций:

$$F_j(x,K) = G_{K_1} \circ G_{K_2} \circ \ldots \circ G_{K_j}(x), \quad F_t(x,K) = F(x,K).$$

**Определение 1.** Разностным соотношением $\widehat{\alpha,\beta}^{F_j}$ для отображения $F_j$ с фиксированным ключом $K$ называется событие

$$\widehat{\alpha,\beta}^{F_j} = \{x : F_j(x,K) \oplus F_j(x \oplus \alpha, K) = \beta\},$$

его вероятность обозначим $\Pr\left[\widehat{\alpha,\beta}^{F_j}\right] = p_{\alpha,\beta} \geqslant 0$.

Невозможным переходом разностей блоков назовём разностное соотношение с вероятностью $p_{\alpha,\beta} = 0$.

Для вектора $\mathbf{v} \in V_n$ веса $k$ введём обозначение $\mathbf{v} = [i_1, i_2 \ldots, i_k]$, где $i_1, i_2 \ldots, i_k$ — номера координат, не равных нулю.

## 2. Алгоритм КБ-256

Алгоритм шифрования КБ-256 есть отображение $F : V_{256} \times V_{256} \to V_{256}$; открытый текст $P = (X_0^0, X_1^0, X_2^0, X_3^0, X_4^0, X_5^0, X_6^0, X_7^0)$, $X_j^0 \in V_{32}$, $j = 0,\ldots,7$; ключ шифрования $K = (K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7)$, $K_j \in V_{32}$, $j = 0,\ldots,7$.

Отображение $F$ представимо в виде композиции

$$F(P,K) = G_{16}(G_{15}(\ldots G_1(P,K),K)\ldots),K) = C,$$

где $G_j(\cdot,K)$ — раундовое отображение, $j = 1,\ldots,16$.

Обозначим: $\Sigma_j = X_1 \boxplus X_3 \boxplus X_4 \boxplus X_6 \boxplus X_7$ — модульная сумма пяти подблоков входного вектора для $j$-й итерации, где $\boxplus$ — сложение по модулю $2^{32}$ (эту сумму будем называть

промежуточной суммой); $T_{\lll 19}$ — циклический сдвиг вектора в сторону старших бит; $S$ — применение к двоичному вектору блока 4-битных подстановок [6]. Тогда

$$G_j\left((X_0, X_1, \ldots, X_7), K\right) = \big(X_1, X_2 \oplus T_{\lll 19}(S(\Sigma_j \boxplus q_1^j)), X_3, X_4,$$
$$X_5 \oplus T_{\lll 19}(S(\Sigma_j \boxplus q_2^j)), X_6, X_7, X_0 \oplus T_{\lll 19}(S(\Sigma_j \boxplus q_3^j))\big),$$

где $q_1^j, q_2^j, q_3^j$ — раундовые ключи $j$-й итерации; $\oplus$ — побитовый XOR. Схематично раундовое отображение приведено на рис. 1, где через $f$ обозначена композиция отображений: $f(X) = T_{\lll 19}(S(X))$.

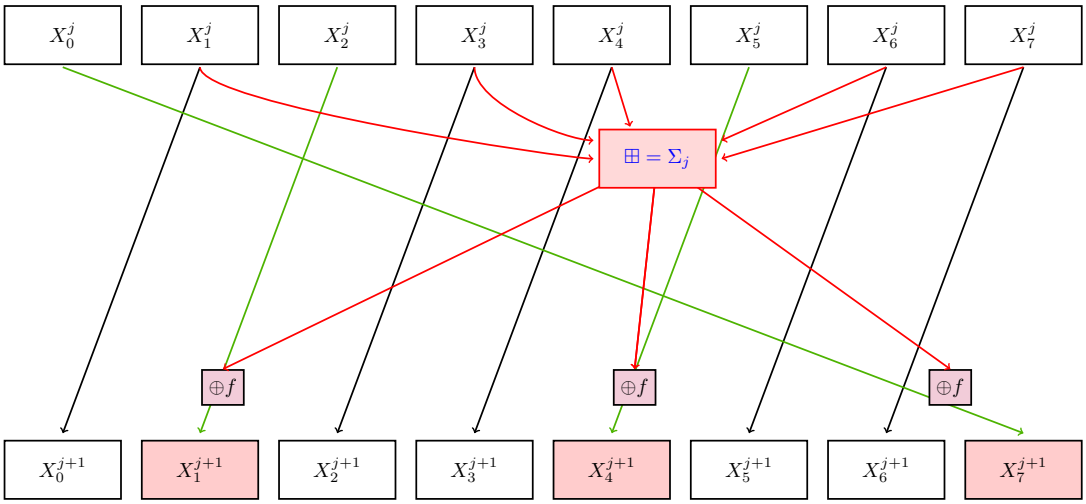


Рис. 1. Одна итерация алгоритма КБ-256

## 3. Циркулянтные матрицы с элементами из $\mathbb{F}_2$

Приведём вспомогательные факты о циркулянтных двоичных матрицах, которые будут использоваться в дальнейшем. Рассмотрим матрицу, образованную циклическим сдвигом строки $a_0, a_1, \ldots, a_{n-1}$ на одну позицию. Подобная матрица называется циркулянтной или просто циркулянтом и имеет вид

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & \ldots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & a_3 & a_4 & \ldots & a_{n-1} & a_0 \\ & & & \ldots & \ldots & & \\ a_{n-1} & a_0 & a_1 & a_2 & \ldots & a_{n-3} & a_{n-2} \end{pmatrix}. \tag{1}$$

В данной работе нас интересует случай, когда $a_i \in \mathbb{F}_2$, $i = 0, \ldots, n-1$.

**Утверждение 1.** Пусть дан двоичный вектор $(a_0, a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1})$ и вектор $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$, тогда

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & \ldots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & a_3 & a_4 & \ldots & a_{n-1} & a_0 \\ & & & \ldots & \ldots & & \\ a_{n-1} & a_0 & a_1 & a_2 & \ldots & a_{n-3} & a_{n-2} \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ \ldots \\ b_{n-1} \end{pmatrix} =$$
$$= \begin{pmatrix} b_0 & b_1 & b_2 & \ldots & b_{n-1} \\ b_{n-1} & b_0 & b_1 & \ldots & b_{n-2} \\ & & \ldots & & \\ b_1 & b_2 & b_3 & \ldots & b_0 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_{n-1} \end{pmatrix}.$$

***Доказательство.***

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & \dots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & a_3 & a_4 & \dots & a_{n-1} & a_0 \\ & & & \dots & \dots & & \\ a_{n-1} & a_0 & a_1 & a_2 & \dots & a_{n-3} & a_{n-2} \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{n-1} \end{pmatrix} =$$

$$= \begin{pmatrix} b_0a_0 \oplus b_1a_1 \oplus b_2a_2 \oplus b_3a_3 \oplus \dots \oplus b_{n-1}a_{n-1} \\ b_0a_1 \oplus b_1a_2 \oplus b_2a_3 \oplus b_3a_4 \oplus \dots \oplus b_{n-1}a_0 \\ \dots \\ b_0a_{n-1} \oplus b_1a_0 \oplus b_2a_1 \oplus b_3a_2 \oplus \dots \oplus b_{n-1}a_{n-2} \end{pmatrix} =$$

$$= a_0 \begin{pmatrix} b_0 \\ b_{n-1} \\ \dots \\ b_1 \end{pmatrix} \oplus a_1 \begin{pmatrix} b_1 \\ b_0 \\ \dots \\ b_2 \end{pmatrix} \oplus a_2 \begin{pmatrix} b_2 \\ b_1 \\ \dots \\ b_3 \end{pmatrix} \oplus \dots \oplus a_{n-1} \begin{pmatrix} b_{n-1} \\ b_{n-2} \\ \dots \\ b_0 \end{pmatrix} =$$

$$= \begin{pmatrix} b_0 & b_1 & b_2 & \dots & b_{n-1} \\ b_{n-1} & b_0 & b_1 & \dots & b_{n-2} \\ & & \dots & & \\ b_1 & b_2 & b_3 & \dots & b_0 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{pmatrix}.$$

Утверждение 1 доказано. ■

Определим многочлен $\varphi_{\mathbf{a}}(x) = a_0 \oplus a_1 x \oplus a_2 x^2 \oplus \dots \oplus a_{n-1}x^{n-1} \in \mathbb{F}_2[x]$. Известно [7, 8], что для обратимости циркулянтной матрицы, образованной строкой $a_0, a_1, \dots, a_{n-1}$, необходимо и достаточно, чтобы многочлен $\varphi_{\mathbf{a}}(x)$ был взаимно прост с многочленом $x^n \oplus 1$: $\text{НОД}(\varphi_{\mathbf{a}}(x), x^n \oplus 1) = 1$.

Если вес вектора $a_0, a_1, \dots, a_{n-1}$ чётный, то $(x \oplus 1) \,|\, \text{НОД}(\varphi_{\mathbf{a}}(x), x^n \oplus 1)$, поэтому матрица (1) необратима.

**Утверждение 2.** Пусть $n = 2^t$ и $A$ — циркулянтная матрица размера $n \times n$, образованная строкой $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ нечётного веса. Тогда матрица $A$ обратима.

***Доказательство.*** Для обратимости матрицы $A$ необходимо и достаточно подтвердить взаимную простоту многочленов $x^{2^t} \oplus 1$ и $\varphi_{\mathbf{a}}(x)$. Поскольку $x^{2^t} \oplus 1 = (x \oplus 1)^{2^t}$, то $(x \oplus 1)$ — его единственный неприводимый делитель, и он не делит $\varphi_{\mathbf{a}}(x)$, поскольку у последнего нечётное число мономов, а значит, 1 не является его корнем. ■

## 4. Невозможные переходы разностей блоков для алгоритма КБ-256 и возможность изменения функции обратной связи

### 4.1. Максимальное число итераций для разности с вероятностью 1

Построение невозможных переходов разностей блоков базируется на наличии разностных соотношений, проходящих с вероятностью 1. Сформулируем очевидное утверждение, с помощью которого строятся невозможные переходы разностей блоков.

**Утверждение 3.** Пусть для алгоритма $F(x, K) = F_t(x, K)$ имеется пара разностных соотношений $\widehat{\alpha, \beta}^{F_i}$ и $\widehat{\gamma, \lambda}^{F_j}$ с вероятностью 1, $i + j < t$. Если разностное соотношение $\widehat{\beta, \gamma}^{F_{t-i-j}}$ является невозможным, то соотношение $\widehat{\alpha, \lambda}^{F_t}$ тоже невозможно.

В работе [5] для построения невозможных переходов разностей блоков найдено разностное соотношение с вероятностью 1 на шесть итераций алгоритма КБ-256 и ещё одно на семь итераций. Для построения этих соотношений использовано следующее

свойство: модульное сложение векторов со значащими старшими битами ведет себя как $\oplus$. Поэтому брались векторы, различающиеся лишь в старшем бите.

Разности с вероятностью 1 для алгоритма КБ-256 имеют следующий вид: на шесть итераций — $(0,[31],0,0,0,0,[31],0) \to ([31],0,0,[31],0,0,0,0)$; на семь итераций — $([31],[31],0,0,0,0,0,[31]) \to ([31],[31],[31],0,0,0,0,0)$. Поскольку для каждого блока разность появляется лишь в старшем бите, то сопоставим разностям 8-битные векторы: $(0,1,0,0,0,0,1,0) \to (1,0,0,1,0,0,0,0)$ и $(1,1,0,0,0,0,0,1) \to (1,1,1,0,0,0,0,0)$; таким образом, каждому биту соответствует разность в старшем бите подблока текста.

Укажем явно это соответствие. Пусть $\mathbf{a} = (a_0,\dots,a_7) \in V_8$ — 8-битный вектор, $\mathbf{a} = [i_1,\dots,i_l]$, $l \leqslant 8$. Этому вектору соответствует вектор разностей в старших битах подблоков $(0,\dots,0,\underbrace{[31]}_{i_1},0,\dots,0,\underbrace{[31]}_{i_2},\underbrace{\dots}_{\dots},\underbrace{[31]}_{i_l},0,\dots,0)$.

Поскольку сложение старших битов вектора ведёт себя как побитовое, для описания распространения разностей рассмотрим битовый вектор $\mathbf{a} = (a_0,\dots,a_7) \in V_8$ — разность в старших битах для входных блоков — и набор индексов для вычисления промежуточной суммы $\mathbf{m} = (0,1,0,1,1,0,1,1) \in V_8$. Следующая система задаёт вычисление разностей в каждом из блоков последовательно от итерации к итерации:

$$\begin{cases} a_0\cdot 0 \oplus a_1\cdot 1 \oplus a_2\cdot 0 \oplus a_3\cdot 1 \oplus a_4\cdot 1 \oplus a_5\cdot 0 \oplus a_6\cdot 1 \oplus a_7\cdot 1 = 0,\\ a_1\cdot 0 + a_2\cdot 1 + a_3\cdot 0 + a_4\cdot 1 + a_5\cdot 1 + a_6\cdot 1 + a_7\cdot 0 + a_0\cdot 1 = 0,\\ a_2\cdot 0 + a_3\cdot 1 + a_4\cdot 0 + a_5\cdot 1 + a_6\cdot 1 + a_7\cdot 0 + a_0\cdot 1 + a_1\cdot 1 = 0,\\ a_3\cdot 0 + a_4\cdot 1 + a_5\cdot 0 + a_6\cdot 1 + a_7\cdot 1 + a_0\cdot 0 + a_1\cdot 1 + a_2\cdot 1 = 0,\\ a_4\cdot 0 + a_5\cdot 1 + a_6\cdot 0 + a_7\cdot 1 + a_0\cdot 1 + a_1\cdot 0 + a_2\cdot 1 + a_3\cdot 1 = 0,\\ a_5\cdot 0 + a_6\cdot 1 + a_7\cdot 0 + a_0\cdot 1 + a_1\cdot 1 + a_2\cdot 0 + a_3\cdot 1 + a_4\cdot 1 = 0,\\ a_6\cdot 0 + a_7\cdot 1 + a_0\cdot 0 + a_1\cdot 1 + a_2\cdot 1 + a_3\cdot 0 + a_4\cdot 1 + a_5\cdot 1 = 0,\\ a_7\cdot 0 + a_0\cdot 1 + a_1\cdot 0 + a_2\cdot 1 + a_3\cdot 1 + a_4\cdot 0 + a_5\cdot 1 + a_6\cdot 1 = 1. \end{cases}$$

Перепишем систему в матричном виде:

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7\\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0\\ a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1\\ a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2\\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3\\ a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4\\ a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5\\ a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} 0\\1\\0\\1\\1\\0\\1\\1 \end{pmatrix} = A\cdot\mathbf{m} = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix}. \tag{2}$$

Если системе (2) удовлетворяют вектор входных разностей и вектор для точек съёма промежуточной суммы, то данное заполнение даёт разностное соотношение с вероятностью 1 на семь итераций, поскольку сохранение сдвига нарушается впервые после седьмой итерации. Аналогично рассмотрим однородную систему

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 \\ a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 \\ a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 \\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 \\ a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 \\ a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = A \cdot \mathbf{m} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3}$$

Матрица, удовлетворяющая системе (3), позволит построить разностное соотношение с вероятностью 1 на любое число итераций.

В работе [5] начальных заполнений (циркулянтных матриц), удовлетворяющих данной системе, не было найдено.

На основе утверждения 1 системы (2) и (3) могут быть переписаны в виде

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Матрица $M$ — циркулянт размера $8 \times 8$, образованный строкой нечётного веса.

Возможны следующие случаи для нечётного веса: 1, 3, 5, 7. По утверждению 2 все такие матрицы являются обратимыми. Значит, найдётся набор входных разностей $\mathbf{a}$, порождающий разностное соотношение на семь итераций, и не найдётся никакого, отличного от нулевого, вектора разностей $\mathbf{a}$, который можно протянуть на восемь итераций и, как следствие, на любое число итераций.

Следовательно, найденная в работе [5] разность с вероятностью 1 на семь итераций — максимальная по числу итераций.

### 4.2. Изменение выбора подблоков для вычисления промежуточной суммы

Рассмотрим битовый вектор $\mathbf{a} = (a_0, \ldots, a_7) \in V_8$ (разность в старших битах для входных блоков) и набор индексов для вычисления промежуточной суммы $\mathbf{m} = (m_0, m_1, \ldots, m_7) \in V_8$. Для каждого вектора $\mathbf{m}$ и каждого начального заполнения $\mathbf{a}$ проведена экспериментальная проверка максимальной длины разностного соотношения с вероятностью 1.

В результате экспериментов выяснилось, что для максимального числа итераций для разностного соотношения с вероятностью 1 возможны две ситуации: оно равно семи итерациям или не ограничено.

Объяснение данных результатов связано с разрешимостью систем

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 \\ a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 \\ a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 \\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 \\ a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 \\ a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \end{pmatrix} = A \cdot \mathbf{m} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4}$$

или

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 \\ a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 \\ a_3 & a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 \\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 \\ a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 \\ a_6 & a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ a_7 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \end{pmatrix} = A \cdot \mathbf{m} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5}$$

Если начальное заполнение и вектор точек съёма удовлетворяют системе (4), то это даёт разность с вероятностью 1 на семь итераций. Если начальное заполнение и вектор точек съёма удовлетворяют системе (5), то получим разностное соотношение с вероятностью 1 на любое число итераций.

По утверждению 1 системы (4) и (5) можно переписать в виде

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 \\ m_7 & m_0 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 \\ m_6 & m_7 & m_0 & m_1 & m_2 & m_3 & m_4 & m_5 \\ m_5 & m_6 & m_7 & m_0 & m_1 & m_2 & m_3 & m_4 \\ m_4 & m_5 & m_6 & m_7 & m_0 & m_1 & m_2 & m_3 \\ m_3 & m_4 & m_5 & m_6 & m_7 & m_0 & m_1 & m_2 \\ m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_0 & m_1 \\ m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}; \tag{6}$$

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 \\ m_7 & m_0 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 \\ m_6 & m_7 & m_0 & m_1 & m_2 & m_3 & m_4 & m_5 \\ m_5 & m_6 & m_7 & m_0 & m_1 & m_2 & m_3 & m_4 \\ m_4 & m_5 & m_6 & m_7 & m_0 & m_1 & m_2 & m_3 \\ m_3 & m_4 & m_5 & m_6 & m_7 & m_0 & m_1 & m_2 \\ m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_0 & m_1 \\ m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{7}$$

В зависимости от количества точек съёма (веса вектора $\mathbf{m}$) циркулянт $M$ или вырожденный (четное число единиц), или обратимый (нечетное число единиц). Если матрица $M$ обратима, то система (7) имеет только нулевое решение, а для системы (6) всегда найдётся решение.

Если матрица $M$ вырождена, иначе — имеет чётное число единиц в строках, то система (7) имеет не только нулевое решение. А система (6) может вообще не иметь решений, да и их поиск теряет смысл, ведь найдётся разность на любое число итераций.

Значит, любой вариант выбора точек съёма не уменьшит число итераций, на которое можно распространить разностное соотношение с вероятностью 1. Это демонстрирует неулучшаемость данной схемы с точки зрения построения невозможных переходов разностей блоков.

## 5. Обобщённая схема Фейстеля и поиск невозможных переходов разностей блоков

Рассмотрим обобщённую схему Фейстеля со следующим устройством итерационного отображения $G(X, K)$.

Входной блок $X \in V_{tn}$ делится на $t$ блоков по $n$ бит: $X = (X_0, X_2, \ldots, X_{t-1})$. Для вычисления функции обратной связи берётся сумма $\sigma$ некоторых блоков, это может быть как модульное сложение, так и покоординатная сумма по модулю 2. Введём $(m_0, m_1, \ldots, m_{t-1})$ — вектор из нулей и единиц, задающий, какие блоки $X_i$ берутся для суммы: $\sigma = \sum\limits_{i=0}^{t-1} m_i X_i$. Результат одной итерации зашифрования вычисляется по следующему правилу:

$$G\big((X_0, X_1, \ldots, X_{t-1}), K\big) = \big(X_1 \oplus \overline{m_1} f(\sigma, K_1), \ldots, X_{t-1} \oplus \overline{m_{t-1}} f(\sigma, K_{t-1}), X_0 \oplus \overline{m_0} f(\sigma, K_0)\big).$$

Здесь $f(\cdot\,, K)$ — отображение, которое при фиксированном значении $K$ является биективным по первому аргументу; последовательность $K_0, K_1, \ldots, K_{t-1}$ — итерационные ключи.

Нетрудно видеть, что рассуждения из п. 4 применимы и для данной схемы, а именно: поиск разностных соотношений с вероятностью 1 связан с разрешимостью систем

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 & \ldots & m_{t-1} \\ m_{t-1} & m_0 & m_1 & m_2 & \ldots & m_{t-2} \\ \vdots & & & & & \\ m_1 & m_2 & m_3 & m_4 & \ldots & m_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 & \ldots & m_{t-1} \\ m_{t-1} & m_0 & m_1 & m_2 & \ldots & m_{t-2} \\ \vdots & & & & & \\ m_1 & m_2 & m_3 & m_4 & \ldots & m_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = M \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

В работе [3] предложены несколько конструкций блочного алгоритма шифрования на основе обобщённой схемы Фейстеля, одна из них — с входным блоком на 512 бит. Одна итерация этого алгоритма изображена на рис. 2.

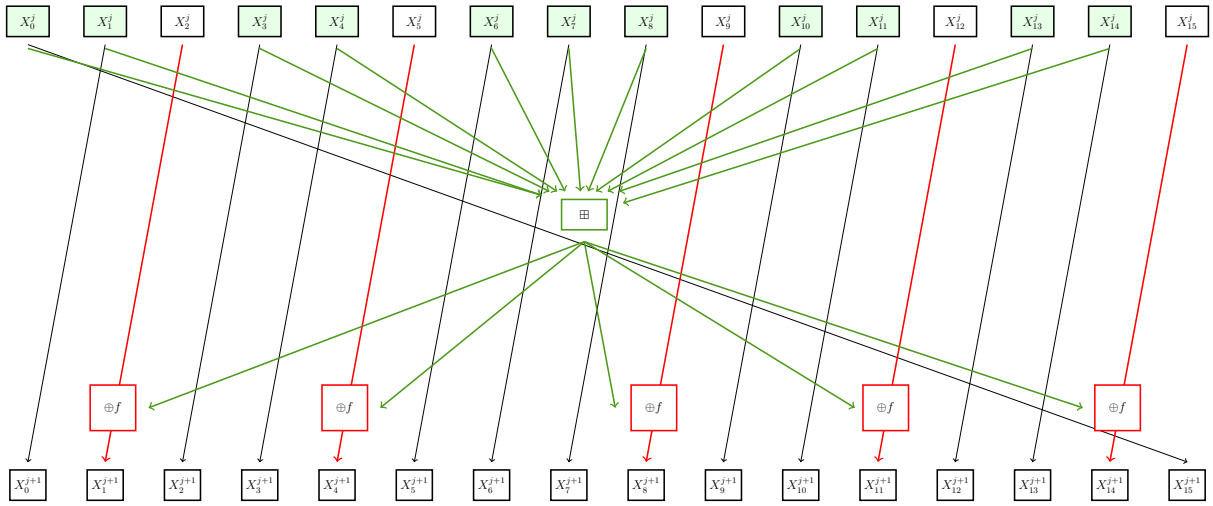

Рис. 2. Пример обобщённой схемы Фейстеля

В качестве функции $f$ изначально предполагалось использовать раундовую функцию алгоритма Магма [6]. У данного варианта алгоритма есть разностное соотношение с вероятностью 1 на 15 итераций, его вид приведён в таблице. Здесь $1$ — вектор из $V_{32}$, у которого 1 в старшем бите, а остальные равны нулю; $0$ — нулевой вектор из $V_{32}$.

| | |
|---|---|
| 0 | $(0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0)$ |
| 1 | $(0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0)$ |
| 2 | $(0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0)$ |
| 3 | $(0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0)$ |
| 4 | $(0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)$ |
| 5 | $(1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$ |
| 6 | $(1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1)$ |
| 7 | $(1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1)$ |
| 8 | $(0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1)$ |
| 9 | $(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0)$ |
| 10 | $(0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0)$ |
| 11 | $(0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0)$ |
| 12 | $(0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0)$ |
| 13 | $(0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0)$ |
| 14 | $(0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0)$ |
| 15 | $(0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0)$ |

На основе этого соотношения можно утверждать, что существует невозможный переход разностей не менее чем на 32 итерации. Непосредственной проверкой можно убедиться, что разностное соотношение

$$(0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0) \;\to\; (0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0)$$

является невозможным для двух итераций алгоритма (см. рис. 2); по утверждению 3 оно невозможно для 32 итераций.

## Заключение

Наличие разностных соотношений с вероятностью 1 позволяет строить невозможные переходы разностей блоков. Поэтому чем большее число итераций может пройти

разность с вероятностью 1, тем на большее число итераций может быть распространён невозможный переход разностей блоков (утверждение 3).

По результатам исследования выяснилось, что для алгоритма КБ-256 при любом сочетании суммирующихся блоков на итерации всегда найдётся разностное соотношение с вероятностью 1 не менее чем на семь итераций. Таким образом показана конструктивная неулучшаемость в плане уменьшения максимального числа итераций, для которой найдётся разность с вероятностью 1.

Автор выражает благодарность А. А. Дмуху за помощь в постановке задачи и ценные рекомендации в процессе её решения.

## ЛИТЕРАТУРА

1. *Biham E., Biryukov A., and Shamir A.* Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials // LNCS. 1999. V. 1592. P. 12–23.
2. *Knudsen L.* Deal — a 128-bit block cipher // Complexity. 1998. V. 258. No. 2.
3. *Fomichev V. M., Koreneva A. M., Miftakhutdinova A. R., et al.* Evaluation of the maximum performance of block encryption algorithms. // Матем. вопр. криптогр. 2019. Т. 10. № 2. С. 181–191.
4. *Fomichev V. M. and Koreneva A. M.* Encryption performance and security of certain wide block ciphers // J. Comput. Virol. Hack. Tech. 2020. V. 16. P. 197–216.
5. *Astrakhantsev R., Chuhno A., Dmukh A., et al.* Differences with high probability and impossible differentials for the KB-256 cipher // J. Comput. Virol. Hack. Tech. 2024. V. 20. P. 525–531.
6. ГОСТ 34.12-2018. Информационная технология. Криптографическая защита информации. Блочные шифры. М.: Стандартинформ, 2018.
7. *Guan Ph. and He Y.* Exact results for deterministic cellular automata with additive rules // J. Stat. Phys. 1986. V. 43. P. 463–478.
8. *Bini D., Del Corso G. M., Manzini G., and Margara L.* Inversion of circulant matrices over $\mathbb{Z}_m$ // LNCS. 1998. V. 1443. P. 719–730.

## REFERENCES

1. *Biham E., Biryukov A., and Shamir A.* Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. LNCS, 1999, vol. 1592, pp. 12–23.
2. *Knudsen L.* Deal — a 128-bit block cipher. Complexity, 1998, vol. 258, no. 2.
3. *Fomichev V. M., Koreneva A. M., Miftakhutdinova A. R., et al.* Evaluation of the maximum performance of block encryption algorithms. Matematicheskie Voprosy Kriptografii, 2019, vol. 10, no. 2, pp. 181–191.
4. *Fomichev V. M. and Koreneva A. M.* Encryption performance and security of certain wide block ciphers. J. Comput. Virol. Hack. Tech., 2020, vol. 16, pp. 197–216.
5. *Astrakhantsev R., Chuhno A., Dmukh A., et al.* Differences with high probability and impossible differentials for the KB-256 cipher. J. Comput. Virol. Hack. Tech., 2024, vol. 20, pp. 525–531.
6. GOST 34.12-2018. Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Blochnye shifry [GOST 34.12-2018. Information Technology. Cryptographic Protection of Information. Block Ciphers]. Moscow, Standartinform, 2018.
7. *Guan Ph. and He Y.* Exact results for deterministic cellular automata with additive rules. J. Stat. Phys., 1986, vol. 43, pp. 463–478.
8. *Bini D., Del Corso G. M., Manzini G., and Margara L.* Inversion of circulant matrices over $\mathbb{Z}_m$. LNCS, 1998, vol. 1443, pp. 719–730.

# МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 519.688 DOI 10.17223/20710410/67/4

## ВАЛИДАЦИЯ ПОВЕДЕНИЯ ПЕРЕДАТЧИКА В КАНАЛЕ ЧАСТИЧНОГО СТИРАНИЯ В СЛУЧАЕ ОТСУТСТВИЯ АБСОЛЮТНО РАЗЛИЧИМЫХ СИМВОЛОВ

И. Б. Казаков

*Российский экономический университет им. Плеханова, г. Москва, Россия*
*Московский государственный технический университет им. Баумана, г. Москва, Россия*

**E-mail:** i_b_kazakov@mail.ru

Работа принадлежит циклу работ, посвящённых каналу частичного стирания, где были введены понятия структуры частичного стирания, канала частичного стирания, правильной функции и корректного протокола. Структура частичного стирания — это тройка, состоящая из алфавита $A$, семейства его разбиений и множества вероятностей, приписанных этим разбиениям. Символы $a_1, a_2 \in A$ называются абсолютно различимыми в структуре частичного стирания, если не существует такого разбиения, что они принадлежат одному его классу. Канал частичного стирания функционирует следующим образом: Алиса посылает Бобу символ $a \in A$, но Боб узнаёт только, какое выбрано разбиение и какому классу принадлежит отправленный Алисой символ. Поведение передатчика в канале частичного стирания задаётся функцией $F : S^* \to A^*$, где $S$ — алфавит входной ленты, с которой передатчик считывает информацию. Функция $F$ однозначно определяет детерминированную функцию $\hat{F} : S^* \to A^*$: для любого слова $\hat{s} \in S^*$, $\hat{s} = s_1 \dots s_m$, положим $\hat{F}(\hat{s}) = F(\Lambda)F(s_1)F(s_1s_2)\dots F(s_1\dots s_m)$, где $\Lambda$ — пустое слово. Функция $\hat{F}$ может быть представлена в виде автомата с входным алфавитом $S$ и выходным алфавитом $A^*$. Автомат задаётся графом, вершины которого соответствуют состояниям, а рёбра имеют две подписи: $s \in S$ и $\alpha \in A^*$. Для существования корректного протокола, включающего в себя функцию $F$ поведения передатчика, необходимо и достаточно принадлежности функции $F$ к классу правильных. Функция является правильной тогда и только тогда, когда отсутствует аномалия 2-го рода, не являющаяся также и аномалией 1-го рода. Под аномалией 2-го рода понимается пара слов $(\hat{s}_1, \hat{s}_2)$, такая, что $|\hat{s}_2| = |\hat{s}_1| + 1$, $|\hat{F}(\hat{s}_2)| \leqslant |\hat{F}(\hat{s}_1)|$; под аномалией 1-го рода — пара слов $(\hat{s}_1, \hat{s}_2)$, для которой найдётся индекс $i$, что символы $\hat{F}(\hat{s}_1)[i]$ и $\hat{F}(\hat{s}_2)[i]$ абсолютно различимы в структуре частичного стирания. В важном частном случае отсутствия абсолютно различимых символов аномалии 1-го рода не могут существовать и условие правильности упрощается до отсутствия аномалий 2-го рода. Показано, что проверка отсутствия аномалий 2-го рода сводится к проверке отсутствия путей (начинающихся в выделенной вершине) отрицательного веса в автомат-квадратном графе, которая может быть выполнена с помощью алгоритма Беллмана — Форда. Общая сложность такой проверки составляет $O(|Q|^4|S|^2)$ по времени и $O(|Q|^2|S|^2 \ln|Q| \ln L \ln|S|)$ по памяти, где $|Q|$ — количество состояний автомата, представляющего функцию $\hat{F}$, и $L$ — максимальная длина передаваемого Алисой слова.

**Ключевые слова:** *скрытые каналы, канал частичного стирания, абсолютно различимые символы, алгоритм Беллмана — Форда, проверка отсутствия аномалий 2-го рода.*

# VALIDATION OF THE TRANSMITTER'S BEHAVIOUR IN THE PARTIAL ERASURE CHANNEL IN THE CASE OF ABSENCE OF ABSOLUTELY DISTINGUISHABLE CHARACTERS

I. B. Kazakov

*Plekhanov Russian University of Ecomonics, Moscow, Russia*
*Bauman Moscow State Technical University, Moscow, Russia*

This paper belongs to a series of papers devoted to the partial erasure channel, in which the concepts of partial erasure structure, partial erasure channel, correct function, and correct protocol have been introduced. The partial erasure structure is a triple consisting of the alphabet $A$, a family of partitions of this alphabet, and a set of probabilities attributed to the partitions. The characters $a_1, a_2 \in A$ are called absolutely distinguishable if there is no partition such that they belong to the same class of this partition. The partial erasure channel functions as follows. Alice sends Bob a symbol $a \in A$, but Bob receives only partial information about the symbol. Bob only knows which partition has been choosen and which partition class the symbol belongs to. The behavior of the transmitter in the partial erase channel can be described by specifying a function $F : S^* \to A^*$, where $S$ is the alphabet of the input tape from which the transmitter reads information. The function $F$ uniquely defines the deterministic function $\hat{F} : S^* \to A^*$ as follows: for any word $\hat{s} \in S^*$, $\hat{s} = s_1 \dots s_m$, let $\hat{F}(\hat{s}) = F(\Lambda)F(s_1)F(s_1 s_2) \dots F(s_1 \dots s_m)$, where $\Lambda$ is the empty word. The function $\hat{F}$ can be represented as an automaton with the input alphabet $S$ and the output alphabet $A^*$. The automaton, in turn, is represented as a graph whose vertices correspond to states and edges have two signatures: the symbol $s \in S$ and the word $\alpha \in A^*$. For the existence of a correct protocol that includes the function $F$ of the transmitter behavior, it is necessary and sufficient that the function $F$ belongs to the class of correct functions. A function is correct if and only if there is no anomaly of the second kind, which is not an anomaly of the first kind. The anomaly of the second kind is a pair of words $(\hat{s}_1, \hat{s}_2)$ such that $|\hat{s}_2| = |\hat{s}_1| + 1$, $|\hat{F}(\hat{s}_2)| \leqslant |\hat{F}(\hat{s}_1)|$. The anomaly of the first kind is a pair of words $(\hat{s}_1, \hat{s}_2)$ such that there is an index $i$ such that the characters $\hat{F}(\hat{s}_1)[i]$ and $\hat{F}(\hat{s}_2)[i]$ are absolutely distinguishable. In the important special case of the absence of absolutely distinguishable symbols, anomalies of the first kind cannot exist. Therefore, the correctness condition is simplified to the absence of anomalies of the second kind. In this paper, it is shown that checking for the absence of anomalies of the second kind is reduced to checking for the absence of paths (starting at the selected vertex) of negative weight in the so-called automaton-square graph. This absence can be detected by the Bellman — Ford algorithm. The total complexity of checking for the absence of anomalies of the second kind is $O(|Q|^4|S|^2)$ in time and $O(|Q|^2|S|^2 \ln|Q| \ln L \ln|S|)$ in memory, where $|Q|$ is the number of states of the automaton representing the function $\hat{F}$, $L$ is the maximum length of the word Alice throws out.

**Keywords:** *covert channels, partial erasure channel, absolutely distinguishable characters, Bellman — Ford algorithm, checking of the absence of an anomaly of the second kind.*

## Введение

В работе представлена процедура валидации, которая по формальному описанию поведения передатчика (традиционно называемого Алисой) в канале частичного стирания проверяет отсутствие аномалий 2-го рода для функции, описывающей данное поведение. В предыдущей работе [1] описан алгоритм проверки принадлежности функции к определённому в [2] классу правильных функций и установлен критерий правильности: функция поведения Алисы $F$ является правильной тогда и только тогда, когда всякая её аномалия 2-го рода является также и аномалией 1-го рода.

Особый интерес представляет частный случай, в котором в структуре частичного стирания не имеется абсолютно различимых символов и, следовательно, у функции поведения не может быть аномалий 1-го рода. В этом случае для проверки принадлежности функции поведения к классу правильных необходимо и достаточно проверить отсутствие аномалий 2-го рода. В настоящей работе излагается процедура, проверяющая принадлежность функции поведения Алисы к классу правильных в этом частном случае.

Представим структуру дальнейшего изложения. В п. 1 воспроизведены изложенные в [1, 2] сведения, задающие контекст настоящей работы. К ним относятся понятия структуры и канала частичного стирания, формализации поведения приемника и передатчика, определения аномалий 1-го и 2-го рода, а также определение правильной функции поведения передатчика.

Поведение передатчика представлено функцией $\hat{F}$, которая, в свою очередь, может быть задана графом, называемым автоматным. Этот граф является входными данными процедуры валидации. Построение автоматного графа по соответствующей функции поведения передатчика представлено в п. 2.

Процедура валидации, проверяющая отсутствие аномалий 2-го рода, описана в п. 3, доказательства сформулированных здесь утверждений приведены в п. 4 и 5.

## 1. Канал частичного стирания

В работе [2] введены понятия структуры и канала частичного стирания.

Структура частичного стирания — это тройка, состоящая из алфавита $A$, семейства определённых на данном алфавите разбиений и набора вероятностей, приписанных разбиениям. Канал частичного стирания функционирует следующим образом. Алиса отправляет Бобу символ $a \in A$, а Боб получает только часть информации: какое выбрано разбиение и какому классу этого разбиения принадлежит отправленный символ. Множество пар, состоящих из разбиения и его класса, т. е. множество возможных ответов, которые может получить Боб, обозначается как алфавит Боба $B$.

Полагаем, что у Алисы имеется входная лента, с которой она читает символы некоторого конечного алфавита $S$. У Боба, в свою очередь, имеется выходная лента, на которую он может печатать символы алфавита $S$, а также специально зарезервированный (не входящий в $S$) символ стирания «$*$».

Канал частичного стирания изучается в связи со скрытым каналом, использующим блуждания по плоскости. Описание скрытого канала блужданий по плоскости, а также того, каким образом он связан с каналом частичного стирания, представлено в [3–5].

В работе [2] рассмотрены формализованные схемы организации передачи информации от Алисы к Бобу, названные протоколами передачи информации в канале частичного стирания. Воспроизведём соответствующие понятия.

Поведение Алисы описывается функцией $F : S^* \to A^*$, а поведение Боба — функцией $G : B^* \to S \cup \{*\}$, где $A^*, B^*, S^*$ — множества всех слов (т. е. конечных последователь-

ностей символов, включая последовательность нулевой длины $\Lambda$, называемую «пустым словом») алфавитов $A, B, S$ соответственно. Протокол — это пара функций $(F, G)$. Длину слова $\alpha \in A^*$ будем обозначать как $|\alpha|$. Функция $F$ однозначно определяет детерминированную функцию $\hat{F} : S^* \to A^*$ следущим образом: для любого слова $\hat{s} \in S^*$, $\hat{s} = s_1 \dots s_m$, положим $\hat{F}(\hat{s}) = \hat{F}(s_1 \dots s_m) = F(\Lambda)F(s_1)F(s_1 s_2) \dots F(s_1 \dots s_m)$.

Представим содержательную интерпретацию, относящуюся к функциям $F$ и $G$. Пусть Алиса только что прочитала с входной ленты очередной символ $s \in S$, предварительно уже считав слово $\hat{s} = s_1 \dots s_m$. Тогда считаем, что на протяжении последующих $|F(\hat{s}s)|$ тактов Алиса посимвольно отправляет по каналу частичного стирания слово $F(\hat{s}s)$. Что касается Боба, то на каждом такте он получает символ $b \in B$ и должен принять решение о том, что печатать на выходной ленте: или какой-нибудь символ из множества $S \cup \{*\}$, или ничего не печатать. Пусть $\beta = b_1 \dots b_n \in B^*$ — слово, состоящее из символов, последовательно полученных Бобом на предыдущих (включая текущий) тактах. Полагаем, что если $G(\beta) \in S \cup \{*\}$, то Боб печатает символ $G(\beta)$. Иначе, т. е. если $G(\beta) = \Lambda$, Боб ничего не печатает.

Общей целью Алисы и Боба является печать на выходной ленте того же содержимого, которое изначально имеется на входной, при этом, может быть, с заменой значащих символов (т. е. символов алфавита $S$) на символ стирания «$*$». Протоколы, соответствующие этой цели, в [2] названы корректными. Если пара $(F, G)$ является корректным протоколом, то будем говорить, что функция поведения Боба $G$ согласована с функцией поведения Алисы $F$.

Основным вопросом, исследованным в [2], является вопрос о том, для каких функций поведения Алисы $F$ существует согласованная с ней функция поведения Боба $G$. Ответом является теорема, что для этого необходимо и достаточно принадлежности функции поведения Алисы $F$ к классу правильных. В [2] дано определение правильной функции.

В работе [1] приведён алгоритм проверки принадлежности функции $F$ классу правильных и представлен критерий правильности, который можно взять за эквивалентное определение. Воспроизведём его формулировку:

**Определение 1.** Символы $a_1, a_2 \in A$ называются абсолютно различимыми в структуре частичного стирания, если в ней не найдётся такого разбиения, что $a_1, a_2$ принадлежат одному классу. Будем говорить, что слова $\alpha_1, \alpha_2$ имеют общую абсолютно различимую позицию, если найдётся такая позиция $i$, что символы $\alpha_1[i], \alpha_2[i]$ абсолютно различимы в соответствующей структуре частичного стирания.

**Определение 2.** Пара слов $(\hat{s}_1, \hat{s}_2)$ называется аномалией 1-го рода для функции $F$, если слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$ имеют общую абсолютно различимую позицию.

**Определение 3.** Пара слов $(\hat{s}_1, \hat{s}_2)$ называется аномалией 2-го рода для функции $F$, если $\big(|\hat{s}_1| < |\hat{s}_2|$ и $|\hat{F}(\hat{s}_1)| \geqslant |\hat{F}(\hat{s}_2)|\big)$ или $\big(|\hat{s}_1| > |\hat{s}_2|$ и $|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|\big)$.

Обозначим через $S_{\text{pairs}}$ множество всех пар слов $(\hat{s}_1, \hat{s}_2) \in (S^*)^2$, таких, что выполнено $|\hat{s}_1| = |\hat{s}_2| + 1$.

**Определение 4.** Функция поведения Алисы $F$ называется правильной, если всякая её лежащая в $S_{\text{pairs}}$ аномалия 2-го рода является также и аномалией 1-го рода.

Особый интерес представляет частный случай отсутствия абсолютно различимых символов в структуре частичного стирания. В данном случае у функции поведения Алисы $F$ не может быть аномалий 1-го рода и, следовательно, проверка на принадлежность функции классу правильных сводится к проверке отсутствия в $S_{\text{pairs}}$ аномалий 2-го рода.

## 2. Автоматный граф

Представим функцию $\hat{F}$ в виде (не обязательно конечного) автомата, у которого $S$ является входным алфавитом, а множество слов $A^*$ — выходным: автомат на каждом шаге принимает символ $s \in S$ и выдаёт слово $\alpha \in A^*$. Действительно, рассмотрим бесконечный автомат, состояния которого находятся во взаимно однозначном соответствии с элементами множества $S^*$. Находясь в состоянии $\hat{s} \in S^*$ и приняв символ $s \in S$, автомат переходит в состояние $\hat{s}s$ и выдаёт слово $F(\hat{s}s)$. Подвергая автомат преобразованию, отождествляющему его неразличимые состояния, получаем приведённый автомат, представляющий функцию $\hat{F}$.

Множество состояний полученного автомата обозначим через $Q$. Автомат имеет функцию переходов $\phi : Q \times S \to Q$ и выходную функцию $\psi : Q \times S \to A^*$, а также начальное состояние $q_\Lambda$. Через $q_{\hat{s}}$ обозначим состояние, в которое автомат переходит, приняв на вход слово $\hat{s} = s_1 \dots s_m$, т. е. последовательно приняв символы $s_1 \dots s_m$. В соответствии с этим определением выполнено $\phi(q_{\hat{s}}, s) = q_{\hat{s}s}$.

Отметим, что все состояния автомата достижимы, т. е. для любого $q \in Q$ найдётся такое $\hat{s} \in S^*$, что $q = q_{\hat{s}}$. Достижимость всех состояний автомата следует из того, что он получен посредством отождествления неразличимых состояний первоначального бесконечного автомата, у которого все состояния достижимы по построению. Таким образом, функция, ставящая в соответствие слову $\hat{s} \in S^*$ состояние $q_{\hat{s}} \in Q$, является сюръекцией. Однако данная функция не обязательно является инъекцией: если состояния первоначального автомата, сооответствующие словам $\hat{s}_1$, $\hat{s}_2$, неразличимы, то выполнено $q_{\hat{s}_1} = q_{\hat{s}_2}$.

Соответствующим образом определена и выходная функция $\psi(q_{\hat{s}}, s) = F(\hat{s}s)$. Значение $F(\Lambda)$, отсутствующее в описании автомата, фиксировано и хранится отдельно.

Автомат представим в виде ориентированного графа, вершины которого соответствуют элементам множества $Q$, а у каждого ребра есть две подписи: символ алфавита $S$ и слово из $A^*$. Из каждой вершины $q \in Q$ выходят $|S|$ рёбер, каждое из которых подписано соответствующим символом $s \in S$ и словом $\psi(q, s)$. Граф имеет выделенную инициальную вершину $q_\Lambda$.

**Определение 5.** Граф, представляющий автомат и, следовательно, функцию поведения Алисы $\hat{F}$, будем называть автоматным графом функции $\hat{F}$.

В настоящей работе, как и в [1], рассматриваются только ограниченно-детерминированные функции поведения Алисы $\hat{F}$, т. е. только такие функции, автоматный граф которых содержит конечное число вершин.

## 3. Процедура валидации

Представим процедуру, проверяющую отсутствие лежащих в $S_{\text{pairs}}$ аномалий 2-го рода. Её работа состоит из двух этапов. На первом этапе по автоматному графу строится взвешенный граф, далее называемый автомат-квадратным. На втором этапе проверяется наличие непустых путей неположительного веса в построенном автомат-квадратном графе, эта проверка осуществляется алгоритмом Беллмана — Форда [6, 7].

Построение автомат-квадратного графа описано в п. 3.1, а алгоритм Беллмана — Форда — в п. 3.2. В п. 3.1 и 3.2 также приведены теоремы 2 и 3 соответственно.

В соответствии с теоремой 2 наличие лежащих в $S_{\text{pairs}}$ аномалий 2-го рода равносильно наличию в автомат-квадратном графе начинающихся в инициальной вершине непустых путей неположительного веса. По теореме 3, алгоритм Беллмана — Форда выдаёт ответ «да» тогда и только тогда, когда в автомат-квадратном графе не име-

ется начинающихся в инициальной вершине непустых путей неположительного веса. Таким образом, выполнено утверждение о корректности работы процедуры проверки:

**Теорема 1.** Процедура проверки выдаёт ответ «да» тогда и только тогда, когда представленная автоматным графом функция поведения Алисы не имеет аномалий 2-го рода.

Заключительный п. 3.3 посвящён оценкам сложности алгоритма по времени и по памяти.

### 3.1. Автомат-квадратный граф

На основе автоматного графа построим взвешенный ориентированный граф, каждое ребро которого, кроме числовой метки (т. е. веса ребра), имеет буквенную метку из множества $(S \cup \{*\})^2$. Опишем вершины, рёбра и метки, тем самым определим граф, далее называемый автомат-квадратным графом функции поведения Алисы.

**Определение 6.** Автомат-квадратным графом для функции $F$ называется ориентированный граф, вершины, рёбра, буквенные и числовые метки которого определены следующим образом:

1) Имеется $|Q|^2 + 1$ вершин: $|Q|^2$ вершин, соответствующих элементам множества $Q^2$, и одна дополнительная вершина, далее называемая инициальной.
2) Из каждой неинициальной вершины $(q_1, q_2) \in Q^2$ выходит $|S|^2$ рёбер, каждое из которых подписано буквенной меткой $(s_1, s_2) \in S^2$. Из инициальной вершины выходит $|S|$ рёбер, каждое из которых подписано буквенной меткой $(s, *)$, $s \in S$.
3) Ребро, выходящее из вершины $(q_1, q_2) \in Q^2$ и имеющее буквенную метку $(s_1, s_2) \in S^2$, ведёт в вершину $(q_1', q_2') \in Q^2$, где $q_1' = \phi(q_1, s_1)$, $q_2' = \phi(q_2, s_2)$, и подписано также числовой меткой $|\psi(q_1, s_1)| - |\psi(q_2, s_2)|$. Ребро, выходящее из инициальной вершины и имеющее буквенную метку $(s, *)$, ведёт в вершину $(\phi(q_\Lambda, s), q_\Lambda)$ и подписано также числовой меткой $|\psi(q_\Lambda, s)|$.

Множество вершин автомат-квадратного графа обозначим через $V$, множество рёбер — через $E$. Множество путей, начинающихся в инициальной вершине, обозначим через $P$, а через $P_0$ — множество путей из $P$, не содержащих циклы. В множества $P$ и $P_0$ включим также пустой путь $\Lambda$ длины 0. Весом пути будем называть сумму весов составляющих его рёбер.

**Теорема 2.** Следующие условия равносильны:

1) существует пара $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$, являющаяся аномалией 2-го рода;
2) в автомат-квадратном графе существует путь $\pi \in P \setminus \{\Lambda\}$ неположительного веса.

Доказательство теоремы приведено в п. 4.

Таким образом, проверка функции $F$ на отсутствие лежащих в $S_{\text{pairs}}$ аномалий 2-го рода (и, следовательно, на принадлежность классу правильных функций в случае отсутствия абсолютно различимых символов) сводится к проверке автомат-квадратного графа на отсутствие пути $\pi \in P \setminus \{\Lambda\}$ неположительного веса.

Вес рёбер автомат-квадратного графа может быть отрицательным. Возможно также наличие цикла отрицательного веса. Все пути, которые требуется проверять на положительность веса, начинаются в инициальной вершине. Для решения поставленной задачи подходит алгоритм Беллмана — Форда.

### 3.2. Алгоритм Беллмана — Форда

Отметим, что в инициальную вершину автомат-квадратного графа не входят рёбра.

Проверим отсутствие начинающихся в инициальной вершине непустых путей неположительного веса с помощью алгоритма Беллмана — Форда. Сложность данного алгоритма составляет $O(|V||E|)$ по времени. Кроме весов рёбер, алгоритм хранит в памяти массив из $|V|$ числовых меток, приписаннных вершинам графа, сложность по памяти составляет $O(|V|)$. В соответствии с описанием автомат-квадратного графа числовые метки рёбер являются целыми числами.

Числовые метки вершин могут быть как целыми числами, так и элементами множества $\{+\infty, -\infty\}$. Введём обозначения: для $e \in E$ через $d(e)$ будем обозначать вес ребра, для $v \in V$ через $d(v)$ — числовую метку, приписанную вершине $v$; $d(\pi)$ — вес пути $\pi$, т. е. сумма весов рёбер, составляющих этот путь.

Алгоритм 1 проверяет отсутствие начинающихся в инициальной вершине непустых путей неположительного веса.

**Алгоритм 1.** Алгоритм Беллмана — Форда

1: Выполнить процедуру инициализации: инициальной вершине присвоить метку 0, а всем остальным вершинам — метку $+\infty$.
2: $|V| - 1$ раз выполнить процедуру регулярного обхода:
3: **Для всех** рёбер $e \in E$, $e : v \to w$,
выполнить процедуру регулярной обработки:
4: **Если** $d(v) + d(e) < d(w)$, **то**
$d(w) := d(v) + d(e)$.
5: Процедура финального обхода:
6: **Для всех** рёбер $e \in E$, $e : v \to w$,
выполнить процедуру финальной обработки:
7: **Если** $d(v) + d(e) < d(w)$, **то**
$d(w) := -\infty$.
8: Проверить, есть ли неинициальная вершина с неположительной числовой меткой. **Если** такая вершина найдётся, **то** выдать ответ «нет». **Иначе** выдать ответ «да».

**Теорема 3.** Следующие условия равносильны:

1) алгоритм 1 выдаёт ответ «да»;
2) не существует пути $\pi \in P \setminus \{\Lambda\}$ неположительного веса.

Доказательство теоремы приведено в п. 5.

Несмотря на то, что алгоритм Беллмана — Форда общеизвестен, считаем полезным представить подробное доказательство его корректности, т. е. доказать теорему 3. В работах [6, 7] указано, что если при исполнении последней итерации длина кратчайшего пути до какой-либо вершины строго уменьшилась, то в графе есть отрицательный цикл, достижимый из инициальной вершины, однако анализ этого обстоятельства не является достаточно подробным.

### 3.3. Оценки сложности

Каждое ребро автоматного графа является сущностью, состоящей из входной и выходной вершин, символьной метки $s \in S$ и метки $\alpha \in A^*$. Для построения автомат-квадратного графа требуются не сами слова из множества $A^*$, являющиеся метками, а только их длины. Поэтому на практике вместо метки ребра автоматного графа, являющейся словом, в памяти можно хранить числовую метку, равную длине слова. Следовательно, для хранения ребра автоматного графа требуется $O(\ln|Q| \ln|L| \ln|S|)$ бит

памяти, где $L$ обозначает максимальную длину слов из множества $A^*$, являющихся метками рёбер автоматного графа.

Каждое ребро автомат-квадратного графа, в свою очередь, является сущностью, состоящей из входной и выходной вершин, двух символьных меток $s_1, s_2 \in S \cup \{*\}$, а также числовой метки, значение которой лежит в интервале $[-L, L]$. Следовательно, для хранения ребра требуется $O(\ln|Q|\ln|2L+1|\ln|S+1|) = O(\ln|Q|\ln|L|\ln|S|)$ бит памяти.

Граф хранится в памяти в виде списка рёбер. Таким образом, для автоматного графа требуется $O(|Q||S|\ln|Q|\ln|L|\ln|S|)$ бит памяти, а для автомат-квадратного — $O((|Q|^2|S|^2+|S|)\ln|Q|\ln|L|\ln|S|) = O(|Q|^2|S|^2\ln|Q|\ln|L|\ln|S|)$ бит памяти.

На втором этапе дополнительно используется $O((|Q|^2+1)\ln|L|) = O(|Q|^2\ln|L|)$ бит памяти для хранения числовых меток вершин автомат-квадратного графа.

Построение каждого ребра автомат-квадратного графа занимает $O(1)$ времени, следовательно, сложность по времени первого этапа составляет $O(|Q|^2|S|^2+|S|) = O(|Q|^2|S|^2)$.

В соответствии с описанием в п. 3.2 сложность второго этапа по времени равна $O(|V||E|) = O((|Q|^2+1)(|Q|^2|S|^2+|S|)) = O(|Q|^4|S|^2)$.

Таким образом, общая сложность процедуры валидации по памяти составляет

$$O(|Q||S|\ln|Q|\ln|L|\ln|S|) + O(|Q|^2|S|^2\ln|Q|\ln|L|\ln|S|) + O(|Q|^2\ln|L|) = \\ = O(|Q|^2|S|^2\ln|Q|\ln|L|\ln|S|).$$

Общая сложность по времени: $O(|Q|^2|S|^2) + O(|Q|^4|S|^2) = O(|Q|^4|S|^2)$.

Сравним полученные оценки с оценками сложности алгоритма, представленного в [1]: его сложность по памяти составляет $O(L^2|Q|^3|S|^2\ln|Q|\ln|A|)$, а по времени — $O(L|Q|^3|S|^4)$. Таким образом, при не слишком большом числе $|Q|$ состояний проверка отсутствия лежащих в $S_{\text{pairs}}$ аномалий 2-го рода существенно менее затратна, чем проверка того, что всякая лежащая в $S_{\text{pairs}}$ аномалия 2-го рода является также аномалией 1-го рода.

## 4. Доказательство теоремы 2

В п. 4.1 построим взаимно однозначное соответствие между элементами множества $S_{\text{pairs}}$ и начинающимися в инициальной вершине путями (т. е. элементами множества $P \setminus \{\Lambda\}$) в соответствии с буквенными метками рёбер автомат-квадратного графа.

Далее в п. 4.2 докажем, что вес пути $\pi((\hat{s}_1, \hat{s}_2))$ равен $|\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|$, и тем самым установим, что аномалиям 2-го рода соответствуют начинающиеся в инициальной вершине непустые пути неположительного веса.

### 4.1. Биекция $\hat{\pi} : S_{\text{pairs}} \to P \setminus \{\Lambda\}$

Определим функцию $\hat{\pi} : S_{\text{pairs}} \to P \setminus \{\Lambda\}$.

Пусть $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$ — пара, для которой $|\hat{s}_2| = n$. Определим путь $\pi((\hat{s}_1, \hat{s}_2))$ как последовательность смежных рёбер $e_1, \ldots, e_{n+1}$.

В качестве $e_1$ возьмём ребро, исходящее из инициальной вершины и имеющее буквенную метку $(\hat{s}_1[1], *)$. Пусть построены рёбра $e_1, \ldots, e_k$. Обозначим вершину, в которую входит ребро $e_k$, как $v_k$. Так как в инициальную вершину не входят рёбра, то $v_k$ — неинициальная. В качестве $e_{k+1}$ возьмем ребро, выходящее из $v_k$ и имеющее буквенную метку $(\hat{s}_1[k+1], \hat{s}_2[k])$.

**Утверждение 1.** Функция $\hat{\pi} : S_{\text{pairs}} \to P \setminus \{\Lambda\}$ является сюръекцией.

***Доказательство.*** Зафиксируем путь $\pi \in P \setminus \{\Lambda\}$, т.е. произвольный непустой путь, начинающийся в инициальной вершине. Пусть путь $\pi$ является последовательностью смежных рёбер $e_1, \ldots, e_n$ и рёбра $e_1, \ldots, e_n$ имеют буквенные метки $(s_{1,1}, s_{1,2}), \ldots, (s_{n,1}, s_{n,2})$, где $s_{1,1}, \ldots, s_{n,1}, s_{1,2}, \ldots, s_{n,2} \in S \cup \{*\}$. Так как ребро $e_1$ исходит из инициальной вершины, то $s_{1,2} = *$. Рёбра $e_2, \ldots, e_n$ не исходят из инициальной вершины, поэтому все остальные символы $s_{i,j}$ принадлежат множеству $S$.

Положим $\hat{s}_1 = s_{1,1} \ldots s_{n,1} \in S^*$, т. е. $\hat{s}_1[i] = s_{i,1}$ для $i = 1, \ldots, n$.

Определим слово $\hat{s}_2 \in S^*$ следующим образом. Если $n = 1$, то $\hat{s}_2 = \Lambda$. Если $n > 1$, то $\hat{s}_2 = s_{2,2} \ldots s_{n,2}$, т. е. $\hat{s}_2[i] = s_{i+1,2}$ для $i = 1, \ldots, n-1$. По построению $|\hat{s}_1| = n$, $|\hat{s}_2| = n - 1$, т. е. $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$.

Положим $\pi' = \hat{\pi}((\hat{s}_1, \hat{s}_2)) \in P \setminus \{\Lambda\}$, пусть $\pi'$ является последовательностью смежных рёбер $e'_1, \ldots, e'_n$. Докажем по индукции, что $\pi' = \pi$, т. е. что для всех $i = 1, \ldots, n$ выполнено $e'_i = e_i$.

Рёбра $e_1, e'_1$ исходят из инициальной вершины. Ребро $e_1$ имеет буквенную метку $(s_{1,1}, s_{1,2}) = (s_{1,1}, *)$. По определению функции $\hat{\pi}$, а также слова $\hat{s}_1$ ребро $e'_1$ имеет буквенную метку $(\hat{s}_1[1], *) = (s_{1,1}, *)$. Так как рёбра $e_1, e'_1$ исходят из одной и той же вершины и имеют тождественные буквенные метки, они совпадают.

Предположим, что $e_k = e'_k$. Докажем, что из этого следует $e_{k+1} = e'_{k+1}$.

Рёбра $e_{k+1}, e'_{k+1}$ начинаются в вершинах, в которых заканчиваются рёбра $e_k, e'_k$, а так как $e'_k = e_k$, то рёбра $e_{k+1}, e'_{k+1}$ начинаются в одной и той же вершине.

Ребро $e_{k+1}$ имеет буквенную метку $(s_{k+1,1}, s_{k+1,2})$. По определению функции $\hat{\pi}$, а также слов $\hat{s}_1, \hat{s}_2$ ребро $e'_{k+1}$ имеет буквенную метку $(\hat{s}_1[k+1], \hat{s}_2[k]) = (s_{k+1,1}, s_{k+1,2})$. Так как рёбра $e_{k+1}, e'_{k+1}$ выходят из одной и той же вершины и имеют тождественные буквенные метки, они совпадают.

Таким образом, $\pi((\hat{s}_1, \hat{s}_2)) = \hat{\pi}' = \hat{\pi}$, т. е. путь $\pi$ имеет прообраз. В силу произвольности выбора $\pi \in P \setminus \{\Lambda\}$ это означает сюръективность функции $\hat{\pi}$. ■

**Утверждение 2.** Функция $\hat{\pi} : S_{\text{pairs}} \to P \setminus \{\Lambda\}$ является инъекцией.

***Доказательство.*** Зафиксируем пары $(\hat{s}_1, \hat{s}_2), (\hat{s}'_1, \hat{s}'_2) \in S_{\text{pairs}}$, такие, что $\hat{\pi}(\hat{s}_1, \hat{s}_2) = \hat{\pi}(\hat{s}'_1, \hat{s}'_2) = \pi \in P \setminus \{\Lambda\}$. Пусть путь $\pi$ является последовательностью смежных рёбер $e_1, \ldots, e_n$. В соответствии с определением функции $\hat{\pi}$ выполнено $|\hat{s}_1| = |\hat{s}'_1| = n$, $|\hat{s}_2| = |\hat{s}'_2| = n - 1$.

Так как $\hat{\pi}((\hat{s}_1, \hat{s}_2)) = \pi$, ребру $e_1$ приписана метка $(\hat{s}_1[1], *)$. С другой стороны, так как $\hat{\pi}((\hat{s}'_1, \hat{s}'_2)) = \pi$, ребру $e_1$ приписана метка $(\hat{s}'_1[1], *)$. Таким образом, $\hat{s}_1[1] = \hat{s}'_1[1]$.

Пусть $k < n$. Так как $\hat{\pi}((\hat{s}_1, \hat{s}_2)) = \pi$, ребру $e_{k+1}$ приписана метка $(\hat{s}_1[k+1], \hat{s}_2[k])$. С другой стороны, так как $\hat{\pi}((\hat{s}'_1, \hat{s}'_2)) = \pi$, ребру $e_{k+1}$ приписана метка $(\hat{s}'_1[k+1], \hat{s}'_2[k])$. Таким образом, для всех $k < n$ выполнено $\hat{s}_1[k+1] = \hat{s}'_1[k+1]$, а также $\hat{s}_2[k] = \hat{s}'_2[k]$.

Следовательно, $(\hat{s}_1, \hat{s}_2) = (\hat{s}'_1, \hat{s}'_2)$, что и означает инъективность функции $\hat{\pi}$. ■

**Следствие 1.** Функция $\hat{\pi} : S_{\text{pairs}} \to P \setminus \{\Lambda\}$ является биекцией.

### 4.2. Вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$

**Утверждение 3.** Пусть $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$. Тогда вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ равен

$$|\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|.$$

***Доказательство.*** Пусть путь $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ является последовательностью смежных рёбер $e_1, \ldots, e_n$, где $|\hat{s}_1| = n$, $|\hat{s}_2| = n-1$. В соответствии с определением функции $\hat{\pi}$ ребро $e_1$ имеет буквенную метку $(\hat{s}_1[1], *)$, при всех $k < n$ ребро $e_{k+1}$ имеет буквенную метку $(\hat{s}_1[k+1], \hat{s}_2[k])$.

Ребро $e_1$ выходит из инициальной вершины, далее обозначаемой как $v_0$. Вершину, из которой выходит ребро $e_{k+1}$, обозначим как $v_k$. Таким образом, ребро $e_k$ идет из вершины $v_{k-1}$ в вершину $v_k$, и путь $\pi((\hat{s}_1, \hat{s}_2))$ последовательно проходит через вершины $v_0, v_1, \dots v_n$.

Обозначим префикс слова $\hat{s}_1$ длины $k$ через $\hat{s}_1^k$, т. е. $\hat{s}_1^k = \Lambda$ при $k = 0$, $\hat{s}_1^k = \hat{s}_1[1] \dots \hat{s}_1[k]$ при $k > 0$. По определению функции $\hat{F}$ выполнено

$$\hat{F}(\hat{s}_1) = F(\Lambda)F(\hat{s}_1[1])F(\hat{s}_1[1]\hat{s}_1[2]) \dots F(\hat{s}_1[1] \dots \hat{s}_1[n]) = F(\hat{s}_1^0) \dots F(\hat{s}_1^n)$$

и, следовательно, $|\hat{F}(\hat{s}_1)| = \sum\limits_{k=0}^{n} |F(\hat{s}_1^k)|$.

Обозначим префикс слова $\hat{s}_2$ длины $k$ как $\hat{s}_2^k$. Тогда

$$\hat{F}(\hat{s}_2) = F(\Lambda)F(\hat{s}_2[1])F(\hat{s}_2[1]\hat{s}_2[2]) \dots F(\hat{s}_2[1] \dots \hat{s}_2[n-1]) = F(\hat{s}_2^0) \dots F(\hat{s}_2^{n-1})$$

и, следовательно, $|\hat{F}(\hat{s}_2)| = \sum\limits_{k=0}^{n-1} |F(\hat{s}_2^k)|$.

Докажем по индукции, что для всех $k$, таких, что $1 \leqslant k \leqslant n$, выполнено $v_k = (q_{\hat{s}_1^k}, q_{\hat{s}_2^{k-1}}) \in Q^2$.

База индукции: $k = 1$. Ребро $e_1$ исходит из инициальной вершины и имеет буквенную метку $(\hat{s}_1[1], *)$. Следовательно, ребро $e_1$ входит в вершину $(\phi(q_\Lambda, \hat{s}_1[1]), q_\Lambda)$. Однако ребро $e_1$ входит в вершину $v_1$. Таким образом, $v_1 = (\phi(q_\Lambda, \hat{s}_1[1]), q_\Lambda) = (q_{\hat{s}_1[1]}, q_\Lambda) = (q_{\hat{s}_1^1}, q_{\hat{s}_2^0})$, где символ $\hat{s}_1[1]$ рассматривается как однобуквенное слово.

Предположим, что $v_k = (q_{\hat{s}_1^k}, q_{\hat{s}_2^{k-1}})$. Ребро $e_{k+1}$ исходит из вершины $v_k$ и имеет буквенную метку $(\hat{s}_1[k+1], \hat{s}_2[k])$. Следовательно, ребро $e_{k+1}$ входит в вершину $(\phi(q_{\hat{s}_1^k}, \hat{s}_1[k+1]), \phi(q_{\hat{s}_2^{k-1}}, \hat{s}_2[k])) = (q_{\hat{s}_1^k \hat{s}_1[k+1]}, q_{\hat{s}_2^{k-1}\hat{s}_2[k]}) = (q_{\hat{s}_1^{k+1}}, q_{\hat{s}_2^k})$. Однако ребро $e_{k+1}$ входит в вершину $v_{k+1}$. Таким образом, $v_{k+1} = (q_{\hat{s}_1^{k+1}}, q_{\hat{s}_2^k})$.

Ребро $e_1$ исходит из инициальной вершины и имеет буквенную метку $(\hat{s}_1[1], *)$. Следовательно, ребро $e_1$ имеет числовую метку $|\psi(q_\Lambda, \hat{s}_1[1])| = |F(\hat{s}_1[1])| = |F(\hat{s}_1^1)|$, где символ $\hat{s}_1[1]$ рассматривается как однобуквенное слово.

Зафиксируем $1 \leqslant k < n$. Ребро $e_{k+1}$ исходит из вершины $v_k = (q_{\hat{s}_1^k}, q_{\hat{s}_2^{k-1}})$ и имеет буквенную метку $(\hat{s}_1[k+1], \hat{s}_2[k])$. Следовательно, ребро $e_{k+1}$ имеет числовую метку $|\psi(q_{\hat{s}_1^k}, \hat{s}_1[k+1])| - |\psi(q_{\hat{s}_2^{k-1}}, \hat{s}_2[k])| = |F(\hat{s}_1^k \hat{s}_1[k+1])| - |F(\hat{s}_2^{k-1}\hat{s}_2[k])| = |F(\hat{s}_1^{k+1})| - |F(\hat{s}_2^k)|$.

Вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ равен сумме числовых меток рёбер $e_1, \dots, e_n$, т. е. $|F(\hat{s}_1^1)|$ в случае $n = 1$ и $|F(\hat{s}_1^1)| + \sum\limits_{k=2}^{n} (|F(\hat{s}_1^k)| - |F(\hat{s}_2^{k-1})|)$ в случае $n > 1$.

Преобразуем выражение для веса пути:

— в случае $n = 1$:

$$|F(\hat{s}_1^1)| = |F(\Lambda)| + |F(\hat{s}_1^1)| - |F(\Lambda)| = |F(\hat{s}_1^0)| + |F(\hat{s}_1^1)| - |F(\hat{s}_2^0)| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|;$$

— в случае $n > 1$:

$$|F(\hat{s}_1^1)| + \sum_{k=2}^{n} (|F(\hat{s}_1^k)| - |F(\hat{s}_2^{k-1})|) = \sum_{k=1}^{n} |F(\hat{s}_1^k)| - \sum_{k=1}^{n-1} |F(\hat{s}_2^k)| =$$

$$= (|F(\Lambda)| + \sum_{k=1}^{n} |F(\hat{s}_1^k)|) - (|F(\Lambda)| + \sum_{k=1}^{n-1} |F(\hat{s}_2^k)|) = \sum_{k=0}^{n} |F(\hat{s}_1^k)| - \sum_{k=0}^{n-1} |F(\hat{s}_2^k)| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|.$$

Таким образом, в обоих случаях вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ равен $|\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|$. ■

**Утверждение 4.** Пара $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$ является аномалией 2-го рода тогда и только тогда, когда вес пути $\pi((\hat{s}_1, \hat{s}_2))$ неположителен.

***Доказательство.***

Необходимость. Пусть $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$ — пара, являющаяся аномалией 2-го рода, тогда $|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|$. В соответствии с утверждением 3 вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ равен $|\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)| \leqslant 0$.

Достаточность. Пусть $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$ — такая пара, что вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2))$ неположителен. Следовательно, $|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|$ и пара $(\hat{s}_1, \hat{s}_2)$ является аномалией 2-го рода.

Утверждение 4 доказано. ■

***Доказательство теоремы 2*** (формулировка на с. 85).

$1 \Rightarrow 2$:

Пусть $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$ — пара, являющаяся аномалией 2-го рода, тогда в соответствии с утверждением 4 вес пути $\hat{\pi}((\hat{s}_1, \hat{s}_2)) \in P \setminus \{\Lambda\}$ неположителен.

$2 \Rightarrow 1$:

Пусть $\pi \in P \setminus \{\Lambda\}$ — путь неположительного веса. По утверждению 1 функция $\hat{\pi}$ является биекцией, поэтому существует такая пара $(\hat{s}_1, \hat{s}_2) \in S_{\text{pairs}}$, что $\hat{\pi}((\hat{s}_1, \hat{s}_2)) = \pi$. Тогда в соответствии с утверждением 4 пара $(\hat{s}_1, \hat{s}_2)$ является аномалией 2-го рода.

Теорема 2 доказана. ■

## 5. Доказательство теоремы 3

В п. 5.1 описаны вспомогательные свойства, которыми может обладать граф во время выполнения алгоритма. Соотношения между данными свойствами установлены в п. 5.2. В п. 5.3 и 5.4 представлены состояния графа после выполнения шага 2 и шага 5 соответственно. Описанию результата заключительной проверки посвящён п. 5.5.

### 5.1. Свойства графа

Введём некоторые вспомогательные свойства графа. Поскольку числовые метки вершин графа изменяются по ходу выполнения алгоритма, то все указанные свойства определены для некоторого момента времени.

**Определение 7.** Будем говорить, что вершина графа $v \in V$ является вершиной достижимой разметки, если выполнено одно из следующих условий:

1) $d(v) = +\infty$;
2) $d(v) \in \mathbb{Z}$ и существует путь $\pi \in P$, оканчивающийся в вершине $v$, такой, что $d(\pi) = d(v)$;
3) $d(v) = -\infty$ и для любого $M \in \mathbb{Z}$ существует путь $\pi \in P$, оканчивающийся в вершине $v$, такой, что $d(\pi) < M$.

**Определение 8.** Будем говорить, что граф является графом достижимой разметки, если все вершины графа являются вершинами достижимой разметки и для инициальной вершины $v_0$ выполнено $d(v_0) = 0$.

**Определение 9.** Будем говорить, что путь $\pi \in P$, закачивающийся в вершине $v$, обработан в графе, если $d(v) \leqslant d(\pi)$.

**Определение 10.** Будем говорить, что граф полностью обработан, если он является графом достижимой разметки и в нём обработаны все пути из множества $P$.

**Определение 11.** Будем говорить, что граф почти обработан, если он является графом достижимой разметки и в нём обработаны все пути из множества $P_0$.

**Определение 12.** Будем говорить, что граф является уменьшаемым, если в нём найдётся такое ребро $e : v \to w$, что $d(v) + d(e) < d(w)$.

Введём обозначения для конкатенации пути с ребром и для конкатенации двух путей. Пусть путь $\pi$ заканчивается в той же вершине, из которой исходит ребро $e$. Тогда путь, полученный добавлением ребра $e$ в конец пути $\pi$, будем обозначать $\pi + e$. Пусть теперь путь $\pi_1$ заканчивается в той же вершине, в которой начинается путь $\pi_2$. Тогда путь, полученный склеиванием конца пути $\pi_1$ с началом пути $\pi_2$, будем обозначать $\pi_1 + \pi_2$.

### 5.2. Соотношения свойств

**Утверждение 5.** Пусть граф не содержит цикла отрицательного веса, достижимого из инициальной вершины. Тогда для любой вершины $v \in V$ и для каждого пути $\pi \in P$, заканчивающегося в вершине $v$, найдётся путь $\pi_0 \in P_0$, также заканчивающийся в вершине $v$, для которого $d(\pi_0) \leqslant d(\pi)$.

***Доказательство.*** Индукция по длине пути $\pi$.

База индукции. Путь длины 0, т. е. пустой путь, уже лежит в множестве $P_0$.

Шаг индукции. Предположим, что утверждение выполнено для всех путей из множества $P$ длины меньше $k$. Докажем, что тогда оно верно для любого пути $\pi \in P$ длины $k$; пусть путь $\pi$ заканчивается в вершине $v$.

Если путь $\pi$ не содержит циклов, то $\pi \in P_0$ и можно положить $\pi_0 = \pi$.

Иначе все циклы, содержащиеся в пути $\pi$, имеют неотрицательный вес; посредством «вырезания» какого-нибудь цикла можно получить путь $\pi' \in P$, заканчивающийся в вершине $v$ и имеющий длину меньше $k$. Для пути $\pi'$ выполнено $d(\pi') \leqslant d(\pi)$ и по предположению индукции существует заканчивающийся в вершине $v$ путь $\pi_0 \in P_0$, для которого $d(\pi_0) \leqslant d(\pi')$. Следовательно, $d(\pi_0) \leqslant d(\pi)$. ■

**Утверждение 6.** Пусть $\pi \in P$ — путь, содержащий цикл отрицательного веса и заканчивающийся в вершине $v$. Тогда для любого $M \in \mathbb{Z} \cup \{+\infty\}$ существует путь $\pi'$, заканчивающийся в вершине $v$, для которого $d(\pi') < M$.

***Доказательство.*** Предположим обратное: существует $M \in \mathbb{Z} \cup \{+\infty\}$, такое, что для любого пути $\pi' \in P$, заканчивающегося в вершине $v$, выполнено $d(\pi') \geqslant M$.

Обозначим множество путей из $P$, содержащих цикл отрицательного веса и заканчивающихся в вершине $v$, как $P'$. По условию $\pi \in P'$, т. е. множество $P'$ непусто. Для любого $\pi' \in P'$ выполнено $d(\pi') \geqslant M$.

Так как множество $P'$ непусто и множество весов путей из $P'$ ограничено снизу, то можно выбрать путь $\pi' \in P'$, который имеет минимальный вес в множестве $P'$. Путь $\pi'$ содержит цикл отрицательного веса, т. е. он представим в виде $\pi' = \pi_1 + c + \pi_2$, где $\pi_1$ — путь из инициальной вершины в некоторую вершину $w$; $c$ — цикл отрицательного веса, начинающийся и заканчивающийся в вершине $w$; $\pi_2$ — путь из вершины $w$ в вершину $v$.

Рассмотрим путь $\pi'' = \pi_1 + c + c + \pi_2$. Он содержит цикл отрицательного веса и заканчивается в вершине $v$, т. е. $\pi'' \in P'$ и $d(\pi'') = d(\pi') + d(c) < d(\pi')$, что противоречит минимальности веса пути $\pi'$. ■

**Утверждение 7.** Если граф является графом достижимой разметки, то следующие условия равносильны:

1) граф является полностью обработанным;
2) граф не является уменьшаемым.

***Доказательство.***

$1 \Rightarrow 2$:

Предположим обратное. Тогда найдётся такое ребро $e : v \to w$, что выполнено $d(v) + d(e) < d(w)$. Формально возможны три случая: $d(v) = +\infty$, $d(v) \in \mathbb{Z}$, $d(v) = -\infty$.

Так как $d(v) + d(e) < d(w)$, то $d(v) \neq +\infty$.

Рассмотрим случай $d(v) \in \mathbb{Z}$, докажем, что он невозможен.

Так как вершина $v$ является вершиной достижимой разметки, то существует такой путь $\pi \in P$, что $d(\pi) = d(v)$ и $\pi$ заканчивается в вершине $v$. Рассмотрим путь $\pi' = \pi + e \in P$. Он заканчивается в вершине $w$ и $d(\pi') = d(\pi) + d(e) = d(v) + d(e) < d(w)$. Так как $d(\pi') < d(w)$, то путь $\pi' \in P$ не является обработанным, что противоречит тому, что граф является полностью обработанным.

Рассмотрим случай $d(v) = -\infty$, докажем, что он невозможен.

Выберем произвольное $M \in \mathbb{Z}$. Так как граф является графом достижимой разметки, то существует заканчивающийся в вершине $v$ путь $\pi \in P$, что $d(\pi) < M - d(e)$. Рассмотрим путь $\pi' = \pi + e \in P$. Он заканчивается в вершине $w$ и $d(\pi') = d(\pi) + d(e) < M - d(e) + d(e) = M$.

Таким образом, для любого $M \in \mathbb{Z}$ существует заканчивающийся в вершине $w$ путь $\pi' \in P$, для которого $d(\pi') < M$. Из того, что граф является графом достижимой разметки, следует, что $d(w) = -\infty$.

Получили противоречие: $-\infty = -\infty + d(e) = d(v) + d(e) < d(w) = -\infty$.

$2 \Rightarrow 1$:

База индукции: пустой путь является обработанным в графе.

Действительно, пустой путь $\Lambda \in P$ заканчивается в инициальной вершине $v_0$, для которой в силу того, что граф является графом достижимой разметки, выполнено $d(v_0) \leqslant 0$.

Шаг индукции. Предположим, что обработаны все пути длины $k$ из множества $P$. Докажем, что обработаны все пути длины $k+1$ из множества $P$.

Предположим обратное. Тогда существует путь $\pi \in P$ длины $k+1$, не являющийся обработанным в графе. Построим путь $\pi' \in P$ длины $k$ удалением последнего ребра $e : v \to w$ пути $\pi$. По построению $d(\pi') + d(e) = d(\pi)$. По предположению индукции, путь $\pi'$, заканчивающийся в вершине $v$, является обработанным. Следовательно, $d(v) \leqslant d(\pi')$.

Так как граф не является уменьшаемым, то $d(w) \leqslant d(v) + d(e) \leqslant d(\pi') + d(e) = d(\pi)$. Так как путь $\pi \in P$ заканчивается в вершине $w$ и не является обработанным, то $d(w) > d(\pi)$. Противоречие.

Утверждение 7 доказано. ■

**Утверждение 8.** Если граф почти обработан и не содержит вершин с меткой $-\infty$, то следующие условия равносильны:

1) граф содержит цикл отрицательного веса, достижимый из инициальной вершины;
2) граф является уменьшаемым.

***Доказательство.***

$1 \Rightarrow 2$:

Так как граф содержит цикл отрицательного веса, достижимый из инициальной вершины, то существует путь $\pi \in P$, содержащий цикл отрицательного веса. Вершину, в которой заканчивается путь $\pi$, обозначим через $v$.

Поскольку $d(v) \in \mathbb{Z} \cup \{+\infty\}$, по утверждению 6 существует путь $\pi' \in P$, заканчивающийся в вершине $v$, для которого $d(\pi') < d(v)$, следовательно, путь $\pi'$ не обработан в графе.

Граф почти обработан, поэтому он является графом достижимой разметки. Так как путь $\pi' \in P$ не обработан, то граф не является полностью обработанным. Следовательно, в соответствии с утверждением 7 граф является уменьшаемым.

$2 \Rightarrow 1$:

Предположим обратное: граф не содержит цикла отрицательного веса, достижимого из инициальной вершины.

Выберем путь $\pi \in P$, вершину, в которой он заканчивается, обозначим как $v$.

Так как граф не содержит цикла отрицательного веса, достижимого из инициальной вершины, то по утверждению 5 существует заканчивающийся в вершине $v$ ациклический путь $\pi_0 \in P_0$, для которого $d(\pi_0) \leqslant d(\pi)$.

Так как граф является почти обработанным, то путь $\pi_0$ обработан в графе, т. е. $d(v) \leqslant d(\pi_0)$. Следовательно, $d(v) \leqslant d(\pi_0) \leqslant d(\pi)$, т. е. путь $\pi$ также обработан в графе. Поскольку это верно для любого пути $\pi \in P$, граф является полностью обработанным, а значит, он является графом достижимой разметки. В соответствии с утверждением 7 граф не является уменьшаемым, что противоречит условию.

Утверждение 8 доказано. ■

### 5.3. Регулярные обходы

Установим, что после выполнения шага 2 алгоритма выполнены условия утверждения 8, т. е. граф является почти обработанным и не содержит вершин с меткой $-\infty$ (см. далее утверждение 12).

**Утверждение 9.** Процедура регулярной обработки сохраняет свойство графа «быть графом достижимой разметки», т. е. если граф перед регулярной обработкой ребра $e : v \to w$ был графом достижимой разметки, то и после выполнения процедуры он остаётся графом достижимой разметки.

***Доказательство.*** Процедура регулярной обработки ребра $e : v \to w$, может быть, меняет только метку вершины $w$. Так как в инициальную вершину рёбра не входят, то вершина $w$ не является инициальной. Таким образом, нужно проверить только сохранение свойства «быть графом достижимой разметки» в вершине $w$ и только в том случае, если процедура меняет метку вершины $w$, т. е. если до её исполнения $d(v) + d(e) < d(w)$.

Формально возможны три случая: $d(v) = +\infty$, $d(v) \in \mathbb{Z}$, $d(v) = -\infty$.

Так как $d(v) + d(e) < d(w)$, то $d(v) \neq +\infty$.

Рассмотрим случай $d(v) \in \mathbb{Z}$, докажем, что после обработки ребра $e$ вершина $w$ останется вершиной достижимой разметки.

Так как граф до обработки ребра $e$ является графом достижимой разметки, то существует заканчивающийся в вершине $v$ путь $\pi \in P$, для которого $d(\pi) = d(v)$. Процедура обработки не меняет метки вершины $v$, поэтому и после неё $d(\pi) = d(v)$.

Положим $\pi' = \pi + e \in P$. Путь $\pi'$ заканчивается в вершине $w$, по построению $d(\pi') = d(\pi) + d(e) = d(v) + d(e)$. После процедуры обработки $d(w) = d(v) + d(e) = d(\pi')$, следовательно, вершина $w$ останется вершиной достижимой разметки.

Рассмотрим случай $d(v) = -\infty$.

После обработки ребра $e$ выполнено $d(w) = -\infty$. Выберем произвольное $M \in \mathbb{Z}$.

Так как граф является графом достижимой разметки до обработки ребра $e$ и $d(v) = -\infty$, то существует заканчивающийся в вершине $v$ путь $\pi \in P$, такой, что $d(\pi) < M - d(e)$. Положим $\pi' = \pi + e \in P$. Путь $\pi'$ заканчивается в вершине $w$. По построению $d(\pi') = d(\pi) + d(e) < M$.

Таким образом, для любого $M \in \mathbb{Z}$ существует заканчивающийся в вершине $w$ путь $\pi' \in P$, для которого $d(\pi') < M$. Значит, после процедуры обработки вершина $w$ останется вершиной достижимой разметки. ■

**Утверждение 10.** Пусть $\pi \in P$ — путь, заканчивающийся в вершине $v$. Если путь $\pi$ обработан в графе перед исполнением процедуры регулярной обработки ребра $e : v \to w$, то после исполнения этой процедуры заканчивающийся в вершине $w$ путь $\pi' = \pi + e \in P$ будет обработан в графе.

***Доказательство.*** Так как путь $\pi \in P$ обработан в графе, то как перед, так и после обработки выполнено $d(v) \leqslant d(\pi)$.

Формально возможны два случая (до исполнения процедуры): $d(v) + d(e) < d(w)$ и $d(v) + d(e) \geqslant d(w)$.

Рассмотрим случай $d(v) + d(e) \geqslant d(w)$, докажем, что путь $\pi'$ обработан в графе после исполнения процедуры.

Так как $d(v)+d(e) \geqslant d(w)$, то процедура не меняет метку вершины $w$. Следовательно, после исполнения процедуры $d(w) \leqslant d(v) + d(e) \leqslant d(\pi) + d(e) = d(\pi')$, где $\pi' \in P$, т. е. путь $\pi'$ обработан в графе.

Рассмотрим случай $d(v) + d(e) < d(w)$. Так как $d(v) + d(e) < d(w)$, то после исполнения процедуры $d(w) = d(v) + d(e) \leqslant d(\pi) + d(e) = d(\pi')$, где $\pi' \in P$, т. е. путь $\pi'$ обработан в графе. ■

**Утверждение 11.** Если в графе обработаны все пути длины не более $k$ из множества $P$, то после выполнения процедуры регулярного обхода будут обработаны все пути длины не более $k + 1$ из множества $P$.

***Доказательство.*** Процедура регулярной обработки рёбер не увеличивает меток вершин, поэтому она сохраняет обработанность путей, т. е. если путь $\pi \in P$ является обработанным в графе до процедуры регулярной обработки, то он останется обработанным и после неё.

Процедура регулярного обхода — это совокупность вызовов процедуры регулярной обработки, следовательно, она также сохраняет обработанность путей и после её исполнения все пути длины $k$ будут обработаны в графе.

Выберем произвольный путь $\pi' \in P$ длины $k+1$, последнее ребро пути $\pi'$ обозначим как $e$. Таким образом, $\pi' = \pi + e$, где $\pi \in P$ — путь длины $k$.

По условию путь $\pi \in P$ обработан. Так как процедура регулярного обхода вызывает процедуру регулярной обработки ребра $e$, то по утверждению 10 после регулярного обхода путь $\pi' = \pi + e$ будет обработан. ■

**Утверждение 12.** После исполнения процедуры инициализации и $|V| - 1$ регулярных обходов граф является почти обработанным и не содержит вершин с меткой $-\infty$

***Доказательство.*** После процедуры инициализации граф является графом достижимой разметки, так как все вершины, кроме инициальной, являются вершинами достижимой разметки, поскольку имеют метку $+\infty$; инициальная вершина, имеющая метку 0, является вершиной достижимой разметки, поскольку в неё ведёт путь $\Lambda \in P$ нулевого веса и выполнено условие $d(v_0) = 0 \leqslant 0$.

После процедуры инициализации пустой путь $\Lambda \in P$ обработан в графе, т. е. обработаны все пути длины 0.

В соответствии с утверждением 11 после исполнения $|V| - 1$ регулярных обходов будут обработаны все пути из множества $P$ длины не более $|V| - 1$. Так как все пути из множества $P_0$ не содержат циклов, то их длина не превышает $|V| - 1$. Таким образом, после исполнения регулярных обходов будут обработаны все пути из множества $P_0$.

По утверждению 9 после регулярных обходов граф останется графом достижимой разметки; так как на этот момент в нём обработаны все пути из множества $P_0$, граф является почти обработанным.

Процедура инициализации не приписывает никакой вершине метку $-\infty$. Процедура регулярной обработки и, следовательно, процедура регулярного обхода также не приписывает метки $-\infty$. Следовательно, после исполнения регулярных обходов нет вершин с меткой $-\infty$. ■

### 5.4. Финальный обход

Установим свойства графа, которыми он обладает на момент завершения шага 5.

**Утверждение 13.** Пусть граф не содержит циклов отрицательного веса, достижимых из инициальной вершины. Тогда на момент завершения шага 5 алгоритма выполнены следующие условия:

1) граф является полностью обработанным;
2) граф не содержит вершин с числовой меткой $-\infty$.

***Доказательство.*** В соответствии с утверждением 12, на момент завершения шага 2 граф почти обработан и не содержит вершин с меткой $-\infty$. Тогда по утверждению 8 граф не является уменьшаемым.

Так как граф почти обработан, он является графом достижимой разметки. Так как граф не является уменьшаемым, то в соответствии с утверждением 7 он полностью обработан на момент завершения шага 2.

Так как граф не является уменьшаемым, для всякого ребра $e \in E$, $e : v \to w$ выполнено $d(v) + d(e) \geqslant d(w)$ и, следовательно, процедура финальной обработки (а значит, и процедура финального обхода) не производит никаких действий на ребре $e$.

Таким образом, на момент завершения шага 5 граф является таким же, как и после шага 2, т. е. полностью обработанным и не содержащим вершин с меткой $-\infty$. ■

**Утверждение 14.** Пусть граф содержит цикл отрицательного веса, достижимый из инициальной вершины. Тогда на момент завершения шага 5 существует вершина с числовой меткой $-\infty$.

***Доказательство.*** По утверждению 12 на момент завершения шага 2 граф является почти обработанным и не содержит вершин с числовой меткой $-\infty$. Так как граф содержит цикл отрицательного веса, достижимый из инициальной вершины, то в соответствии с утверждением 8 граф является уменьшаемым.

Предположим, что на момент завершения шага 5 граф не содержит вершин с числовой меткой $-\infty$.

Как на момент завершения шага 2, так и на момент завершения шага 5 граф не содержит вершин с числовой меткой $-\infty$. Следовательно, каждый вызов процедуры финальной обработки не производит никаких действий. Значит, для каждого ребра $e \in E$, $e : v \to w$, выполнено $d(v) + d(e) \geqslant d(w)$, т. е. граф не является уменьшаемым на момент завершения шага 2 — противоречие. ■

### 5.5. Заключительная проверка

Опишем результат работы шага 8.

**Утверждение 15.** Следующие условия равносильны:

1) на момент завершения шага 5 существует неинициальная вершина с неположительной числовой меткой;
2) существует непустой путь $\pi \in P \setminus \{\Lambda\}$ неположительного веса.

***Доказательство.*** Возможны два случая: граф содержит или не содержит цикл отрицательного веса, достижимый из инициальной вершины.

С л у ч а й 1: граф содержит цикл отрицательного веса, достижимый из инициальной вершины.

$1 \Rightarrow 2$:

По условию существует путь $\pi \in P$, содержащий отрицательный цикл. Вершину, в которой заканчивается путь $\pi$, обозначим через $v$. Путь $\pi$ не пустой и, значит, вершина $v$ не инициальная.

В соответствии с утверждением 6 существует путь $\pi' \in P$, также заканчивающийся в вершине $v$, для которого $d(\pi') < 0$, т.е. путь $\pi'$ имеет неположительный вес; а так как путь $\pi'$ заканчивается в неинициальной вершине $v$, то он не является пустым, т. е. $\pi' \in P \setminus \{\Lambda\}$.

$2 \Rightarrow 1$:

Так как граф содержит цикл отрицательного веса, достижимый из инициальной вершины, то в соответствии с утверждением 14 на момент завершения шага 5 существует вершина $v$ с неположительной числовой меткой $-\infty$.

Вершина $v$ не является инициальной, потому что на момент завершения шага 5 инициальная вершина имеет метку 0: процедура инициализации назначает ей метку 0, а процедуры регулярной и финальной обработки не изменяют метку, поскольку в инициальную вершину не входят рёбра.

С л у ч а й 2: граф не содержит цикла отрицательного веса, достижимого из инициальной вершины.

В соответствии с утверждением 13 на момент завершения шага 5 граф является полностью обработанным (а значит, является графом достижимой разметки) и не содержит вершин с числовой меткой $-\infty$.

$1 \Rightarrow 2$:

Пусть $v$ — неинициальная вершина, для которой $d(v) \leqslant 0$. По условию $d(v) \neq -\infty$.

Так как $d(v) \in \mathbb{Z}$ и граф является графом достижимой разметки, то существует путь $\pi \in P$, заканчивающийся в вершине $v$, для которого $d(\pi) = d(v) \leqslant 0$, а так как вершина $v$ не инициальная, то путь $\pi \in P$ не пустой.

$2 \Rightarrow 1$:

Пусть $\pi \in P \setminus \{\Lambda\}$ — путь, для которого $d(\pi) \leqslant 0$; вершину, в которой заканчивается путь $\pi$, обозначим через $v$; очевидно, что она не является инициальной.

Так как граф является полностью обработанным, то в нём обработан путь $\pi \in P$, т. е. $d(v) \leqslant d(\pi) \leqslant 0$.

Утверждение 15 доказано. ■

***Доказательство теоремы 3*** (формулировка на с. 86).

В соответствии с описанием шага 8 алгоритм Беллмана — Форда выдаёт ответ «да» тогда и только тогда, когда к моменту завершения шага 5 не существует неинициальной вершины с неположительной числовой меткой.

По утверждению 15 к моменту завершения шага 5 существование неинициальной вершины с неположительной числовой меткой равносильно существованию непустого пути $\pi \in P \setminus \{\Lambda\}$ неположительного веса. ■

## Заключение

В работе представлен алгоритм проверки отсутствия лежащих в $S_{\text{pairs}}$ аномалий 2-го рода, т. е. решена частная задача, относящаяся к каналу частичного стирания.

Такие поставленные в работах [2, 4, 5] вопросы, как способы построения правильных функций поведения передатчика, построение поведения приемника по заданному поведению передатчика, а также оценки пропускной способности канала остаются за пределами настоящей работы и работы [1], поскольку они посвящены исключительно задаче проверки передатчика на предмет существования согласованного с ним приемника, которую требуется решить прежде иных упомянутых задач. Таким образом, указанные задачи являются предметом дальнейшего исследования.

## ЛИТЕРАТУРА

1. *Казаков И. Б.* Алгоритм валидации ограниченно-детерминированного поведения передатчика в канале частичного стирания // Прикладная дискретная математика. 2023. № 59. С. 88–110.
2. *Казаков И. Б.* Критерий существования корректного протокола в канале частичного стирания // Чебышевский сборник. 2021. Т. 22. № 1. С. 133–151.
3. *Казаков И. Б.* Критерий надёжности канала с запрещениями // Интеллектуальные системы. Теория и приложения. 2019. Т. 23. № 2. С. 33–55.
4. *Казаков И. Б.* Передача информации в каналах, задаваемых структурами частичного стирания. Ч. 1 // Программная инженерия. 2020. Т. 11. № 5. С. 277–284.
5. *Казаков И. Б.* Передача информации в каналах, задаваемых структурами частичного стирания. Ч. 2 // Программная инженерия. 2020. Т. 11. № 6. С. 322–329.
6. *Bellman R. E.* On a routing problem // Quarterly Appl. Math. 1958. V. 16. P. 87–90.
7. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algoritms. 2nd ed. Cambridge, Massachusetts: MIT Press and McGraw-Hill, 2001. 1203 p.

## REFERENCES

1. *Kazakov I. B.* Algoritm validatsii ogranichenno-determinirovannogo povedeniya peredatchika v kanale chastichnogo stiraniya [A validation algorithm of the transmitter‘s boundedly deterministic behaviour in a partial erasure channel]. Prikladnaya Diskretnaya Matematika, 2023, no. 59, pp. 88–110. (in Russian)
2. *Kazakov I. B.* Kriteriy sushchestvovaniya korrektnogo protokola v kanale chastichnogo stiraniya [Criterion for the existence of a consistent protocol in a partial erasure channel]. Chebyshevskiy Sbornik, 2021, vol. 22, no. 1, pp.. 133–151. (in Russian)
3. *Kazakov I. B.* Kriteriy nadezhnosti kanala s zapreshcheniyami [Reliabilty criterion for channels with prohibitions]. Intellektual’nye Sistemy. Teoriya i Prilozheniya, 2019, vol. 22, no. 2, pp. 33–55. (in Russian)
4. *Kazakov I. B.* Peredacha informatsii v kanalakh, zadavaemykh strukturami chastichnogo stiraniya. Ch. 1 [Transmission of information in channels specified by structures of partial erasure. P. 1]. Programmnaya Inzheneriya, 2020, vol. 11, no. 5, pp. 277–284. (in Russian)
5. *Kazakov I. B.* Peredacha informatsii v kanalakh, zadavaemykh strukturami chastichnogo stiraniya. Ch. 2 [Transmission of information in channels specified by structures of partial erasure. P. 2]. Programmnaya Inzheneriya, 2020, vol. 11, no. 6, pp. 322–329. (in Russian)
6. *Bellman R. E.* On a routing problem. Quarterly Appl. Math., 1958, vol. 16, pp. 87–90.
7. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algoritms. 2nd ed. Cambridge, Massachusetts, MIT Press and McGraw-Hill, 2001. 1203 p.

# ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.17 DOI 10.17223/20710410/67/5

## К ВОПРОСУ О МАКСИМАЛЬНОМ ЧИСЛЕ ВЕРШИН В ПРИМИТИВНЫХ РЕГУЛЯРНЫХ ГРАФАХ С ЭКСПОНЕНТОМ 3

И. В. Лось, М. Б. Абросимов

*Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, г. Саратов, Россия*

**E-mail:** los.ilia.ru@gmail.com, mic@rambler.ru

Рассматривается вопрос о максимальном числе вершин в примитивных неориентированных регулярных графах с экспонентом, равным 3. Получена оценка сверху этого числа в зависимости от порядка графа $p$: $n_p \leqslant p^3 - p^2 - 3p + 5$. Найдено точное значение максимального числа вершин в примитивных кубических графах с экспонентом, равным 3: $n_3 = 12$. Проведён вычислительный эксперимент и найдено число примитивных регулярных графов порядка $p \leqslant 9$ с числом вершин $n \leqslant 16$ и экспонентом, равным 3, для всех пар $(n, p)$.

**Ключевые слова:** *примитивный граф, регулярный граф, максимальное число вершин.*

## ABOUT THE MAXIMUM NUMBER OF VERTICES IN PRIMITIVE REGULAR GRAPHS WITH EXPONENT EQUALS 3

I. V. Los, M. B. Abrosimov

*Saratov State University, Saratov, Russia*

Some results on the maximum number of vertices in primitive regular graphs with exponent 3 are presented. We have found upper bound of this number depending on the degree $p$: $n_p \leqslant p^3 - p^2 - 3p + 5$. Also, the exact value of the maximum number of vertices in primitive cubic graphs with exponent 3 is given: $n_3 = 12$. A computation experiment has been conducted, and we have found the number of primitive regular graphs with degree $p \leqslant 9$, number of vertices $n \leqslant 16$ and exponent 3 for each $(n, p)$ pair.

**Keywords:** *primitive graph, regular graph, the maximum number of vertices.*

### Введение

Неотрицательная квадратная матрица $A$ называется *примитивной*, если существует натуральное $k$, такое, что $A^k$ положительна. Минимальное такое значение $k$ называется *экспонентом* матрицы $A$ [1]. Понятие примитивности легко переносится на графы. В данной работе рассматриваются простые неориентированные графы. Напомним некоторые определения.

*Регулярным* или *однородным* графом *порядка* $p$ называется граф, все вершины которого имеют степень $p$. Регулярные графы порядка 3 называются *кубическими*. *Диаметром* $\mathrm{d}(G)$ связного графа $G$ называется расстояние между наиболее удалёнными друг от друга вершинами этого графа. Связный граф $G$ называется *примитивным*, если существует $k \in \mathbb{N}$, такое, что между любыми двумя вершинами графа (в том числе из вершины в саму себя) существует маршрут длины $k$. Минимальное такое число $k$ называется *экспонентом* графа и обозначается $\exp(G)$.

Примитивные графы представляют теоретический и практический интерес [2–5]. Ряд работ посвящены исследованию примитивных регулярных графов [6–10]. Один из вопросов — при каком числе вершин могут существовать примитивные регулярные графы с заданным экспонентом. В данной работе представлены результаты для регулярных примитивных графов с экспонентом 3. В п. 1 приведены теоретические результаты. В п. 2 описан вычислительный эксперимент, который позволяет оценить эффективность полученных теоретических результатов. Ранее аналогичное исследование проводилось для регулярных примитивных графов с экспонентом 2 [6, 7]. В частности, в работе [6] описан вычислительный эксперимент по поиску всех регулярных примитивных графов с экспонентом 2 с числом вершин до 16.

## 1. Теоретические результаты

Очевидно, что у примитивного графа с экспонентом 3 диаметр может быть 2 или 3. Для упрощения дальнейших рассуждений доказано вспомогательное утверждение — необходимое условие примитивности графа с экспонентом 3.

**Утверждение 1.** В примитивном графе с экспонентом, равным 3, каждая вершина лежит хотя бы на одном цикле длины 3.

***Доказательство.*** По определению в примитивном графе с экспонентом 3 должен, в том числе, существовать маршрут длины 3 из любой вершины в саму себя, то есть каждая вершина должна лежать хотя бы на одном цикле длины 3. ■

Условие утверждения 1 не является достаточным. Например, на рис. 1 приведён пример графа, каждая вершина которого лежит на цикле длины 3, однако его экспонент равен 2. Данный контрпример является минимальным по числу вершин.

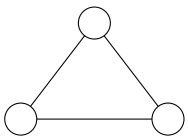

Рис. 1. Контрпример с экспонентом 2 для утверждения 1

Таким образом, в условие требуется включить ограничение на диаметр графа. Рассмотрим несколько достаточных условий, которые не являются необходимыми.

**Утверждение 2.** Если в графе $G$ с диаметром $\mathrm{d}(G) \leqslant 3$ каждое ребро входит в состав некоторого цикла длины 3, то граф $G$ является примитивным с экспонентом $\exp(G) \leqslant 3$, причём если $\mathrm{d}(G) = 3$, то и $\exp(G) = 3$.

***Доказательство.*** Для графов с диаметром $\mathrm{d}(G) \leqslant 2$ в [6] доказано, что граф $G$ является примитивным с экспонентом $\exp(G) = 2$. Таким образом, требуется доказать верность утверждения для графов с $\mathrm{d}(G) = 3$ и $\exp(G) = 3$. Рассмотрим пару произвольных вершин $x$ и $y$. В силу того, что $\mathrm{d}(G) = 3$, возможны три случая:

1) Длина кратчайшего пути между $x$ и $y$ равна 1, то есть вершины являются смежными. Тогда между ними существует маршрут длины 3: $x - y - x - y$.
2) Длина кратчайшего пути между $x$ и $y$ равна 2, и этот путь проходит через промежуточную вершину $t$. По условию любое из рёбер $x - t$ и $t - y$ лежит на цикле длины 3. Пусть ребро $x - t$ лежит на цикле $x - t - w$ (рис. 2). Тогда между $x$ и $y$ существует маршрут $x - w - t - y$ длины 3.

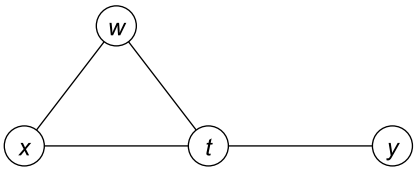


Рис. 2. Длина кратчайшего пути равна 2

3) Длина кратчайшего пути между $x$ и $y$ равна 3. Тогда маршрут длины 3 между этими вершинами совпадает с кратчайшим путём.

Утверждение 2 доказано. ■

На рис. 3 представлен 8-вершинный граф $G_1$ — минимальный по числу вершин регулярный граф, который показывает, что условие утверждения 2 не является необходимым: ребро между вершинами 3 и 5 (аналогично для ребра между вершинами 4 и 6) не лежит ни на одном цикле длины 3.

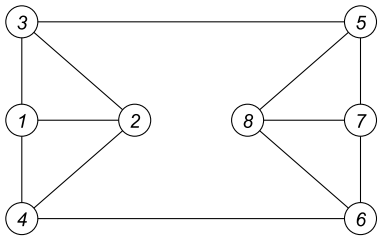


Рис. 3. Граф $G_1$

Интересен вопрос о том, насколько эффективно условие утверждения 2. С помощью вычислительного эксперимента, который описан в п. 2, получены результаты, приведённые в табл. 1, — количество контрпримеров для различных $n$ и $p$. Можно сделать вывод, что большая доля примитивных регулярных графов с экспонентом 3 не удовлетворяет предложенному условию. Поэтому усилим его; доказательство аналогично доказательству утверждения 2.

**Утверждение 3.** Если в графе $G$ с диаметром $\mathrm{d}(G) \leqslant 3$ из каждой пары смежных рёбер хотя бы одно входит в состав некоторого цикла длины 3, то граф $G$ является примитивным с экспонентом $\exp(G) \leqslant 3$, причём если $\mathrm{d}(G) = 3$, то и $\exp(G) = 3$.

Условие утверждения 3 также не является необходимым; на рис. 4 представлен 10-вершинный граф $G_2$ — минимальный по числу вершин регулярный граф, для которого оно не выполняется: рёбра $\{5, 6\}$ и $\{5, 8\}$ смежны и не лежат ни на одном цикле длины 3.

Данные о выполнении условия утверждения 3 приведены в табл. 2. Видно, что число контрпримеров уменьшилось в несколько раз, однако всё ещё велико.

Таблица 1

**Число графов с экспонентом 3, для которых условие, обратное утверждению 2, не выполняется**

| $n$ | $p$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 0 | – | – | – | – | – | – |
| 5 | – | 0 | – | – | – | – | – |
| 6 | 0 | 0 | 0 | – | – | – | – |
| 7 | – | 0 | – | 0 | – | – | – |
| 8 | 1 | 0 | 0 | 0 | 0 | – | – |
| 9 | – | 0 | – | 0 | – | 0 | – |
| 10 | 1 | 24 | 0 | 0 | 0 | 0 | 0 |
| 11 | – | 123 | – | 0 | – | 0 | – |
| 12 | 1 | 553 | 7 506 | 2 235 | 0 | 0 | 0 |
| 13 | – | 2 395 | – | 0 | – | 0 | – |
| 14 | 0 | 10 368 | 2 858 557 | 2 401 761 | 0 | 0 | 0 |

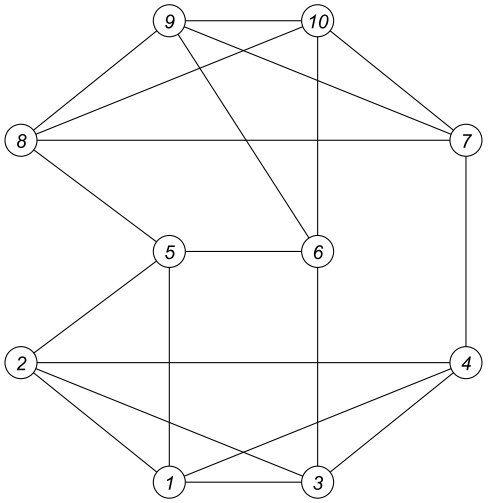


Рис. 4. Граф $G_2$

Обозначим через $n_p$ максимально возможное число вершин в регулярном примитивном графе с порядком $p$ и экспонентом, равным 3.

**Теорема 1.** Для максимально возможного числа вершин в регулярном примитивном графе с экспонентом 3 и порядком $p$ имеет место неравенство

$$n_p \leqslant p^3 - p^2 - 3p + 5.$$

***Доказательство.*** Диаметр графа с экспонентом, меньшим или равным 3, не больше 3. Оценим число вершин в $p$-регулярных графах с диаметром 3. В [11] приводится оценка максимального числа вершин в $p$-регулярных графах с диаметром $d$:

$$|V| \leqslant 1 + p\sum_{i=0}^{d-1}(p-1)^i.$$

Граф, который имеет число вершин, равное верхней границе, называется графом Мура. В нашем случае $d \leqslant 3$. Если мы выберем произвольную вершину $x$, то на расстоянии 1 от неё будет находиться $p$ вершин; на расстоянии 2 — максимум $p(p-1)$ вершин;

Таблица 2

**Число графов с экспонентом 3, для которых условие, обратное утверждению 3, не выполняется**

| $n$ | $p$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 0 | – | – | – | – | – | – |
| 5 | – | 0 | – | – | – | – | – |
| 6 | 0 | 0 | 0 | – | – | – | – |
| 7 | – | 0 | – | 0 | – | – | – |
| 8 | 0 | 0 | 0 | 0 | 0 | – | – |
| 9 | – | 0 | – | 0 | – | 0 | – |
| 10 | 0 | 18 | 0 | 0 | 0 | 0 | 0 |
| 11 | – | 109 | – | 0 | – | 0 | – |
| 12 | 0 | 524 | 1 463 | 0 | 0 | 0 | 0 |
| 13 | – | 2 345 | – | 0 | – | 0 | – |
| 14 | 0 | 10 290 | 2 634 777 | 946 878 | 0 | 0 | 0 |

на расстоянии 3 — максимум $p(p-1)^2$ вершин. Для $d=3$ оценка принимает вид

$$|V| \leqslant 1 + p + p(p-1) + p(p-1)^2. \tag{1}$$

После упрощения можно получить оценку $|V| \leqslant p^3 - p^2 + p + 1$, которая для $p = 3, 4, 5$ приведёт к значениям 22, 53 и 106 соответственно.

Вернёмся к неравенству (1) и улучшим оценку. Обозначим через $N(x)$ множество всех смежных с $x$ вершин. Тогда неравенство (1) для $d=3$ можно расписать следующим образом:

$$|V| \leqslant |\{x\}| + |N(x)| + \sum_{y \in N(x)} \Big( |N(y) \setminus \{x\}| + \sum_{y' \in N(y) \setminus \{x\}} |N(y') \setminus \{y\}| \Big).$$

Воспользуемся утверждением 1 и используем факт, что в графах с экспонентом, равным 3, каждая вершина лежит хотя бы на одном цикле длины 3. В этом случае для вершины $x$ существует ребро между хотя бы двумя смежными с $x$ вершинами, пусть это вершины $u$ и $v$. Таким образом, вершины $x$, $u$ и $v$ лежат хотя бы на одном цикле длины 3. Тогда оценка будет выглядеть следующим образом:

$$\begin{aligned} |V| \leqslant |\{x\}| + |N(x)| + |N(u) \setminus \{x\} \setminus \{v\}| + |N(v) \setminus \{x\} \setminus \{u\}| + \\ + \sum_{y' \in N(u) \setminus \{x\} \setminus \{v\}} |N(y') \setminus \{u\}| + \sum_{y' \in N(v) \setminus \{x\} \setminus \{u\}} |N(y') \setminus \{v\}| + \\ + \sum_{y \in N(x) \setminus \{u\} \setminus \{v\}} \left( |N(y) \setminus \{x\}| + \sum_{y' \in N(y) \setminus \{x\}} |N(y') \setminus \{y\}| \right). \end{aligned}$$

Расширим это условие на все остальные вершины множества $N(x)$. Если мы хотим максимизировать число вершин в графе, то необходимо иметь ребро между хотя бы двумя вершинами, соседними с данной. Пусть для вершины $y \in N(x)$ это вершины $l_y$ и $t_y$. Новая оценка выглядит следующим образом:

$$\begin{aligned} |V| \leqslant |\{x\}| + |N(x)| + |N(u) \setminus \{x\} \setminus \{v\}| + |N(v) \setminus \{x\} \setminus \{u\}| + \\ + \sum_{y' \in N(u) \setminus \{x\} \setminus \{v\}} |N(y') \setminus \{u\}| + \sum_{y' \in N(v) \setminus \{x\} \setminus \{u\}} |N(y') \setminus \{v\}| + \end{aligned}$$

$$+\sum_{y\in N(x)\setminus\{u\}\setminus\{v\}}\Bigg(|N(y)\setminus\{x\}|+|N(t_y)\setminus\{y\}\setminus\{l_y\}|+|N(l_y)\setminus\{y\}\setminus\{t_y\}|+$$
$$+\sum_{y'\in N(y)\setminus\{x\}\setminus\{l_y\}\setminus\{t_y\}}|N(y')\setminus\{y\}|\Bigg).$$

Так как граф $p$-регулярный, для каждой вершины $x$ выполняется $|N(x)|=p$. Поэтому верхняя оценка принимает следующий вид:

$$|V|\leqslant 1+p+(p-2)+(p-2)+(p-2)(p-1)+$$
$$+(p-2)\big((p-1)+(p-2)+(p-2)+(p-3)(p-1)\big)=$$
$$=3(p-1)+2(p-2)(p-1)+(p-2)^2(p+1)=p^3-p^2-3p+5.$$

Теорема 1 доказана. ■

В частности, имеем $n_3\leqslant 14$, $n_4\leqslant 41$ и $n_5\leqslant 90$. Эти значения существенно лучше тех, которые можно получить из формулы Мура. Более того, удалось получить точное значение для $n_3$.

**Теорема 2.** Не существует кубических примитивных графов с экспонентом 3 и числом вершин больше 12; $n_3=12$.

***Доказательство.*** На рис. 5 приведено изображение 12-вершинного кубического графа с экспонентом 3. Поэтому $n_3\geqslant 12$.

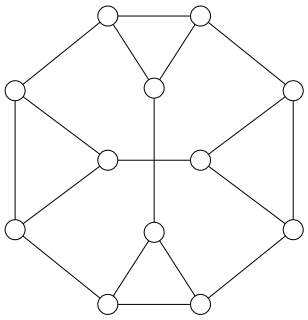

Рис. 5. 12-Вершинный кубический граф с экспонентом 3

Так как 3-регулярные графы не могут иметь нечётное число вершин, для доказательства теоремы нужно показать, что не существует кубического примитивного графа с экспонентом 3 и числом вершин 14. Рассмотрим все возможные графы и покажем, что ни один из них не удовлетворяет нужному условию.

На рис. 6 представлен вид 3-регулярного графа с экспонентом 3 и 14 вершинами. Каждая из вершин $1,\dots,6$ имеет свободную степень 2, остальные вершины свободных степеней не имеют. Поэтому вершины $1,\dots,6$ можно соединять рёбрами только между собой. Теперь посмотрим на вершину $v$. Расстояние от неё до любой другой вершины графа должно быть меньше или равно 3. Для вершин $1,2,3,4$ это можно сделать только одним способом: соединить каждую из них хотя бы с одной из вершин $5,6$. Аналогично рассмотрим вершину $u$: каждая из вершин $5,6$ должна быть соединена хотя бы с одной из вершин $3,4$. И, наконец, рассмотрим вершину $x$: каждая из вершин $5,6$ должна быть соединена хотя бы с одной из вершин $1,2$. Это означает, что вершина 5 должна быть соединена с вершинами 1 или 2 и 3 или 4. Аналогично для вершины 6. С другой стороны, каждая из вершин $1,2,3,4$ должна быть соединена

с вершинами 5 или 6. Единственный с точностью до изоморфизма способ это сделать представлен на рис. 7.

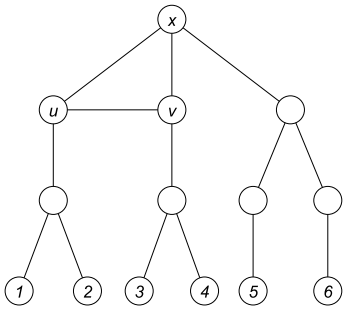


Рис. 6. Неполный вид 3-регулярного графа с экспонентом 3 и числом вершин 14

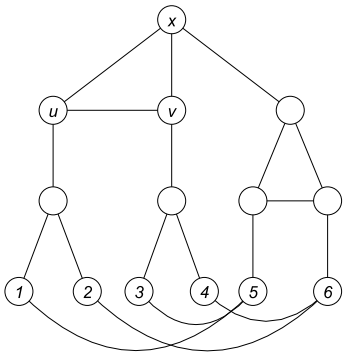


Рис. 7. Дополненный вид 3-регулярного графа с экспонентом 3 и числом вершин 14

Воспользуемся тем, что каждая из вершин 5 и 6 должна лежать хотя бы на одном цикле длины 3. Единственный способ добиться этого — соединить между собой пары вершин $1, 3$ и $2, 4$. Итоговый вид графа без свободных степеней представлен на рис. 8.

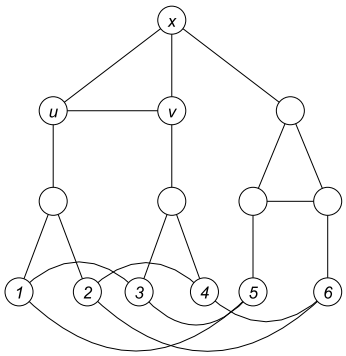


Рис. 8. Полный вид 3-регулярного графа с экспонентом 3 и числом вершин 14

Покажем, что экспонент этого графа не равен 3. Рассмотрим вершины $u$ и 1: между ними не существует маршрута длины 3. Следовательно, экспонент данного графа не равен 3, и мы доказали, что не существует 3-регулярных графов с экспонентом 3 и числом вершин 14. ■

Приведём результаты, касающиеся условий равенства экспонента трём для различных графов.

**Теорема 3.** В графе с диаметром 2, в котором каждая вершина лежит хотя бы на одном цикле длины 3, между любой парой вершин (в том числе из самой вершины в себя) существует маршрут длины 3.

***Доказательство.*** Существование маршрута длины 3 из любой вершины в саму себя следует из условия теоремы.

Рассмотрим две произвольные различные вершины $x$ и $y$. Так как диаметр графа равен 2, между ними должен существовать путь длины 1 и/или длины 2. Пусть между ними существует путь длины 1, т. е. они соединены ребром напрямую. Тогда между ними существует маршрут $x - y - x - y$ длины 3.

Пусть теперь между $x$ и $y$ не существует пути длины 1, тогда между ними существует путь длины 2 через некоторую промежуточную вершину $t$. Рассмотрим вершину $x$. По условию теоремы она лежит на некотором цикле длины 3, обозначим его $x - a - b$. Возможны два варианта:

1) Цикл $x - a - b$ не содержит вершину $t$ (то есть $t$ не совпадает ни с вершиной $a$, ни с вершиной $b$).

Рассмотрим пару вершин $a$ и $y$: так как диаметр графа равен 2, между ними должен существовать путь длины 1 или путь длины 2 через промежуточную вершину $c$. Пусть между ними существует путь длины 1, как показано на рис. 9. Тогда между $x$ и $y$ существует маршрут $x - b - a - y$ длины 3.

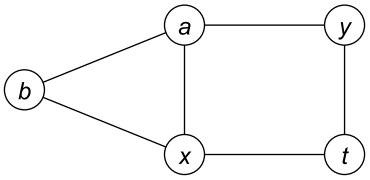


Рис. 9. Случай существования пути длины 1

Пусть теперь между $a$ и $y$ существует путь длины 2 через промежуточную вершину $c$, как показано на рис. 10. Тогда между $x$ и $y$ существует маршрут $x - a - c - y$ длины 3.

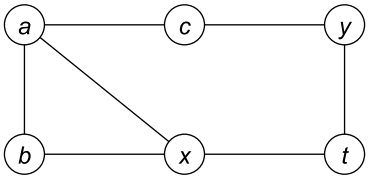


Рис. 10. Случай существования пути длины 2 через промежуточную вершину $c$

2) Пусть в цикле $x - a - b$ вершина $b$ совпадает с вершиной $t$. Тогда имеем цикл $x - t - a$ и между $x$ и $y$ существует маршрут $x - a - t - y$ длины 3, как показано на рис. 11. Случай $t = a$ является полностью симметричным.

Других случаев нет, так как вершина $y$ не может принадлежать циклу (тогда между $x$ и $y$ будет существовать путь длины 1). В силу произвольности выбора вершин $x$ и $y$ теорема доказана. ■

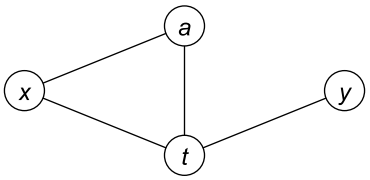


Рис. 11. Случай совпадения вершины $t$ с одной из вершин цикла

**Следствие 1.** Граф с диаметром 2, в котором каждая вершина лежит хотя бы на одном цикле длины 3, является примитивным, и его экспонент меньше или равен 3.

***Доказательство.*** Согласно теореме 3, в таком графе между любой парой вершин (в том числе из вершины в саму себя) существует маршрут длины 3. По определению данный граф примитивный и его экспонент не больше 3. ■

**Теорема 4.** Граф с диаметром 2 является примитивным и имеет экспонент, равный 3, тогда и только тогда, когда каждая его вершина лежит хотя бы на одном цикле длины 3 и существует хотя бы одно ребро, не лежащее ни на одном цикле длины 3.

***Доказательство.***

Необходимость. Если граф является примитивным и имеет экспонент 3, то, согласно утверждению 1, каждая его вершина лежит хотя бы на одном цикле длины 3. Кроме того, если граф имеет диаметр 2 и каждое его ребро лежит хотя бы на одном цикле длины 3, то, согласно [6], он является примитивным с экспонентом, равным 2. Значит, должно существовать хотя бы одно ребро, не лежащее ни на одном цикле длины 3.

Достаточность. Согласно следствию 1 из теоремы 3, если в графе с диаметром 2 каждая вершина лежит на хотя бы одном цикле длины 3, то граф является примитивным и его экспонент меньше или равен 3. Так как в примитивном графе с диаметром 2 экспонент больше или равен 2, для экспонента остаются только варианты 2 или 3. Однако, согласно [6], экспонент этого графа не может быть равен 2, поскольку существует хотя бы одно ребро, не лежащее на цикле длины 3. Таким образом, граф является примитивным и его экспонент равен 3. ■

## 2. Вычислительный эксперимент

Проведён вычислительный эксперимент с использованием кластера высокопроизводительных вычислений ПРЦ НИТ СГУ по подсчёту регулярных графов с экспонентом, равным 3, в рамках которого построена таблица числа примитивных регулярных графов со степенью $p \leqslant 9$, числом вершин $n \leqslant 16$ и экспонентом 3. Для этого написана программа на языке C++; генерация всех связных регулярных графов степени $p$ с фиксированным числом вершин $n$ производилась с помощью генератора графов `genreg` [12]. Для каждого из сгенерированных графов проверялось равенство экспонента трём. Дополнительно подсчитывалось количество контрпримеров для оценки эффективности достаточных условий, сформулированных в утверждениях 2 и 3 (см. табл. 1 и 2). Основная цель эксперимента состояла в том, чтобы посмотреть распределение регулярных графов с экспонентом 3 в зависимости от степеней вершин, а также как эти значения согласуются с теоретическими. В частности, экспериментальные данные о максимально возможном числе вершин для кубических графов подтверждают полученный теоретический результат: $n_3 \leqslant 14$.

Для проверки использовались возведение матрицы смежности графа в степень 3 и проверка полученной матрицы на отсутствие нулей. Важно отметить, что при те-

кущих ограничениях на $n \leqslant 16$ строки матрицы можно хранить в виде двоичных масок в любом 32-битном типе данных, например в типе `int`. В этом случае легко свести перемножение двух матриц к последовательному применению побитовой операции «И» к парам нужных строк матриц, если вторую матрицу хранить в транспонированном виде. Это позволяет выполнять умножение двух матриц за время порядка $O(n^2)$. Для возведения матрицы в степень можно воспользоваться бинарным алгоритмом [13], который позволяет возводить число или матрицу в степень $k$ за $O(\log k)$ действий, что даёт итоговую временну́ю сложность решения $O(n^2 \log k)$. Так как в нашем случае $k = 3$, имеем сложность $O(n^2)$.

В табл. 3 приводится результат работы программы — число графов с экспонентом 3 для различных $n$ и $p$. Символ «–» означает, что графов с такими $n$ и $p$ не существует. Это может быть в двух случаях: $p \geqslant n$ или произведение $pn$ нечётно. Серый фон клетки означает, что все связные регулярные графы со степенью $p$ и числом вершин $n$ имеют экспонент, равный 2. В работе [6] показано, что это верно при $p > n/2$.

Таблица 3

**Число графов с экспонентом 3 для различных $n$ и $p$**

| $n$ | $p$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 0 | – | – | – | – | – | – |
| 5 | – | 0 | – | – | – | – | – |
| 6 | 1 | 0 | 0 | – | – | – | – |
| 7 | – | 0 | – | 0 | – | – | – |
| 8 | 1 | 3 | 0 | 0 | 0 | – | – |
| 9 | – | 11 | – | 0 | – | 0 | – |
| 10 | 1 | 41 | 35 | 0 | 0 | 0 | 0 |
| 11 | – | 143 | – | 0 | – | 0 | – |
| 12 | 1 | 568 | 7 506 | 2 391 | 0 | 0 | 0 |
| 13 | – | 2 403 | – | 232 080 | – | 0 | – |
| 14 | 0 | 10 377 | 3 093 569 | 18 801 129 | 2 757 433 | 0 | 0 |
| 15 | – | 42 197 | – | 1 429 344 906 | – | 0 | – |
| 16 | 0 | 151 684 | 1 797 671 946 | 112 705 503 963 | 467 764 092 656 | 34 831 303 586 | 0 |

## Заключение

В работе рассмотрен вопрос о максимальном числе вершин в примитивных неориентированных регулярных графах с экспонентом, равным 3. Помимо общей оценки, удалось найти точное значение для кубических графов: не существует кубических графов с экспонентом 3 и числом вершин больше 12. Кубический 12-вершинный граф с экспонентом 3 представлен в работе. Ранее аналогичный вопрос рассматривался для регулярных графов с экспонентом 2, где удалось найти точные значения для регулярных графов порядка 3, 4 и 5: $n_3 = 4$, $n_4 = 11$ и $n_5 = 16$. Приведены результаты вычислительного эксперимента по построению всех регулярных графов с экспонентом 3 и числом вершин до 16, предложено несколько достаточных условий для регулярных графов с экспонентом 3 и проведён их анализ.

## ЛИТЕРАТУРА

1. *Wielandt H.* Unzerlegbare nicht negative Matrizen // Math. Zeitschr. 1950. V. 52. P. 642–648.
2. *Сачков В. Н., Ошкин И. Б.* Экспоненты классов неотрицательных матриц // Дискретная математика. 1993. Т. 5. № 2. С. 150–159.

3. *Салий В. Н.* Минимальные примитивные расширения ориентированных графов // Прикладная дискретная математика. 2008. № 1(1). С. 116–119.
4. *Фомичев В. М.* Оценки экспонентов примитивных графов // Прикладная дискретная математика. 2011. № 2(11). С. 101–112.
5. *Фомичев В. М., Авезова Я. Э.* Точная формула экспонентов перемешивающих орграфов регистровых преобразований // Дискрет. анализ исслед. опер. 2020. № 2(27). С. 117–135.
6. *Абросимов М. Б., Лось И. В., Костин С. В.* Примитивные однородные графы с экспонентом 2 и числом вершин до 16 // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2021. Т. 21. Вып. 2. С. 238–245.
7. *Абросимов М. Б., Костин С. В., Лось И. В.* О наибольшем числе вершин примитивных однородных графов порядка 2, 3, 4 с экспонентом, равным 2 // Прикладная дискретная математика. 2021. № 52. С. 97–104.
8. *Jin M., Lee S. G., and Seol H. G.* Exponents of $r$-regular primitive matrices // Inform. Center for Math. Sci. 2003. V. 6. No. 2. P. 51–57.
9. *Bueno M. I. and Furtado S.* On the exponent of $r$-regular primitive matrices // ELA. Electronic J. Linear Algebra. 2008. V. 17. P. 28–47.
10. *Kim B., Song B., and Hwang W.* Nonnegative primitive matrices with exponent 2 // Linear Algebra Appl. 2005. No. 407. P. 162–168.
11. *Hoffman A. J. and Singleton R. R.* Moore graphs with diameter 2 and 3 // IBM J. Research and Development. 1960. V. 4. P. 497–504.
12. *Meringer M.* Fast generation of regular graphs and construction of cages // J. Graph Theory. 1999. V. 30. P. 137–146.
13. *Смарт Н.* Криптография. М.: Техносфера, 2005. 528 с.

REFERENCES

1. *Wielandt H.* Unzerlegbare nicht negative Matrizen. Math. Zeitschr., 1950, vol. 52, pp. 642–648. (in German)
2. *Sachkov V. N. and Oshkin I. B.* Exponents of classes of non-negative matrices. Discrete Math. Appl., 1993, vol. 3, no. 4, pp. 365–375.
3. *Saliy V. N.* Minimal'nye primitivnye rasshireniya orientirovannykh grafov [Minimal primitive extensions of oriented graphs]. Prikladnaya Diskretnaya Matematika, 2008, no. 1(1), pp. 116–119. (in Russian)
4. *Fomichev V. M.* Otsenki eksponentov primitivnykh grafov [The estimates of exponents for primitive graphs]. Prikladnaya Diskretnaya Matematika, 2011, no. 2(11), pp. 101–112. (in Russian)
5. *Fomichev V. M. and Avezova Ya. E.* The exact formula for the exponents of the mixing digraphs of register transformations. J. Appl. Ind. Math., 2020, no. 14, pp. 308–320.
6. *Abrosimov M. B., Los' I. V., and Kostin S. V.* Primitivnyye odnorodnyye grafy s eksponentom 2 i chislom vershin do 16 [Primitive regular graphs with exponent 2 and up to 16 vertices]. Izvestiya of Saratov University. Mathematics. Mechanics. Informatics, 2021, vol. 21, no. 2, pp. 238–245. (in Russian)
7. *Abrosimov M. B., Kostin S. V., and Los' I. V.* O naibol'shem chisle vershin primitivnykh odnorodnykh grafov poryadka 2, 3, 4 s eksponentom, ravnym 2 [The maximum number of vertices of primitive regular graphs of orders 2, 3, 4 with exponent 2]. Prikladnaya Diskretnaya Matematika, 2021, no. 52, pp. 97–104. (in Russian)
8. *Jin M., Lee S. G., and Seol H. G.* Exponents of $r$-regular primitive matrices. Inform. Center for Math. Sci., 2003, vol. 6, no. 2, pp. 51–57.

9. *Bueno M. I. and Furtado S.* On the exponent of $r$-regular primitive matrices. ELA. Electronic J. Linear Algebra, 2008, vol. 17, pp. 28–47.
10. *Kim B., Song B., and Hwang W.* Nonnegative primitive matrices with exponent 2. Linear Algebra Appl., 2005, no. 407, pp. 162–168.
11. *Hoffman A. J. and Singleton R. R.* Moore graphs with diameter 2 and 3. IBM J. Research and Development, 1960, vol. 4, pp. 497–504.
12. *Meringer M.* Fast generation of regular graphs and construction of cages. J. Graph Theory, 1999, vol. 30, pp. 137–146.
13. *Smart N. P.* Cryptography: An Intoduction. McGraw-Hill, 2003. 433 p.

# МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52 DOI 10.17223/20710410/67/6

## О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ РЕШЕНИЯ УРАВНЕНИЙ НАД БИЦИКЛИЧЕСКИМ МОНОИДОМ[1]

А. А. Лопатин*, А. Н. Рыбалов**

*Государственный университет города Кампинас, г. Кампинас, Сан-Паулу, Бразилия
**Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

**E-mail:** dr.artem.lopatin@gmail.com, alexander.rybalov@gmail.com

Изучается вычислительная сложность проблемы проверки разрешимости систем уравнений над бициклическим моноидом. Этот моноид, помимо теоретического значения в топологии и теории полугрупп, имеет приложения в информатике и языках программирования, например как модель для языка Дика сбалансированных скобочных выражений. Доказывается NP-полнота проблемы проверки разрешимости систем уравнений над бициклическим моноидом. Также доказывается, что при $P \neq NP$ и $P = BPP$ для этой проблемы не существует сильно генерического полиномиального алгоритма. Это означает, что для любого генерического полиномиального алгоритма имеется эффективный метод случайной генерации входов, на которых этот алгоритм не может решить рассматриваемую проблему. Полученный результат указывает на возможное применение данной проблемы в криптографии, где нужно, чтобы проблема взлома криптосистемы была трудной для почти всех входов.

**Ключевые слова:** *генерическая сложность, NP-полнота, бициклический моноид.*

## ON GENERIC COMPLEXITY OF EQUATION SOLVING OVER THE BICYCLIC MONOID

A. A. Lopatin*, A. N. Rybalov**

*Universidade Estadual de Campinas (UNICAMP), Brazil
**Sobolev Institute of Mathematics, Omsk, Russia

In this paper, we study computational complexity of the problem of determining solvability equations over bicyclic monoid. This monoid, in addition to its theoretical significance in topology and semigroup theory, has applications in computer science and programming languages, for example, as a model for the Dyck language of balanced bracket expressions. We prove NP-completeness of the problem of determining solvability equations over bicyclic monoid. Also, we prove that if $P \neq NP$ and $P = BPP$, then for this problem there is no strongly generic polynomial algorithm. This means

[1]Первый автор поддержан грантом FAPESP 2023/17918-2. Работа второго автора выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0003.

that for any generic polynomial algorithm there exists an efficient method of randomly generating inputs on which the algorithm cannot solve the problem under consideration. The result points to a possible application of this problem in cryptography, where it is necessary that the problem of breaking a cryptosystem be hard for almost all inputs. To prove this theorem, we use the method of generic amplification, which allows to construct generically hard problems from the problems hard in the classical sense. The basis of this method is a technique of cloning, which combines the input data of a problem into sufficiently large sets of equivalent input data. Equivalence is understood in the sense that the problem is solved similarly for them.

**Keywords:** *generic complexity, NP-completeness, bicyclic monoid.*

## Введение

Решение уравнений и систем уравнений над вещественными, комплексными, рациональными, целыми числами является классической темой исследований в различных областях математики в течение тысяч лет. Классическая алгебраическая геометрия изучает множества решений алгебраических уравнений над полями вещественных и комплексных чисел. В рамках диофантовой геометрии и диофантова анализа изучаются решения алгебраических уравнений над целыми и рациональными числами. В XX в. большую роль начали играть вычислительные аспекты этих теорий. Изучение алгоритмических проблем, связанных с определением наличия решения у систем уравнений, а также с нахождением и описанием множества решений, является темой многочисленных теоретических и практических исследований.

В последние десятилетия фокус исследований перемещается на неклассические области, такие, как группы [1], полугруппы [2, 3], графы [4], частичные порядки [5]. Потребность решения уравнений в этих системах возникает при рассмотрении различных практических проблем информатики, криптографии, теории языков программирования. Например, свободные полугруппы являются базисом для описания важнейших классов формальных языков и грамматик: регулярных, контекстно свободных. Часто при этом изучаемый формальный язык задаётся некоторым набором уравнений, множество решений которых даёт нужный язык. К необходимости решения уравнений над графами приводят задачи проверки вложимости (совместимости) одной коммуникационной сети в другую сеть.

К сожалению, как правило, проблема решения систем уравнений над различными алгебраическими системами является либо неразрешимой, либо имеет большую вычислительную сложность. Даже над конечными алгебраическими системами эта проблема оказывается NP-полной. Это означает, что при условии $\mathrm{P} \neq \mathrm{NP}$ для неё не существует полиномиальных алгоритмов. Поэтому актуальным является изучение генерической сложности [6] данных проблем. В рамках генерического подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. С одной стороны, положительные результаты о возможности эффективного решения каких-либо трудных задач для почти всех входов полезны для практики. С другой стороны, негативные результаты о генерической трудности некоторых проблем дают надежду на возможное их использование в криптографии, где как раз важно, чтобы проблема взлома криптосистемы была трудной для почти всех входов. Генерическая сложность проблем решения уравнений над конечными полями и полугруппами изучалась в [7].

В данной работе рассматривается проблема разрешимости уравнений над бициклическим моноидом $B = \langle a, b \,|\, ab = e \rangle$. Этот моноид, помимо теоретического значения

в топологии и теории полугрупп, имеет приложения в информатике и языках программирования, например, как модель для так называемого языка Дика сбалансированных скобочных выражений [8]. Что касается проблемы разрешимости уравнений над бициклическим моноидом, то известно, что она разрешима [9]. Однако вопрос о вычислительной сложности этой проблемы пока не изучался. В настоящей работе исследуется вычислительная сложность проблемы проверки разрешимости систем уравнений над бициклическим моноидом.

## 1. Предварительные сведения

*Бициклическим моноидом* будем называть моноид, заданный конечно определённым представлением $B = \langle a, b \,|\, ab = e\rangle$, где $e$ — пустое слово, выполняющее роль единицы, с операцией умножения — конкатенацией слов. *Нормальной формой* слова $w$ над алфавитом $\{a, b\}$ в $B$ назовём слово $b^m a^n$. Очевидно, любой элемент моноида $B$ можно привести к такой нормальной форме. Под *размером* элемента $b^m a^n$ будем понимать сумму длин двоичных записей чисел $m$ и $n$. Легко подсчитать, что нормальной формой произведения двух произвольных элементов $x = b^m a^n$ и $y = b^k a^l$ в бициклическом моноиде $B$ является

$$x \cdot y = b^m a^n \cdot b^k a^l = b^{m+k-\min\{n,k\}} \cdot a^{l+n-\min\{n,k\}}.$$

*Уравнение* от переменных $x_1, \ldots, x_n$ над бициклическим моноидом $B$ имеет вид

$$c_1 x_1 c_2 \ldots c_m x_m c_{m+1} = c_{m+2} x_{m+1} c_{m+3} \ldots c_{n+1} x_n c_{n+2},$$

где $c_1, \ldots, c_{n+2} \in B$. Набор элементов $d_1, \ldots, d_n$ является *решением* этого уравнения, если при его подстановке вместо соответствующих переменных получается верное равенство. Под *размером* уравнения будем понимать сумму размеров элементов $c_1, \ldots, c_{n+2}$ плюс число переменных $n$. *Системой уравнений* называется конечный набор уравнений. *Решением* системы называется такой набор элементов, который является одновременно решением всех уравнений данной системы. Под *размером* системы будем понимать сумму размеров всех её уравнений. *Проблема совместности систем уравнений над бициклическим моноидом* состоит в следующем: по произвольной заданной системе уравнений над $B$ определить, имеет ли она решение в $B$.

Напомним, что алгоритмическая проблема распознавания $A \subseteq I$ принадлежит *классу* NP, если существует полиномиальный алгоритм $\mathcal{A}$ и полином $p(n)$, такие, что

$$x \in A \Leftrightarrow \exists y \in I \ \big(|y| < p(|x|) \ \&\ \mathcal{A}(x, y) = 1\big).$$

Здесь через $|x|$ обозначена длина входа $x$. Элемент $y$ называют подсказкой и говорят, что алгоритм $\mathcal{A}$ проверяет эту подсказку.

Алгоритмическая проблема распознавания $A \subseteq I$ называется NP-*трудной*, если к ней за полиномиальное время сводится любая проблема $B \subseteq I$ из класса NP. То есть существует функция $f : I \to I$, вычислимая некоторым полиномиальным алгоритмом, такая, что

$$x \in B \Leftrightarrow f(x) \in A.$$

Если при этом NP-трудная проблема сама принадлежит классу NP, то она называется NP-*полной*.

Основные определения генерического подхода можно найти в [6]. Определения вычислительных классов P, NP и BPP содержатся в [10].

## 2. NP-полнота

Нетрудно показать, что любую систему уравнений над бициклическим моноидом можно привести к эквивалентной системе, в которой каждое уравнение является равенством одного из следующих типов:

1) $x_i = x_j x_k$;
2) $x_i = a$;
3) $x_i = b$.

Будем называть такие системы уравнений *системами в форме Сколема*. Размер такой системы — это число уравнений в ней. Заметим, что при таком переходе размер системы в форме Сколема по сравнению с размером исходной системы увеличивается не более чем полиномиально.

**Теорема 1.** Проблема проверки совместности систем уравнений над бициклическим моноидом является NP-полной.

***Доказательство.*** Докажем сначала принадлежность проблемы классу NP. По заданной системе уравнений $S$ над $B$ построим эквивалентную ей систему в форме Сколема $S'$. Затем по системе $S'$ построим бесквантерную формулу $\Phi_{S'}$ арифметики Пресбургера (то есть над натуральными числами со сложением, вычитанием и предикатом порядка $<$), такую, что выполнимость $\Phi_{S'}$ равносильна существованию решения системы $S$. Для этого каждой переменной $x$ системы $S'$ сопоставим пару натуральнозначных переменных $(y, z)$, которая соответствует нормальной форме $x = b^y a^z$. Каждому уравнению системы $S'$ сопоставим формулу арифметики Пресбургера следующим образом:

1) если уравнение имеет вид $x_i = x_j x_k$, то соответствующая формула будет такой:

$$\Big((y_k \leqslant z_j)\&(y_i = y_j)\&(z_i = z_j - y_k + z_k)\Big) \vee \Big((z_j < y_k)\&(y_i = y_k - z_j + y_j)\&(z_i = z_k)\Big).$$

   Эта формула соответствует правилу изменения степеней букв $a$ и $b$ в нормальной форме элементов бициклического моноида при перемножении $x_j = b^{y_j} a^{z_j}$ на $x_k = b^{y_k} a^{z_k}$;
2) если уравнение имеет вид $x_i = a$, то соответствующая формула будет такой:

$$(y_i = 0)\&(z_i = 1);$$

3) если уравнение имеет вид $x_i = b$, то соответствующая формула будет такой:

$$(y_i = 1)\&(z_i = 0).$$

Итоговая формула $\Phi_{S'}$ есть коньюнкция формул, соответствующих всем уравнениям системы $S'$. Из построения формулы $\Phi_{S'}$ следует, что система $S'$ совместна над $B$ тогда и только тогда, когда формула $\Phi_{S'}$ выполнима в арифметике Пресбургера. Из свойств целочисленных решений систем линейных неравенств (см., например, [11]) следует, что если $\Phi_{S'}$ выполнима, то размер в двоичной записи минимальных значений переменных, делающих истинной формулу $\Phi_{S'}$, ограничен полиномиально от длины $\Phi_{S'}$. Таким образом, в качестве подсказки (решения) можно взять значения этих переменных. Это доказывает принадлежность проблемы совместности систем уравнений в бициклическим моноиде классу NP.

Докажем теперь, что к проблеме совместности систем уравнений в бициклическим моноиде полиномиально сводится известная NP-полная проблема о выполнимости

3-КНФ, которая заключается в следующем: 3-КНФ $\Phi(x_1, \ldots, x_n)$ — это конъюнкция дизъюнкций вида $(\alpha_1 \vee \alpha_2 \vee \alpha_3)$, где $\alpha_i$, $i = 1, 2, 3$, есть либо булева переменная $x_k$, либо отрицание булевой переменной $\overline{x_k}$. Нужно определить, является ли заданная 3-КНФ $\Phi(x_1, \ldots, x_n)$ выполнимой, то есть существуют ли значения $a_1, \ldots, a_n \in \{0, 1\}^n$, такие, что $\Phi(a_1, \ldots, a_n) = 1$.

Построим по 3-КНФ $\Phi(x_1, \ldots, x_n)$ систему уравнений над бициклическим моноидом, которая имеет решение тогда и только тогда, когда $\Phi(x_1, \ldots, x_n)$ выполнима. Для этого каждой булевой переменной из $\Phi$ сопоставим две переменные $x$ и $y$, а также запишем уравнения

$$xa = ax,$$
$$ya = ay,$$
$$xy = a.$$

Первые два уравнения гарантируют, что возможные значения для $x$ и $y$ принадлежат подмоноиду, порождённому $a$, а третье — что $x, y$ могут принимать только значения $a$ или $e$. Элемент $a$ выполняет роль логической 1, а $e$ — роль логического 0. У самих переменных роли таковы: $x$ моделирует соответствующую логическую переменную, а $y$ — её отрицание.

Далее смоделируем дизъюнкцию $(\alpha_1 \vee \alpha_2 \vee \alpha_3)$. Для этого возьмём переменные $z_1, z_2, z_3$ над бициклическим моноидом, соответствующие переменным (или их отрицаниям) из дизъюнкции, и запишем уравнения

$$z_1 z_2 z_3 = au,$$
$$ua = au,$$

где $u$ — некоторая новая переменная. Заметим, что эта система уравнений разрешима в бициклическом моноиде тогда и только тогда, когда хотя бы одна из переменных $z_1, z_2, z_3$ равна $a$, что соответствует истинности соответствующей дизъюнкции.

Теперь по каждой дизъюнкции 3-КНФ $\Phi(x_1, \ldots, x_n)$ строим подобные уравнения и получаем систему $S_\Phi$, которая имеет решения в бициклическом моноиде тогда и только тогда, когда 3-КНФ $\Phi(x_1, \ldots, x_n)$ выполнима. Полиномиальность данной сводимости следует из описанного процесса построения. ■

## 3. Генерическая трудноразрешимость

Будем называть систему в форме Сколема *нормализованной*, если в $k$-м уравнении системы могут встречаться только переменные $x_i$, где $i \leqslant 3k$. Очевидно, что любую систему в форме Сколема можно нормализовать с помощью подходящей перенумерации переменных. Обозначим через $\mathcal{S}$ множество всех нормализованных систем уравнений в форме Сколема.

**Лемма 1.** Число нормализованных систем уравнений в форме Сколема размера $n$ равно

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (27k^3 + 6k). \tag{1}$$

***Доказательство.*** Для $t$-го уравнения в системе $S \in \mathcal{S}_n$ существует $(3t)^3$ вариантов выбрать уравнение вида $x_i = x_j x_k$ и $6t$ вариантов выбрать уравнение вида $x_i = a$ или $x_i = b$. Итого для $t$-го уравнения есть $27t^3 + 6t$ вариантов и для системы из $n$ уравнений выполняется (1). ■

Для произвольной системы уравнений $S = \{S_1, \ldots, S_m\}$ рассмотрим множество систем $eq(S)_n$, которые получаются добавлением к системе $S$ произвольных уравнений $S_{m+1}, \ldots, S_n$, где $l$-е уравнение имеет вид $x_i = x_j x_k$, причём $3m < i, j, k < 3(l+m)$. Очевидно, что любая система из $eq(S)_n$ совместна над $B$ тогда и только тогда, когда совместна над $B$ система $S$.

**Лемма 2.** Для любой системы $S$ размера $m$ и любого $n > m$ имеет место оценка

$$\frac{|eq(S)_n|}{|\mathcal{S}_n|} > \frac{1}{2(33n^3)^m}.$$

***Доказательство.*** Пусть $n > m$. Для $t$-го добавленного к $S$ уравнения вида $x_i = x_j x_k$, где $3m < i, j, k < 3(t+m)$, имеется $(3t)^3 = 27t^3$ вариантов. Поэтому

$$|eq(S)_n| = \prod_{t=1}^{n-m} (27t^3).$$

Теперь по лемме 1 получим

$$\rho_n(eq(S)_n) = \frac{|eq(S)_n|}{|\mathcal{S}_n|} = \frac{\prod\limits_{k=1}^{n-m} (27k^3)}{\prod\limits_{k=1}^{n} (27k^3 + 6k)} = \prod_{k=1}^{n-m} \Big(\frac{27k^2}{27k^2+6}\Big) \prod_{k=n-m+1}^{n} \frac{1}{27k^3+6k}.$$

Оценим снизу первое произведение:

$$\prod_{k=1}^{n-m} \Big(\frac{27k^2}{27k^2+6}\Big) = \prod_{k=1}^{n-m} \Big(1 - \frac{1}{9k^2+2}\Big) > \prod_{k=2}^{n-m+1} \Big(1 - \frac{1}{k^2}\Big) = \prod_{k=2}^{n-m+1} \frac{(k-1)(k+1)}{k^2} =$$
$$= \frac{1\cdot 3}{2^2} \frac{2\cdot 4}{3^2} \cdot \ldots \cdot \frac{(n-m-1)(n-m+1)}{(n-m)^2} \frac{(n-m)(n-m+2)}{(n-m+1)^2} = \frac{n-m+2}{2(n-m+1)} > \frac{1}{2}.$$

Теперь оценим второе произведение:

$$\prod_{k=n-m+1}^{n} \frac{1}{27k^3+6k} > \frac{1}{(27n^3+6n)^m}.$$

Итого получаем $\rho_n(eq(S)) = \dfrac{|eq(S)_n|}{|\mathcal{S}_n|} > \dfrac{1}{2(27n^3+6n)^m} > \dfrac{1}{2(33n^3)^m}$. ■

**Теорема 2.** Пусть P $\neq$ NP и P = BPP. Тогда для проблемы проверки совместности нормализованных систем уравнений в форме Сколема над бициклическим моноидом не существует сильно генерического полиномиального алгоритма.

***Доказательство.*** Допустим, что существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, определяющий совместность систем уравнений над $B$. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, решающий эту проблему на всём множестве входов. На системе $S$ размера $n$ алгоритм $\mathcal{B}$ работает следующим образом:

1) генерирует случайно и равновероятно систему $S'$ из множества $eq(S)$ размера $n^2$;
2) запускает алгоритм $\mathcal{A}$ на системе $S'$;
3) если $\mathcal{A}(S') \neq ?$, то алгоритм правильно определяет, совместна ли система $S'$, а вместе с ней и система $S$;
4) если $\mathcal{A}(S') = ?$, то выдаёт ответ «НЕТ».

Заметим, что полиномиальный вероятностный алгоритм $\mathcal{B}$ выдаёт правильный ответ на шаге 3, а на шаге 4 может выдать неправильный ответ. Надо доказать, что вероятность того, что ответ выдаётся на шаге 4, меньше $1/3$.

Оценим вероятность выдачи ответа на шаге 4. Вероятность того, что для $S'$ имеет место $\mathcal{A}(S') = ?$, не больше

$$\frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|eq(S)_{n^2}|} = \frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|\mathcal{S}_{n^2}|} \frac{|\mathcal{S}_{n^2}|}{|eq(S)_{n^2}|}.$$

Так как множество $\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}$ сильно пренебрежимое, то существует константа $\alpha > 0$, такая, что

$$\frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|\mathcal{S}_{n^2}|} < \frac{1}{2^{\alpha n^2}}$$

для любого $n$. По лемме 2

$$\frac{|\mathcal{S}_{n^2}|}{|eq(S)_{n^2}|} < 2(33n^6)^n.$$

Поэтому искомая вероятность ответа на шаге 4 не больше

$$\frac{2(33n^6)^n}{2^{\alpha n^2}} = \frac{2^{1+\log 33n+6n\log n}}{2^{\alpha n^2}} = \frac{1}{2^{\alpha n^2-6n\log n-\log 33n-1}} < \frac{1}{3}$$

при достаточно больших $n$.

Таким образом, проблема проверки совместности систем уравнений над $B$ принадлежит классу BPP. Так как BPP = P, она принадлежит классу P. Это, по теореме 1, противоречит условию P $\neq$ NP. ■

Авторы выражают искреннюю благодарность рецензенту за полезные замечания и предложения по улучшению текста статьи.

## ЛИТЕРАТУРА

1. *Baumslag G., Myasnikov A., and Remeslennikov V.* Algebraic geometry over groups I. Algebraic sets and ideal theory // J. Algebra. 1999. V. 219. No. 1. P. 16–79.
2. *Shevlyakov A. N.* Equationally Noetherian varieties of semigroups and B. Plotkin's problem // Сиб. электрон. матем. изв. 2023. Т. 20. № 2. С. 724–734.
3. *Шевляков А. Н.* Сплетения полугрупп и проблема Б. И. Плоткина // Алгебра и логика. 2023. Т. 62. № 5. С. 665–691.
4. *Рыбалов А. Н.* О сложности решения уравнений над графами // Сиб. электрон. матем. изв. 2024. Т. 21. № 1. С. 62–69.
5. *Nikitin A. Yu. and Shevlyakov A. N.* On radicals over strict partial order sets // J. Physics: Conf. Ser. 2021. V. 1791. No. 012080. P. 1–6.
6. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
7. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields // J. Physics: Conf. Ser. 2021. V. 1901. No. 012047. P. 1–8.
8. *Hopcroft J. E. and Ullman J. D.* Formal Languages and Their Relation to Automata. Addison-Wesley, 1969. 288 p.
9. *Diekert V. and Lohrey M.* Word equations over graph products // Intern. J. Algebra Computation. 2008. V. 18. No. 3. P. 493–533.

10. *Китаев А., Шень А., Вялый М.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999. 192 с.
11. *Схрейвер А.* Теория линейного и целочисленного программирования. М.: Мир, 1991. 360 с.

REFERENCES

1. *Baumslag G., Myasnikov A., and Remeslennikov V.* Algebraic geometry over groups I. Algebraic sets and ideal theory. J. Algebra, 1999, vol. 219, no. 1, pp. 16–79.
2. *Shevlyakov A. N.* Equationally Noetherian varieties of semigroups and B. Plotkin's problem. Siberian Electronic Math. Reports, 2023, vol. 20, no. 2, pp. 724–734.
3. *Shevlyakov A. N.* Spleteniya polugrupp i problema B. I. Plotkina [Wreath products of semigroups and Plotkin's problem]. Algebra i Logika, 2023, vol. 62, no. 5, pp. 665–691. (in Russian)
4. *Rybalov A. N.* O slozhnosti resheniya uravneniy nad grafami [On complexity of solving of equations over graphs]. Siberian Electronic Math. Reports, 2024, vol. 21, no. 1, pp. 62–69. (in Russian)
5. *Nikitin A. Yu. and Shevlyakov A. N.* On radicals over strict partial order sets. J. Physics: Conf. Ser., 2021, vol. 1791, no. 012080, pp. 1–6.
6. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
7. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields. J. Physics: Conf. Ser., 2021, vol. 1901, no. 012047, pp. 1–8.
8. *Hopcroft J.E. and Ullman J.D.* Formal Languages and Their Relation to Automata. Addison-Wesley, 1969. 288 p.
9. *Diekert V. and Lohrey M.* Word equations over graph products. Intern. J. Algebra Computation, 2008, vol. 18, no. 3, pp. 493–533.
10. *Kitaev A., Shen' A., and Vyalyy M.* Klassicheskie i kvantovye vychisleniya [Classical and Quantum Computations]. Moscow, MCCME Publ., 1999. 192 p. (in Russian)
11. *Schrijver A.* Theory of Linear and Integer Programming. Wiley, 1998. 484 p.

# ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.8 DOI 10.17223/20710410/67/7

## КОНСТРУКТИВНЫЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЙ НА ДВУХ ПРОЦЕССОРАХ С КРИТЕРИЕМ МАКСИМАЛЬНОГО ВРЕМЕННОГО СМЕЩЕНИЯ ПРИ УЧЁТЕ РАСПАРАЛЛЕЛИВАНИЯ И РАСХОДА ЭНЕРГИИ[1]

Ю. В. Захарова*, А. О. Евтина**

*Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия
**Омский государственный университет им. Ф. М. Достоевского, г. Омск, Россия

**E-mail:** yzakharova@ofim.oscsbras.ru, aevtina55@gmail.com

Проводится теоретический и экспериментальный анализ вычислительной сложности одной актуальной задачи теории расписаний, возникающей в компьютерных системах и приложениях. Особенностью постановки является возможность распараллеливания операций и учёт ресурсных ограничений, влияющих на длительности операций. Критерием выступает минимизация максимального временного смещения. Исследуется вопрос труднорешаемости задачи и предлагаются алгоритмы с гарантированными оценками точности. Результаты экспериментальных исследований показывают перспективность предложенных алгоритмов.

**Ключевые слова:** *расписание, ресурс, алгоритм, NP-трудность.*

## CONSTRUCTIVE ALGORITHMS FOR THE SCHEDULING PROBLEM ON TWO PROCESSORS WITH THE MAXIMUM TIME OFFSET CRITERION TAKING INTO ACCOUNT PARALLELIZATION AND ENERGY CONSUMPTION

Y. V. Zakharova*, A. O. Evtina**

*Sobolev Institute of Mathematics, Omsk, Russia
**Dostoevsky Omsk State University, Omsk, Russia

We provide theoretical and experimental analysis of the computational complexity of one scheduling problem arising in computer systems and applications. The feature of the statement is the parallelization of operations and resource-dependent durations of operations. Here the processor speed affects the duration of operations and power consumption. For each operation the number of required processors is given. The criterion is the minimization of the maximum lateness under the given energy budget. We prove that the problem is NP-hard using the polynomial reduction from the well-known 3-Partition problem and constructing non-idle instances. An approximate algorithm with polynomial running time is proposed. The algorithm consists

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0020.

of reducing the two-processor problem to the single-processor one. A convex program is proposed for solving the single-processor problem. We investigate the block properties of the solutions generated by the algorithm and show that their objective value is at most doubled optimum maximum lateness plus the maximal due date. The experimental results show the applicability of the proposed algorithm. We construct a series of instances in which the relative deviation from the lower bound is less than 15 %. We also experimentally identify a tendency in decreasing the relative deviation when the number of fully parallelizable operations increases. The theoretical analysis guarantees that the problem is polynomially solvable when all operations are fully parallelizable.

**Keywords:** *schedule, resource, algorithm, NP-hardness.*

## Введение

В работе рассматриваются задачи составления расписания с малым числом процессоров, возникающие в компьютерных системах при планировании работы приложений на многоядерных процессорах. Современные компьютерные системы обуславливают необходимость или возможность обработки одной задачи несколькими процессорами, в микропроцессорных системах некоторые задачи должны выполняться на нескольких процессорах одновременно, кроме того, один процессор может использоваться для тестирования или проверки другого процессора [1].

Задачи составления расписаний широко исследованы при различных типах распараллеливания операций (задано число используемых процессоров или верхняя граница степени распараллеливания; заданы подмножества процессоров, которые могут использоваться для работы приложения). Исследованы вопросы труднорешаемости различных частных случаев, предложены модели целочисленного линейного программирования, алгоритмы с гарантированной точностью и метаэвристики [1].

Процессоры, выполняя различные приложения, имеют возможность совместного использования общих ресурсов (ядра, кэш-памяти, шины данных, энергии и т. д.), поэтому в задачах планирования могут быть заданы ограничения на ресурсы. Память относится к возобновимым ресурсам, при этом ограничивается возможность размещения операций на процессоре и совмещения с другими операциями в системах с общей памятью. Для вариантов задачи составления расписаний с возобновимыми ресурсами предложены алгоритмы динамического программирования, модели целочисленного программирования и метаэвристики [2–4]. Классу возобновимых ресурсов принадлежит также пропускная способность шины данных. Во время передачи информации по шине данных операции могут замедлять друг друга, когда выполняются одновременно, таким образом, скорость выполнения работы может меняться в зависимости от загрузки других ядер процессора [5]. Для задачи составления расписаний с учётом замедления операций при использовании шины данных построена модель целочисленного программирования, разработан алгоритм жадного типа и проведено экспериментальное исследование на реальных данных [6].

Энергия относится к невозобновимым ресурсам. Процессоры могут работать с различной скоростью, что влияет на длительность выполнения операций и потребление энергии. Задачам составления расписаний с учётом расхода энергии посвящены, например, работы [7–10]. Здесь рассматриваются задачи с бюджетными ограничениями на расход энергии при критериях, в основном, минимизации длины расписания и суммы моментов завершения операций, а также задачи с энергетическим критерием. Предлагаются приближённые алгоритмы с гарантированными оценками точности и

доказывается NP-трудность в частных случаях. Например, для задачи с распараллеливаемыми операциями на двух процессорах предложен 3/2-приближённый алгоритм для критерия «длина расписания» и 2-приближённый алгоритм для критерия «суммарное время завершения».

Задача с двумя устройствами актуальна, например, в вычислительных системах с конфигурацией из двух NUMA-узлов или с двухъядерными процессорами. Более того, подходы на основе декомпозиции могут использовать решение одно- и двухпроцессорных подзадач [1, 9].

В настоящей работе исследуется задача составления операций с учётом расхода энергии при возможности распараллеливания операций. В качестве критерия оптимизации в отличие от предыдущих исследований рассматривается минимизация максимального временного смещения $L_{\max}$. В п. 1 рассматривается задача, где учитывается только возможность распараллеливания операций, и предлагается приближённый алгоритм с гарантированной оценкой точности. В п. 2 осуществляется переход к задаче с учётом расхода энергии, доказывается её NP-трудность, адаптируется алгоритм из п. 1. Здесь же предлагается модель выпуклого программирования, позволяющая получить оптимальное решение за полиномиальное время в случае одного процессора. В п. 3 для оценки разработанных алгоритмов представлены результаты экспериментальных исследований.

## 1. Постановка задачи и алгоритм решения

Рассматривается задача составления расписания на одном или двух процессорах ($m = 1$ или 2) [11]. Для каждой операции (работы) $j$ из $J = \{1, \ldots, n\}$ заданы длительность $p_j$, количество требуемых процессоров $\mathrm{size}_j$ и директивный срок $d_j$. Все операции поступают в момент времени 0, прерывания не допускаются. Процессоры обозначаются через $M_1, M_2$.

Рассматривается критерий минимизации максимального временного смещения $L_{\max} = \max\limits_{j \in J}\{L_j\}$, где $L_j = c_j - d_j$ — временное смещение операции $j$; $c_j$ — момент окончания операции $j$, $j \in J$. Данная задача обозначается как $P|\mathrm{size}_j|L_{\max}$ [1] и является NP-трудной в сильном смысле [12], причём она остается NP-трудной, даже если все операции являются однопроцессорными [13].

В случае $m = 1$ и $\mathrm{size}_j = 1$ для всех операций $j \in J$ задача полиномиально разрешима. Для построения оптимального решения операции необходимо упорядочить по неубыванию их директивных сроков [13]. Обозначим через $\pi = (\pi_1, \ldots, \pi_n)$ указанную последовательность операций, тогда оптимальное временное смещение равно

$$L^*_{\max} = \max_{i \in \{1,\ldots,n\}} \left\{ \sum_{l=1}^{i} p_{\pi_l} - d_{\pi_i} \right\}.$$

### 1.1. Два процессора

Опишем приближённый алгоритм с гарантированной оценкой точности, основанный на сведении задачи с двумя процессорами к задаче с одним процессором.

**Алгоритм** A2-1-2 для задачи $P2|\mathrm{size}_j|L_{\max}$:

1) Задача $P$ на двух процессорах сводится к задаче $P1$ на одном процессоре так, что директивные сроки не меняются, а длительности пересчитываются для каждой операции $j \in J$ по следующему правилу:
   - если $\mathrm{size}_j = 1$, то $p'_j = p_j/2$;
   - если $\mathrm{size}_j = 2$, то $p'_j = p_j$.

2) Строится последовательность операций $\pi$, соответствующая их упорядочению по неубыванию директивных сроков, то есть $d_{\pi_i} \leqslant d_{\pi_{i+1}}$. Эта последовательность даёт оптимальное решение задачи $P1$.
3) Для задачи $P$ допустимое решение строится по следующему правилу: на шаге $i = 1, \ldots, n$ операция $\pi_i$ назначается в расписание так, что если $\text{size}_{\pi_i} = 1$, то операция ставится на процессор с минимальным временем завершения текущего набора операций, а если $\text{size}_{\pi_i} = 2$, то операция ставится на два процессора с момента времени, когда они оба доступны.

Пример работы алгоритма представлен на рис. 1. Рассматривается пример задачи $P$ с двумя процессорами и девятью работами; вектор длительностей: $p = (3, 2, 5, 4, 4, 2, 4, 7, 6)$, вектор директивных сроков: $d = (10, 3, 7, 6, 5, 8, 9, 13, 12)$, вектор размеров: $\text{size} = (2, 1, 2, 1, 1, 1, 1, 2, 1)$. При переходе к задаче $P1$ длительности однопроцессорных работ уменьшаются в 2 раза и новый вектор длительностей равен $p' = (3, 1, 5, 2, 2, 1, 2, 7, 3)$. Оптимальное расписание для задачи $P1$ соответствует перестановке $\pi = (2, 5, 4, 3, 6, 7, 1, 9, 8)$. Согласно этой же перестановке, завершаются операции в расписании для $P$.

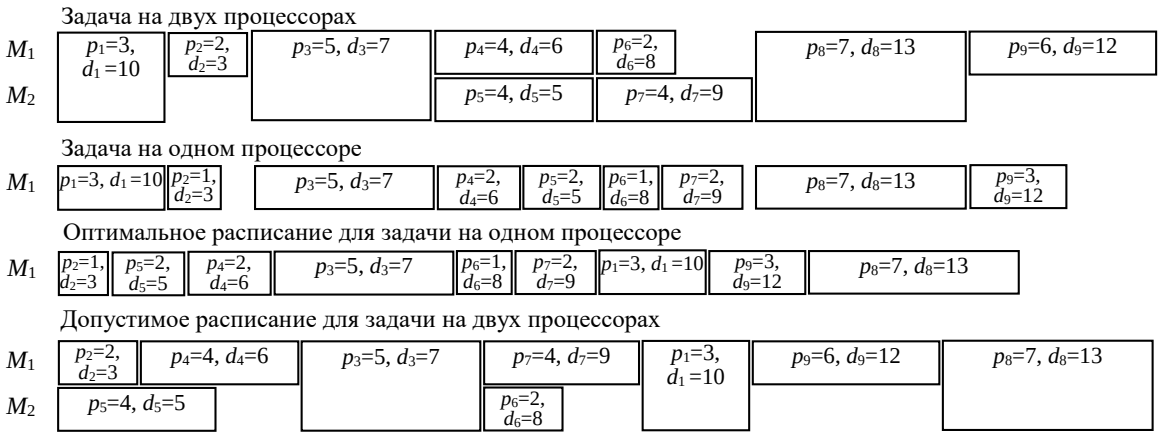


Рис. 1. Иллюстрация к работе алгоритма A2-1-2

Под ранним расписанием, соответствующим произвольной последовательности $\sigma$, будем понимать расписание, при котором операции завершают выполнение согласно $\sigma$, а начинаются в минимальные моменты времени, раньше которых они начаться не могут. Справедлива следующая

**Лемма 1.** Пусть последовательность $\sigma$ задаёт порядок завершения операций; $L_{\max}(P, \sigma)$ — максимальное временное смещение в раннем расписании для задачи $P$ на двух процессорах, соответствующем последовательности $\sigma$; $L_{\max}(P1, \sigma)$ — максимальное временное смещение в раннем расписании для задачи $P1$ на одном процессоре, соответствующем последовательности $\sigma$. Тогда $L_{\max}(P, \sigma) \geqslant L_{\max}(P1, \sigma)$.

***Доказательство.*** Перестановка $\sigma$ задает последовательность, согласно которой операции завершают выполнение. Обозначим через $c_j(P, \sigma)$ момент окончания операции $j \in J$ в задаче $P$ на двух процессорах и через $c_j(P1, \sigma)$ — в задаче $P1$ на одном процессоре согласно раннему расписанию, соответствующему последовательности $\sigma$.

Для задачи $P$ разделим последовательность на блоки: первый блок включает все однопроцессорные операции и первую встретившуюся двухпроцессорную операцию, во

второй блок входят все однопроцессорные операции до следующей двухпроцессорной операции и т. д. В каждом блоке при переходе к задаче на одном процессоре ставим на один процессор операции в том же порядке, в котором они заканчивались в двухпроцессорной задаче. Тогда для любой операции её время завершения на одном процессоре $c_j(P1,\sigma)$ станет меньше $c_j(P,\sigma)$ или останется таким же за счёт того, что длительности однопроцессорных заданий уменьшаются в 2 раза и предыдущие работы размещаются в том же порядке (так как предыдущие работы располагаются плотно и без простоев в двухпроцессорном расписании, в однопроцессорном расписании при уменьшении длительностей работ в 2 раза момент окончания работы может только уменьшиться). Длительности двухпроцессорных работ не меняются, поэтому моменты окончания тоже не увеличиваются. Тогда $c_j(P1,\sigma) \leqslant c_j(P,\sigma)$, $j \in J$. В итоге $L_{\max}(P,\sigma) = \max\limits_{j\in J}\{c_j(P,\sigma) - d_j\} \leqslant \max\limits_{j\in J}\{c_j(P1,\sigma) - d_j\} = L_{\max}(P1,\sigma)$. ■

**Теорема 1.** Если $L_{\max}(P)$ — значение целевой функции расписания, построенного для задачи $P$ с помощью алгоритма A2-1-2, а $L^*_{\max}(P)$ — оптимальное значение целевой функции для задачи $P$, то $L_{\max}(P) \leqslant 2L^*_{\max}(P) + \max\limits_{j\in J} d_j$.

***Доказательство.*** Момент окончания операции $j \in J$ в расписании, построенном алгоритмом A2-1-2 для задачи $P$, обозначим через $c_j(P)$, а в оптимальном расписании для задачи $P1$ — через $c_j(P1)$.

По построению в расписании, полученном для $P$ алгоритмом A2-1-2, для операции $j \in J$ имеет место

$$L_j(P) = c_j(P) - d_j \leqslant 2c_j(P1) - d_j = c_j(P1) - d_j + c_j(P1) - d_j + d_j = 2(c_j(P1) - d_j) + d_j.$$

Отметим, что $c_j(P) \leqslant 2c_j(P1)$ для любой операции $j \in J$, это следует из того, что при переходе к двум процессорам момент окончания каждой операции увеличится не больше чем в 2 раза, так как длительности увеличиваются не больше чем в 2 раза. Тогда для любой операции $j$ имеем $L_j(P) \leqslant 2(c_j(P1) - d_j) + d_j$, следовательно,

$$L_{\max}(P) \leqslant 2L^*_{\max}(P1) + \max_{j\in J} d_j,$$

где $L^*_{\max}(P1)$ есть оптимальное временное смещение для задачи $P1$.

Обозначим через $\sigma_{\text{opt}}$ оптимальную последовательность завершения операций для $P$, тогда по лемме 1 имеем

$$L_{\max}(P1,\sigma_{\text{opt}}) \leqslant L_{\max}(P,\sigma_{\text{opt}}) = L^*_{\max}(P).$$

Ввиду того, что $L^*_{\max}(P1) \leqslant L_{\max}(P1,\sigma_{\text{opt}})$, получаем $L_{\max}(P) \leqslant 2L^*_{\max}(P) + \max\limits_{j\in J} d_j$. ■

## 2. Задача с учётом ресурсных ограничений

Перейдём к задаче составления расписания с ограничением на общее потребление невозобновимого ресурса — энергии. Для каждой операции $j \in J$ задан объём $W_j$, на основе которого вычисляются длительности $p_j$. Если процессор работает со скоростью $s_j$ при выполнении операции $j$, то мгновенная мощность (потребление энергии в единицу времени) равна $s_j^\alpha$, где $\alpha > 1$ — константа. Например, $\alpha = 1{,}11$ для Intel PXA270, $\alpha = 1{,}66$ для TCP offload engine, $\alpha = 3$ для CMOS устройств. Длительность операции $j$ в таком случае равна $p_j = W_j/s_j = \left(W_j^\alpha/E_j\right)^{\alpha-1}$, а потребление энергии $E_j = (W_j/s_j)\, s_j^\alpha$. Общий доступный объём ресурса обозначим через $E$. Данная задача обозначается как $P|\text{size}_j, \text{energy}|L_{\max}$ в случае произвольного числа процессоров и $1|\text{energy}|L_{\max}$ — в случае одного процессора [1, 9].

Такого рода задачи с выпуклой функцией потребления ресурсов также возникают и в промышленном производстве [10]. Здесь длительности работ зависят от объёма потребляемого ресурса, а именно: длительность работы $j \in J$ равна $p_j(R_j) = (W_j/R_j)^\kappa$, где $\kappa \leqslant 1$ — константа; $W_j$ — объём работы; $R_j$ — объём потребляемого ресурса.

Случай $\kappa = 1$ соответствует многим реальным государственным и промышленным операциям, а случай $\kappa = 0{,}5$ — проектированию схем СБИС, где для отдельной задачи произведение площади кремния (ресурса) на квадрат затраченного времени равно константе (рабочей нагрузке).

### 2.1. Один процессор

Для задачи с одним процессором и заданным порядком следования операций $\pi = (\pi_1, \ldots, \pi_n)$ предлагается модель выпуклого программирования, которая разрешима за полиномиальное время с помощью метода эллипсоидов [14]:

$$L_{\max} \longrightarrow \min; \tag{1}$$

$$\sum_{i=1}^{j} p_{\pi_i} - d_{\pi_j} \leqslant L_{\max}, \quad j = 1, \ldots, n; \tag{2}$$

$$\sum_{j=1}^{n} W_j^\alpha \cdot p_{\pi_j}^{1-\alpha} = E; \tag{3}$$

$$L_{\max} \geqslant 0, \quad p_{\pi_j} \geqslant 0, \quad j = 1, \ldots, n. \tag{4}$$

Ограничение (2) соответствует вычислению временных смещений для операций, условие (3) ограничивает общее потребление ресурса.

**Алгоритм** A1E для задачи $1|\text{energy}|L_{\max}$:

1) Построить перестановку $\pi^{d\leqslant}$, соответствующую упорядочению операций по неубыванию директивных сроков.
2) Решить задачу (1)–(4) при перестановке $\pi^{d\leqslant}$.
3) Построить расписание по перестановке $\pi^{d\leqslant}$ с длительностями, найденными по модели (1)–(4).

**Теорема 2.** Алгоритм A1E позволяет найти оптимальное решение для задачи $1|\text{energy}|L_{\max}$ за полиномиальное время.

***Доказательство.*** Чтобы решение задачи (1)–(4) было оптимальным для случая одного процессора, операции должны идти по неубыванию их директивных сроков согласно перестановке $\pi^{d\leqslant}$. Допустим, что это не так. Поменяем местами две любые подряд идущие операции, будем считать, что операции имеют номера $i$ и $i+1$ и директивные сроки $d_i < d_{i+1}$. Их моменты окончания изменятся: для операции $i$ временное смещение станет равным $c_i + p_{i+1} - d_i = c_{i+1} - d_i$, а для операции $i+1$ оно будет равно $c_{i+1} - p_i - d_{i+1}$. Таким образом, $c_{i+1} - d_i > c_i - d_i$, $c_{i+1} - d_i > c_{i+1} - d_{i+1}$ и $c_{i+1} - p_i - d_{i+1} < c_{i+1} - d_{i+1}$. Значит, после перестановки максимум из двух временных смещений достигается на $c_{i+1} - d_i$, и эта величина больше, чем временные смещения до обмена местами. ■

### 2.2. Два процессора

Докажем NP-трудность задачи на двух процессорах с распараллеливанием операций в случае длительностей работ, зависящих от потребления ресурсов, и опишем полиномиальный приближённый алгоритм с гарантированной оценкой точности.

**Теорема 3.** Задача $P2|\text{size}_j, \text{energy}|L_{\max}$ является NP-трудной в сильном смысле.

***Доказательство.*** Используем подход, предложенный в [7]. Рассмотрим классическую задачу 3-Разбиение [15], где имеется $3k$ элементов с суммарным весом $\sum\limits_{i=1}^{3k} e_i = kB$. Вопрос заключается в том, существует ли разбиение на $k$ подмножеств $A_1, \ldots, A_k$, где $\sum\limits_{i \in A_1} e_i = \ldots = \sum\limits_{i \in A_k} e_i = B$. В случае положительного ответа каждое подмножество содержит 3 элемента.

Построим пример задачи $P2|\text{size}_j, \text{energy}|L_{\max}$. Пусть число работ $n = 6k$, а объёмы, число используемых процессоров и директивные сроки определены следующим образом:

$$W_j = e_j,\ d_j = 4kB,\ \text{size}_j = 1,\ j = 1, \ldots, 3k,$$
$$W_j = 2B,\ d_j = (4(j-3k)-2)B,\ \text{size}_j = 1,\ j = 3k+1, \ldots, 4k,$$
$$W_j = 3B,\ d_j = (4(j-4k)-1)B,\ \text{size}_j = 1,\ j = 4k+1, \ldots, 5k,$$
$$W_j = B,\ d_j = 4(j-5k)B,\ \text{size}_j = 2,\ j = 5k+1, \ldots, 6k.$$

Общий объём энергии положим равным $E = 8kB$. Требуется ответить на вопрос, существует ли расписание с $L_{\max} \leqslant 0$, то есть когда все работы выполняются до их директивных сроков.

Заметим, что суммарный объем всех работ равен $8kB$, а максимальный директивный срок равен $4kB$ при наличии двух процессоров. Значит, в любом допустимом расписании нет простоев, следовательно, работы могут выполняться только с единичной скоростью (ввиду выпуклости функции расчёта мгновенной мощности). В таком случае допустимое расписание имеет структуру, представленную на рис. 2, и существует тогда и только тогда, когда в задаче 3-Разбиение ответ «да». ■


Рис. 2. Иллюстрация к доказательству NP-трудности (структура допустимого расписания)

Для построения приближённого решения используется подход из п. 1.1 и алгоритм для однопроцессорной задачи из п. 2.1: сначала реализуется переход к задаче на одном процессоре путём уменьшения объёмов, далее решается задача выпуклого программирования и осуществляется возврат к задаче на двух процессорах путём увеличения длительностей.

**Алгоритм** A2-1-2E для задачи $P2|\text{size}_j, \text{energy}|L_{\max}$:

1) Задача $PE$ на двух процессорах сводится к задаче $PE1$ на одном процессоре так, что директивные сроки не меняются, а объёмы пересчитываются для каждой операции $j \in J$ по следующему правилу:
   - если $\text{size}_j = 1$, то $W'_j = W_j/2$;
   - если $\text{size}_j = 2$, то $W'_j = W_j$.

   Объём энергии уменьшается в 2 раза: $E' = E/2$.
2) Строится оптимальное решение для задачи $PE1$ на одном процессоре с помощью алгоритма A1E, где операции упорядочиваются по неубыванию директивных сроков согласно перестановке $\pi^{d\leqslant}$.

3) Вычисляются длительности операций для задачи $PE$:
   — если $\text{size}_j = 1$, то $p_j = 2p'_j$;
   — если $\text{size}_j = 2$, то $p_j = p'_j$.
4) Для задачи $PE$ допустимое решение строится по следующему правилу: на шаге $i = 1, \ldots, n$ операция $\pi_i^{d\leqslant}$ назначается в расписание так, что если $\text{size}_{\pi_i^{d\leqslant}} = 1$, то операция ставится на процессор с минимальным временем завершения текущего набора операций, а если $\text{size}_{\pi_i^{d\leqslant}} = 2$, то операция ставится на два процессора с момента времени, когда они оба доступны.

**Теорема 4.** Если $L_{\max}(PE)$ — значение целевой функции расписания, построенного для задачи $PE$ алгоритмом A2-1-2E, а $L^*_{\max}(PE)$ — оптимальное значение целевой функции для задачи $PE$, то $L_{\max}(PE) \leqslant 2L^*_{\max}(PE) + \max\limits_{j\in J} d_j$.

***Доказательство.*** Для алгоритма A2-1-2E сохраняются оценки точности, доказанные в теореме 1 для алгоритма A2-1-2 и задачи без учёта расхода энергии. Здесь принимается во внимание тот факт, что алгоритм A1E для однопроцессорной задачи позволяет получить её оптимальное решение за полиномиальное время, а оптимальное временное смещение для задачи PE1 является нижней оценкой на целевую функцию для задачи с двумя процессорами. Нижняя оценка гарантируется за счёт того, что уменьшение объёмов операций и общего бюджета энергии в 2 раза и переход к однопроцессорной задаче равносильны релаксации исходной задачи до варианта, когда все работы могут распараллеливаться на два процессора с уменьшением длительностей в 2 раза. ■

## 3. Вычислительный эксперимент

Для экспериментального исследования алгоритмов A2-1-2 и A2-1-2E были случайным образом сгенерированы примеры с равномерным распределением. Для задачи построено 25 серий. В каждой серии генерируется 50 задач с 50 операциями с заданными параметрами. Операции рассматривались двух типов — малой (из интервала $[10, 20]$) и большой (из интервала $[100, 120]$) длительности. Тип операции задаётся так, что с вероятностью $P_{\text{small}}$ операция имеет малую длительность, а с вероятностью $(1 - P_{\text{small}})$ — большую. Число используемых процессоров для каждой операции задаётся по правилу: один процессор выбирается с вероятностью $P_{\text{proc}}$ и два процессора — с вероятностью $(1 - P_{\text{proc}})$. Вероятности выбираются из множества $\{0,\ 0{,}25,\ 0{,}5,\ 0{,}75,\ 1\}$. Директивные сроки задаются в зависимости от общего объёма операций: для группы из первых 25 серий директивные сроки задаются от 25 до 50 % объёма (суммы всех длительностей работ, деленной на два), для второй группы из следующих 25 серий — от 45 до 90 % от объёма.

Для анализа результатов эксперимента вычисляются среднее относительное отклонение от нижней границы и стандартное отклонение. В табл. 1 представлены значения среднего относительного отклонения от нижней границы для первой группы (с округлением до сотых).

Значения среднего относительного отклонения увеличиваются с ростом доли операций малой длительности. Среднее относительное отклонение уменьшается с увеличением доли двухпроцессорных операций. Отметим, что значения не превышают 15 %.

Поскольку значения целевой функции зависят от правила генерации директивных сроков, целесообразно провести исследование для серий задач с большими, относительно общего объёма операций, директивными сроками. В табл. 2 представлены значения

Таблица 1

**Среднее относительное отклонение в % результата алгоритма A2-1-2 от нижней границы для первой группы примеров (в скобках указано стандартное отклонение)**

| Доля операций малой длительности | Доля двухпроцессорных операций | | | | | Среднее |
|---|---|---|---|---|---|---|
| | 0 | 0,25 | 0,5 | 0,75 | 1 | |
| 0 | 1,34 (1,21) | 6,67 (3,30) | 5,50 (2,17) | 2,98 (2,42) | 0 | 3,30 (1,82) |
| 0,25 | 2,75 (1,23) | 7,36 (3,36) | 6,54 (2,15) | 5,27 (2,12) | 0 | 4,38 (1,77) |
| 0,5 | 3,71 (2,17) | 8,37 (2,56) | 6,90 (3,87) | 6,74 (4,12) | 0 | 5,14 (2,54) |
| 0,75 | 4,78 (3,37) | 9,54 (4,79) | 8,18 (4,33) | 7,05 (4,25) | 0 | 5,91 (3,35) |
| 1 | 1,98 (1,24) | 7,93 (2,35) | 8,18 (2,45) | 4,66 (1,98) | 0 | 4,50 (1,60) |
| Среднее | 2,91 (1,46) | 7,97 (2,89) | 7,02 (2,66) | 5,34 (2,66) | 0 | 4,65 (2,21) |

среднего относительного отклонения от нижней границы для второй группы (с округлением до сотых).

Таблица 2

**Среднее относительное отклонение в % результата алгоритма A2-1-2 от нижней границы для второй группы примеров (в скобках указано стандартное отклонение)**

| Доля операций малой длительности | Доля двухпроцессорных операций | | | | | Среднее |
|---|---|---|---|---|---|---|
| | 0 | 0,25 | 0,5 | 0,75 | 1 | |
| 0 | 8,27 (4,29) | 34,95 (5,42) | 23,67 (3,49) | 15,35 (9,37) | 0 | 16,44 (4,51) |
| 0,25 | 13,82 (6,43) | 31,37 (4,25) | 29,36 (3,42) | 24,21 (8,47) | 0 | 19,75 (4,51) |
| 0,50 | 17,11 (8,15) | 38,22 (8,37) | 34,44 (7,99) | 26,20 (4,35) | 0 | 23,19 (5,77) |
| 0,75 | 32,07 (7,41) | 42,56 (5,22) | 46,17 (6,24) | 35,63 (8,39) | 0 | 31,28 (5,45) |
| 1 | 11,68 (5,37) | 37,47 (7,28) | 34,08 (10,11) | 23,35 (9,87) | 0 | 21,31 (6,52) |
| Среднее | 16,59 (6,11) | 36,91 (6,25) | 33,54 (6,25) | 24,94 (8,09) | 0 | 22,40 (5,35) |

Заметим, что для второй группы значения среднего относительного отклонения при генерации двухпроцессорных операций с долями 0,25 и 0,5 отличаются несущественно, кроме случая, когда все операции большой длительности. В остальном сохраняются тенденции, наблюдаемые для первой группы; значения не превышают 55 %.

По результатам эксперимента для обеих групп можно заметить, что среднее относительное отклонение увеличивается с увеличением директивных сроков. С уменьшением доли двухпроцессорных операций увеличивается среднее относительное отклонение от нижней границы, за исключением крайних случаев: когда все операции двухпроцессорные (нижняя граница даёт оптимальное решение) и когда все операции однопроцессорные, что даёт меньшее значение среднего относительного отклонения, чем случаи с наличием двухпроцессорных операций.

Как видно из результатов вычислительного эксперимента, на примерах, сгенерированных случайным образом с равномерным распределением, среднее относительное отклонение от нижней границы существенно меньше, чем получено для алгоритма в худшем случае. Отметим необходимость дальнейшего теоретического и экспериментального исследования отклонения нижней границы от оптимального решения для различных классов примеров.

Результаты для алгоритма A2-1-2E идентичны и в обобщённом виде представлены в табл. 3. Для решения выпуклой программы, соответствующей подзадаче с од-

ним процессором, использовался коммерческий пакет Baron (сервер NEOS — `https://neos-server.org/neos/`). Общий объём энергии полагался равным $E = 1000$.

Таблица 3

**Среднее относительное отклонение в % результата алгоритма A2-1-2E от нижней границы (в скобках указано стандартное отклонение)**

| Доля операций | 0 | 0,25 | 0,5 | 0,75 | 1 |
|---|---|---|---|---|---|
| | Первая группа примеров | | | | |
| Двухпроцессорных | 5,17 (2,3) | 8,25 (3,5) | 7,13 (3,3) | 6,18 (3,2) | 0 |
| Малой длительности | 5,23 (1,3) | 5,27 (1,1) | 7,43 (3,3) | 7,45 (2,1) | 5,32 (2,2) |
| | Вторая группа примеров | | | | |
| Двухпроцессорных | 18,25 (5,1) | 36,35 (7,3) | 33,11 (5,1) | 24,35 (6,1) | 0,09 |
| Малой длительности | 19,42 (6,4) | 22,25 (7,3) | 25,45 (8,4) | 32,40 (8,3) | 21,13 (7,2) |

## Заключение

Проведено исследование задачи составления расписаний на двух процессорах с возможностью распараллеливания и учётом ограничения на потребление энергии. Доказана NP-трудность и предложен полиномиальный приближённый алгоритм с гарантированной оценкой точности в худшем случае. Для случая одного процессора предложена модель выпуклого программирования, позволяющая получать оптимальное решение за полиномиальное время. Результаты экспериментального исследования показали перспективность используемого подхода. В дальнейшем представляет интерес анализ генерической сложности рассматриваемых задач и обобщение результатов на случай произвольного числа процессоров, например на основе декомпозиции.

## ЛИТЕРАТУРА


1. *Drozdowski M.* Scheduling for Parallel Processing. Dordrecht: Springer, 2009. 386 p.
2. *Сервах В. В.* Эффективно-разрешимый случай задачи календарного планирования с возобновимыми ресурсами // Дискретн. анализ и исслед. опер. 2000. Сер. 2. Т. 7. № 1. С. 75–82.
3. *Коваленко Ю. В.* О задаче календарного планирования с возобновимым ресурсом // Автомат. и телемех. 2012. № 6. С. 140–153.
4. *Goncharov E. N.* An improved genetic algorithm for the resource-constrained project scheduling problem // N. Olenev, Yu. Evtushenko, M. Jaćimović, et al. (eds). Advances in Optimization and Applications. Springer, Cham, 2022. P. 35–47.
5. *Zhuravlev S., Blagodurov S., and Fedorova A.* Addressing shared resource contention in multicore processors via scheduling // Comput. Archit. News. 2010. V. 38. No. 1. P. 129–142.
6. *Еремеев А. В., Сахно М. Ю.* Построение расписания для многоядерного процессора с учетом взаимного влияния работ // Выч. мет. программирование. 2023. Т. 24. № 1. С. 115–126.
7. *Kononov A. and Kovalenko Y.* On speed scaling scheduling of parallel jobs with preemption // LNCS. 2016. V. 9869. P. 309–321.
8. *Kononov A. and Zakharova Y.* Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion // Intern. J. Artif. Intell. 2023. V. 21. No. 2. P. 109–129.
9. *Gerards M. E. T., Hurink J. L., and Holzenspies P. K. F.* A survey of offline algorithms for energy minimization under deadline constraints // J. Scheduling. 2016. V. 19. P. 3–19.
10. *Shabtay D. and Kaspi M.* Parallel machine scheduling with a convex resource consumption function // Europ. J. Oper. Res. 2006. V. 173. No. 1. P. 92–107.

11. https://www2.informatik.uni-osnabrueck.de/knust/class/ — Complexity Results for Scheduling Problems. 2024.
12. *Lee C.-Y. and Cai X.* Scheduling one and two-processor tasks on two parallel processors // IIE Trans. 1999. V. 31. P. 445–455.
13. *Lenstra J. K., Rinnooy Kan A. H. G., and Brucker P.* Complexity of machine scheduling problems // Ann. Discrete Math. 1977. V. 1. P. 343–362.
14. *Grotschel M., Lovasz L., and Schrijver A.* Geometric Algorithms and Combinatorial Optimizations. Heidelberg: Springer, 2009. 332 p.
15. *Garey M. and Johnson D.* Computers and Intractability. A Guide to the Theory of NP-completeness. N.Y.: W. H. Freeman and Company, 1979. 338 p.

REFERENCES

1. *Drozdowski M.* Scheduling for Parallel Processing. Dordrecht, Springer, 2009. 386 p.
2. *Servakh V. V.* Effektivno-razreshimyy sluchay zadachi kalendarnogo planirovaniya s vozobnovimymi resursami [An efficiently solvable case of the scheduling problem with renewable resources]. Diskretn. Analiz i Issled. Oper., 2000, ser. 2, vol. 7, no. 1, pp. 75–82. (in Russian)
3. *Kovalenko Yu. V.* On the calendar planning problem with renewable resource. Autom. Remote Control, 2012, vol. 73, no. 6, pp. 1046–1055.
4. *Goncharov E. N.* An improved genetic algorithm for the resource-constrained project scheduling problem. N. Olenev, Yu. Evtushenko, M. Jaćimović, et al. (eds). Advances in Optimization and Applications, Springer, Cham, 2022, pp. 35–47.
5. *Zhuravlev S., Blagodurov S., and Fedorova A.* Addressing shared resource contention in multicore processors via scheduling. Comput. Archit. News, 2010, vol. 38, no. 1, pp. 129–142.
6. *Eremeev A. V. and Sakhno M. Yu.* Postroenie raspisaniya dlya mnogoyadernogo protsessora s uchetom vzaimnogo vliyaniya rabot [Multi-core processor scheduling with respect to the mutual influence of jobs]. Vych. Met. Programmirovanie, 2023, vol. 24, no. 1, pp. 115–126. (in Russian)
7. *Kononov A. and Kovalenko Y.* On speed scaling scheduling of parallel jobs with preemption. LNCS, 2016, vol. 9869, pp. 309–321.
8. *Kononov A. and Zakharova Y.* Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion. Intern. J. Artif. Intell., 2023, vol. 21, no. 2, pp. 109–129.
9. *Gerards M. E. T., Hurink J. L., and Holzenspies P. K. F.* A survey of offline algorithms for energy minimization under deadline constraints. J. Scheduling, 2016, vol. 19, pp. 3–19.
10. *Shabtay D. and Kaspi M.* Parallel machine scheduling with a convex resource consumption function. Europ. J. Oper. Res., 2006, vol. 173, no. 1, pp. 92–107.
11. https://www2.informatik.uni-osnabrueck.de/knust/class/ — Complexity Results for Scheduling Problems, 2024.
12. *Lee C.-Y. and Cai X.* Scheduling one and two-processor tasks on two parallel processors. IIE Trans., 1999, vol. 31, pp. 445–455.
13. *Lenstra J. K., Rinnooy Kan A. H. G., and Brucker P.* Complexity of machine scheduling problems. Ann. Discrete Math., 1977, vol. 1, pp. 343–362.
14. *Grotschel M., Lovasz L., and Schrijver A.* Geometric Algorithms and Combinatorial Optimizations. Heidelberg, Springer, 2009. 332 p.
15. *Garey M. and Johnson D.* Computers and Intractability. A Guide to the Theory of NP-completeness. N.Y., W. H. Freeman and Company, 1979. 338 p.

# СВЕДЕНИЯ ОБ АВТОРАХ

**АБРОСИМОВ Михаил Борисович** — доктор физико-математических наук, профессор, заведующий кафедрой теоретических основ компьютерной безопасности и криптографии Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского, г. Саратов. E-mail: **mic@rambler.ru**

**ЕВТИНА Анастасия Олеговна** — студентка Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **aevtina55@gmail.com**

**ЗАХАРОВА Юлия Викторовна** — кандидат физико-математических наук, старший научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **yzakharova@ofim.oscsbras.ru**

**КАЗАКОВ Илья Борисович** — кандидат физико-математических наук, старший преподаватель кафедры информатики РЭУ им. Плеханова; доцент кафедры высшей математики МГТУ им. Баумана, г. Москва. E-mail: **i_b_kazakov@mail.ru**

**КОСОЛАПОВ Юрий Владимирович** — кандидат технических наук, доцент кафедры алгебры и дискретной математики Южного федерального университета, г. Ростов-на-Дону. E-mail: **itaim@mail.ru**

**ЛЕЕВИК Антон Георгиевич** — криптограф-исследователь ООО «КуАпп», г. Москва; аспирант университета ИТМО, г. Санкт-Петербург.
E-mail: **anton.leevik@gmail.com**

**ЛЕЛЮК Евгений Андреевич** — соискатель Южного федерального университета, г. Ростов-на-Дону. E-mail: **lelukevgeniy@mail.ru**

**ЛОПАТИН Артем Анатольевич** — доктор физико-математических наук, доцент государственного университета города Кампинас (UNICAMP), г. Кампинас, Сан-Паулу. E-mail: **dr.artem.lopatin@gmail.com**

**ЛОСЬ Илья Викторович** — аспирант Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского, г. Саратов.
E-mail: **los.ilia.ru@gmail.com**

**МАЛЫГИНА Екатерина Сергеевна** — кандидат физико-математических наук, криптограф-исследователь ООО «КуАпп», г. Москва; научный сотрудник Международного научно-образовательного математического центра НГУ, г. Новосибирск.
E-mail: **emalygina@qapp.tech**

**МЕЛЬНИЧУК Евгений Михайлович** — криптограф-исследователь ООО «КуАпп», г. Москва. E-mail: **emelnichuk@qapp.tech**

**НАБОКОВ Денис Алексеевич** — криптограф-исследователь ООО «КуАпп», г. Москва. E-mail: **dnabokov@qapp.tech**

**РЫБАЛОВ Александр Николаевич** — кандидат физико-математических наук, старший научный сотрудник лаборатории комбинаторных и вычислительных методов алгебры и логики Института математики им. С. Л. Соболева СО РАН, г. Омск.
E-mail: **alexander.rybalov@gmail.com**

**ЧУХНО Андрей Борисович** — старший преподаватель кафедры компьютерной безопасности МИЭМ НИУ ВШЭ, г. Москва. E-mail: **achuhno@hse.ru**

Журнал «Прикладная дискретная математика» входит в перечень ВАК рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата и доктора наук по специальностям 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (технические науки), 2.3.6. «Методы и системы защиты информации, информационная безопасность» (физико-математические и технические науки), 1.1.5. «Математическая логика, алгебра, теория чисел и дискретная математика» (физико-математические науки), 1.2.3. «Теоретическая информатика, кибернетика» (физико-математические науки), а также в перечень журналов, рекомендованных ФУМО ВО ИБ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал индексируется в базах данных Web of Science (Emerging Sources Citation Index (ESCI) и Russian Science Citation Index (RSCI)), Scopus, MathSciNet и Zentralblatt MATH. По решению ВАК от 21.12.2023 он отнесён к первой категории (К1) научных журналов, входящих в Перечень ВАК.

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте journals.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также правила подготовки рукописей статей для публикации в журнале.

**Тематика публикаций журнала:**

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*