МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

# ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

*Научный журнал*

2020 № 50

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:** vestnik_pdm@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*
Верстка *И. А. Панкратовой*

Подписано к печати 09.12.2020. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 14,8. Тираж 300 экз.
Заказ № 4516. Цена свободная. Дата выхода в свет 18.12.2020.

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

# СОДЕРЖАНИЕ

## МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ



## ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ



## ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ



## МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

# CONTENTS

## MATHEMATICAL METHODS OF CRYPTOGRAPHY



## APPLIED CODING THEORY



## APPLIED GRAPH THEORY



## MATHEMATICAL BACKGROUNDS OF INFORMATICS AND PROGRAMMING

# МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 519.7

## О БЕЗОПАСНОСТИ ПРОТОКОЛА SESPAKE

Е. К. Алексеев, С. В. Смышляев

*ООО «КРИПТО-ПРО», г. Москва, Россия*

Протокол SESPAKE описан в Рекомендациях по стандартизации Р 50.1.115-2016 «Информационная технология. Криптографическая защита информации. Протокол выработки общего ключа с аутентификацией на основе пароля». В настоящей работе приводятся результаты криптографического анализа данного протокола в релевантных моделях противника. Вводится новая модель противника с угрозой ложной аутентификации, являющаяся расширением базовой модели, применяемой для протоколов с внутренней диверсификацией ключей. Стойкость протокола оценивается в двух моделях: в модели на основе задачи отличения и в расширенной.

**Ключевые слова:** *модели и методы защиты информации, криптографические протоколы.*

DOI 10.17223/20710410/50/1

## ON SECURITY OF THE SESPAKE PROTOCOL

E. K. Alekseev, S. V. Smyshlyaev

*CryptoPro, Moscow, Russia*

**E-mail:** alekseev@cryptopro.ru, svs@cryptopro.ru

The Security Evaluated Standardized Password Authenticated Key Exchange (SESPAKE) protocol is standardized in Russia as R 50.1.115-2016. The current paper provides analysis of the protocol in relevant adversary models. We define new indistinguishability-based adversary model with a threat of false authentication that is an extension of the original indistinguishability-based model up to the case of protocols with authentication step without key diversification. We prove the protocol security in two adversary models with a classic threat of distinguishing a generated session key from a random string and with a threat of false authentication.

**Keywords:** *models and methods in information security, cryptographic protocols.*

### Введение

Суть протоколов, которым посвящена данная работа, состоит в том, что стороны, изначально разделяя лишь малоэнтропийный ключ (который обычно называют паролем), в процессе взаимодействия вырабатывают высокоэнтропийный общий ключ. Специфичное требование, предъявляемое к таким протоколам, таково: даже активный противник, имеющий доступ к каналу связи, не должен получить информацию,

достаточную для подбора пароля «off-line», то есть без взаимодействия с легитимными участниками. Являясь по сути протоколами выработки общего ключа с аутентификацией на основе пароля, протоколы такого типа часто объединяют под общим названием «PAKE» — «Password Authenticated Key Exchange».

Первый протокол, получивший название EKE, предложен S. Bellovin и M. Merritt в 1992 г. [1]. На сегодняшний день существует множество протоколов данного типа (см., например, [2 – 6]). В 1993 г. M. Bellare и P. Rogaway в работе [7] предложили *модель на основе неотличимости* для оценки стойкости протоколов аутентифицированной выработки общего ключа. Эта модель была расширена в [8] для анализа PAKE-протоколов (обычно её называют BPR-моделью). Она позволила получить оценки стойкости для некоторых протоколов данного типа [9].

Структурно протоколы типа AKE (и PAKE в частности) обязательно содержат этап выработки ключа и опционально — этап его подтверждения. Нижние оценки стойкости получены как для протоколов без этапа подтверждения [9], так и для ряда протоколов, содержащих этот этап [4]. В обоих случаях доказывается стойкость относительно угрозы отличения выработанного общего ключа от случайной строки. Однако не для всех протоколов, содержащих этап подтверждения ключа, удаётся корректно определить такую угрозу. В случаях, когда на этапе подтверждения доказывается владение непосредственно выработанным ключом, у противника естественным образом появляется критерий, позволяющий проверить, является ли целевая строка случайной. На это явно указывают J. Bender, M. Fischlin и D. Kugler в [10] (п. «The final authentication step» в разделе с описанием протокола PACE), такая же проблема возникает при оценке стойкости протокола TLS 1.2 (см. аннотацию работы [11]). В работах, в которых получены оценки стойкости для протоколов с подтверждением ключа, проблема корректного определения угрозы отличения решается на уровне строения протокола путём введения дополнительного «ветвления» полученных ключей. Это означает, что ключ для этапа подтверждения производится из общего секрета одним необратимым образом, а основной ключ производится из этого же секрета другим необратимым образом.

В настоящей работе исследуется протокол SESPAKE аутентифицированной выработки общего ключа на основе пароля, содержащий этап подтверждения ключа. При этом на этапе подтверждения используется непосредственно сам выработанный в процессе взаимодействия сторон секрет (то есть отсутствует упомянутое ранее «ветвление»). Этап выработки ключа является модификацией протокола, предложенного в [9], этап подтверждения ключа является оригинальным. Для данного протокола предложена модель на основе неотличимости с угрозой ложной аутентификации, которая формализуется с помощью задачи отличения настоящей аутентификационной информации от случайной. Доказательство оценки стойкости протокола SESPAKE в этой модели делится на три этапа, первый из которых существенно основывается на идеях, представленных в работе [9]. Второй и третий этапы являются оригинальными. На втором и третьем этапах доказательства использованы приёмы, которые могут помочь при анализе и обосновании стойкости других протоколов типа PAKE. Насколько известно авторам, представленный протокол является первым протоколом без ветвления ключей на этапе подтверждения, для которого предложена модель и в этой модели получена оценка стойкости.

Обзор уязвимостей протоколов данного семейства, а также принципы построения протокола SESPAKE, стандартизированного в России [12], представлены в работе [13].

## 1. Модель противника и оценки стойкости известных протоколов

Опишем модель BPR, которая используется в настоящей работе для оценки стойкости этапа выработки ключа протокола SESPAKE, и сделаем краткий обзор известных результатов, касающихся оценок стойкости PAKE-протоколов. Рассмотрим также общие принципы моделирования противника, применяемые в области практически ориентированной доказуемой стойкости.

В [14] выделены следующие типы моделей противника: на основе неотличимости, на основе универсальной композиции и на основе симулирования. В настоящей работе стойкость протокола исследована в модели на основе неотличимости; рассмотрим эту модель противника подробнее.

Все результаты относительно стойкости протоколов типа PAKE, известные авторам, относятся к области так называемой практически ориентированной доказуемой стойкости, принципы которой описаны в [15]. Особенность этого подхода состоит в том, что он позволяет анализировать стойкость конкретных схем, не обладающих бесконечно возрастающим параметром безопасности. Данный подход использует классическую для теории сложности технику сведения одной задачи к другой, а результатом проведённого анализа является верхняя оценка возможностей (например, вероятности успешного осуществления угрозы) противника, зависящая от доступных ему вычислительных и информационных ресурсов (подробнее процесс получения оценок описан, например, в [16, разд. 8.3.1]). Ограничения этих ресурсов, продиктованные конкретными условиями эксплуатации оцениваемой криптосистемы, позволяют получить численные оценки её стойкости. Таким образом, вопросы выбора целевого порогового значения стойкости и, как следствие, допустимости использования криптосистемы могут быть решены исключительно на этапе её встраивания в более высокоуровневую систему. Вне такого контекста получаемые оценки можно прокомментировать с точки зрения предполагаемых границ применения протокола.

### 1.1. Общие принципы формализации противника

Под *противником* будем понимать вероятностную интерактивную машину Тьюринга. Неформально это программа, которая может делать запросы к другим программам и отвечать на запросы, сделанные к ней самой. Формальное определение с подробными пояснениями можно найти в [17, Definitions 4.2.1, 4.2.2].

Мера успешности противника при решении задач, определяемых моделью, называется *преимуществом противника* и обозначается $\mathbf{Adv}(\mathcal{A})$. Обычно это обозначение снабжается верхним и нижним индексами, которые указывают соответственно на используемую модель и конкретный анализируемый в её рамках криптографический объект.

Задачи противника, которые подразумевают бинарный ответ, будем называть *задачами распознавания*. В таких задачах $\mathrm{SUCC}$ — это событие, состоящее в том, что результат работы противника совпал со значением неизвестного ему бита $b$, выбранного случайно в процессе его модельного взаимодействия с анализируемой криптосистемой. Для противника $\mathcal{A}$, решающего задачу распознавания $Task$, принято оценивать не вероятность успеха, а величину $\mathbf{Adv}^{Task}(\mathcal{A})$, которая определяется следующим образом:

$$\mathbf{Adv}^{Task}(\mathcal{A}) = 2\Pr[\mathrm{SUCC}] - 1.$$

Отсутствие в данном определении модуля объясняется тем, что для любого противника с отрицательным значением $\mathbf{Adv}^{Task}(\mathcal{A})$ существует противник $\mathcal{A}'$, который работает точно так же, как $\mathcal{A}$, но возвращает противоположные ему значения.

Если задача $Task$ вычислительная (результатом работы является значение из достаточно большого множества), то через $\mathbf{Adv}^{Task}(\mathcal{A})$ обозначается вероятность вычисления противником искомого значения (например, ключа криптосистемы).

Вычислительными ресурсами противника называют сумму максимального времени его работы и размера записи кода его программы. В зависимости от решаемой задачи противник может иметь возможность осуществлять обращения к одному или нескольким оракулам. Через $\mathcal{A}(t, q_1, q_2, \ldots, q_k)$ будем обозначать множество противников, которые обладают вычислительными ресурсами, не превышающими $t$, и делают не более $q_1, q_2, \ldots, q_k$ запросов к оракулам, доступным им в рамках используемой модели. Через $\mathbf{Insec}^{Task}(t, q_1, q_2, \ldots, q_k)$ будем обозначать следующую величину, называемую для краткости *нестойкостью задачи* $Task$:

$$\mathbf{Insec}^{Task}(t, q_1, q_2, \ldots, q_k) = \max_{\mathcal{A} \in \mathcal{A}(t, q_1, q_2, \ldots, q_k)} \mathbf{Adv}^{Task}(\mathcal{A}).$$

В том случае, когда требуется рассмотреть множество противников, ограниченное не по всем параметрам (или какие-то ограничения не имеют смысла для рассматриваемой задачи), лишние параметры не указываются.

Подробнее этот подход к формализации задачи оценки стойкости криптосистем обсуждается в работе [18].

### 1.2. Модель BPR

Опишем модель BPR для протоколов, предполагающих взаимодействие двух сторон (иногда мы также будем называть их участниками). Процесс реализации действий, предписываемых протоколом, будем называть сессией. В рамках различных сессий может взаимодействовать несколько сторон, совокупность которых будем называть сетью.

Традиционно для этого подхода в криптографии возможности противника по взаимодействию с криптосистемой в BPR-модели моделируются с помощью предоставления ему доступа к ряду *оракулов*, являющихся интерактивными машинами Тьюринга. Оракулы в BPR-модели можно разделить на два типа:

— Оракулы, моделирующие участников сети (далее — оракулы-участники): они носят технический характер и выделяются больше для ясности изложения и прозрачности проецирования модели на практику. Противник напрямую не может делать запросы к этим оракулам.

— Оракулы, моделирующие различные возможности противника: к этим оракулам у противника есть непосредственный доступ. При этом они задействуют оракулов из предыдущего пункта.

Если $\mathcal{O}$ — некоторый оракул, принимающий на вход запросы из множества $R$, то через $\mathcal{O}(r)$, $r \in R$, будем обозначать значение, возвращаемое оракулом $\mathcal{O}$ в качестве ответа на запрос $r$.

Заметим, что использование понятия «оракул» для моделирования в математической криптографии является стандартным приёмом, позволяет не вводить лишних конструкций и не приводит к внутренним противоречиям.

**Оракулы-участники.** Прежде чем переходить к описанию оракулов первого типа, введём некоторые сопутствующие понятия.

Множество участников сети является объединением непустых и непересекающихся множеств Client и Server, состоящих из строк-идентификаторов клиентов и серверов соответственно. Клиент может взаимодействовать с одним сервером, для этого используется секретный параметр, порождённый независимо от секретных параметров

других участников сети. Сервер может взаимодействовать с несколькими клиентами и хранит список своих секретных параметров, необходимых для такого взаимодействия в соответствии с протоколом. Заметим, что секретные параметры клиента и сервера не обязательно равны друг другу. Если они связаны таким образом, что могут быть эффективно получены друг из друга, то говорят, что протокол является протоколом *симметричного типа*, иначе — протоколом *асимметричного типа*.

Оракулы первого типа моделируют работу участников сети. Потенциально количество таких оракулов в модели не ограничено. Причина этого в том, что каждый из участников может осуществлять взаимодействие в рамках многих сессий, поэтому каждому оракулу соответствует пара $(U, i)$, где $U$ — идентификатор участника сети; $i$ — номер сессии участника $U$, взаимодействие в которой моделируется этим оракулом. Часто можно встретить аналогию, что каждый такой оракул соответствует процессу операционной системы, реализующему протокольное взаимодействие некоторого пользователя. Оракулы, к которым не было запросов, находятся в одинаковом состоянии ожидания начала взаимодействия. Запросом к оракулу является строка, которую он интерпретирует как сообщение, пришедшее к нему из канала связи. Ответом оракула является строка, сформированная им в соответствии с протоколом и его текущим внутренним состоянием. Особым образом определяется первый запрос к оракулам, соответствующим клиентам сети, так как по протоколу клиент отправляет первое сообщение сам. В модели BPR первым запросом к клиенту должна быть строка-идентификатор того сервера, с которым этому клиенту следует начать взаимодействие. После последнего по протоколу запроса оракул переходит в состояние, в котором на любой запрос он возвращает один и тот же специальный символ (например, $\perp$), свидетельствующий о том, что сессия для него завершена.

Оракулы-участники имеют следующие специальные внутренние параметры, которые крайне важны для модели в целом:

— *sid*: идентификатор сессии. Перед началом взаимодействия этот параметр имеет специальное значение UNDEF, а после того, как оракул-участник успешно завершил взаимодействие по протоколу, становится равным конкатенации всех сообщений, переданных по каналу связи в рамках сессии;
— *pid*: идентификатор партнера. Перед началом взаимодействия этот параметр равен UNDEF. Он устанавливается равным строке-идентификатору того участника, с которым данный оракул-участник, по его мнению, осуществил успешное взаимодействие по протоколу и выработал общий ключ;
— *sk*: выработанный общий сеансовый ключ. Изначально также равен UNDEF и устанавливается равным ключу, выработанному в результате взаимодействия;
— *acc*: параметр, говорящий об успешности осуществленного оракулом взаимодействия. Изначально равен FALSE; устанавливается равным TRUE в тот момент, когда у оракула-участника становятся определены параметры *sid*, *pid* и *sk*;
— *term*: параметр, указывающий на то, что оракул завершил взаимодействие. Изначально равен FALSE.

**Пример 1.** Для иллюстрации рассмотрим в качестве $AKE$-протокола классический протокол Диффи — Хеллмана, в котором каждая из сторон снабжает посылаемое ей сообщение своим идентификатором. Рассмотрим последовательность запросов к оракулам-участникам и то, как меняются их внутренние состояния.

1) Запрос $(B)$ к оракулу $\mathcal{O}_A^1$. Оракул выберет число $x$, вычислит $X = g^x$ и вернёт в качестве ответа на запрос пару $(A, X)$. После этого он перейдёт в состояние ожидания ответа от сервера.
2) Запрос $(A, X)$ к оракулу $\mathcal{O}_B^1$. Оракул $\mathcal{O}_B^1$ выберет $y$, вычислит $Y = g^y$ и общий ключ $sk = X^y$. Также он определит остальные внутренние параметры: $sid = (A, X)||(B, Y)$, $pid = A$, $acc = \text{TRUE}$ и $term = \text{TRUE}$. В качестве ответа на запрос оракул вернёт $(B, Y)$.
3) Запрос $(B, Y)$ к оракулу $\mathcal{O}_A^1$. Оракул вычислит общий ключ $sk = Y^x$, а также другие параметры: $sid = (A, X)||(B, Y)$, $pid = B$, $acc = \text{TRUE}$ и $term = \text{TRUE}$. В качестве ответа оракул вернёт пустую строку, свидетельствующую об успешном завершении взаимодействия.

Если запросом к оракулу $\mathcal{O}_B^1$ была бы не строка $(A, X)$, а только идентификатор $A$, то оракул счёл бы это сообщение не соответствующим протоколу, вернул символ $\perp$, свидетельствующий об ошибке, и перешел в состояние завершённого взаимодействия. О том, что взаимодействие не завершилось успешно, свидетельствовали бы неустановленные (UNDEF) значения $sid, pid, sk$ и то, что $acc = \text{FALSE}$.

Нестойкость этого протокола в модели BPR показана ниже в примере 2.

**Оракулы, моделирующие возможности противника.** Неформально, модель BPR предполагает, что противник полностью контролирует каналы связи между участниками сети, т. е. читает все передаваемые по ним сообщения, может изменять их, отправлять по каналам сообщения, сформированные им самостоятельно, и т. п. Он может также навязывать любому участнику сети выполнение протокола с любым другим участником сети. Помимо этого, он может получать сеансовые ключи некоторых выбранных им сессий (причины наличия такой возможности обсуждаются при описании соответствующего оракула). Цель противника — получить какую-либо информацию о сеансовом ключе выбранной им, но не вскрытой ранее с помощью упоминавшегося специального запроса сессии.

В модели BPR противник не может осуществлять запросы непосредственно к оракулам-участникам. Вместо этого ему предоставляется доступ к четырём специальным оракулам $\mathcal{O}_{\text{send}}$, $\mathcal{O}_{\text{exec}}$, $\mathcal{O}_{\text{rev}}$ и $\mathcal{O}_{\text{test}}$. Первые два моделируют возможности противника по влиянию на каналы связи. Третий оракул даёт возможность противнику получать выработанные участниками сеансовые ключи. Эта возможность не является естественной с точки зрения практики, поэтому мы подробно рассмотрим её далее. Четвёртый оракул $\mathcal{O}_{\text{test}}$ формализует угрозу в модели BPR, т. е. цель действий противника. В данной работе не анализируются криптографические свойства протокола при вскрытии противником долговременного ключа, поэтому мы не включаем в описание оракул $\mathcal{O}_{\text{cor}}$.

Прежде чем переходить к подробному описанию каждого из перечисленных оракулов, необходимо отметить, что модель BPR может быть параллельной (concurrent) и непараллельной (non-concurrent). В первом случае для одного участника допускается существование двух и более параллельных сессий, во втором — нет.

Четыре оракула, к которым может обращаться противник, имеют возможность получать значения любых внутренних параметров $acc, sid, pid, sk$ и $term$ оракулов-участников:

1) $\mathcal{O}_{\text{send}}$: этот оракул моделирует возможность противника активно вмешиваться в обмен сообщениями в канале связи. Запросом к нему является тройка $(U, i, M)$. В ответ на запрос оракул вернёт противнику пятёрку $(resp, sid, pid, acc, term)$,

где $resp$ — ответ оракула-участника $\mathcal{O}_U^i$ на запрос $M$; $sid, pid, acc, term$ — значения параметров оракула $\mathcal{O}_U^i$ после ответа на запрос $M$.

2) $\mathcal{O}_{\text{exec}}$: этот оракул моделирует возможность противника прочитать сообщения, пересланные участниками друг другу в ходе сессии. Запросом к нему является четвёрка $(A, i, B, j)$. Оракул формирует ответ на запрос, используя оракул $\mathcal{O}_{\text{send}}$ для реализации штатного протокольного взаимодействия участников $A$ и $B$. Ответом на запрос будет последовательность ответов оракула $\mathcal{O}_{\text{send}}$ на запросы, сделанные к нему оракулом $\mathcal{O}_{\text{exec}}$.

   Для этого оракула есть некоторые исключительные ситуации. Во-первых, если хотя бы один из оракулов-участников ($\mathcal{O}_A^i$ или $\mathcal{O}_B^j$) не находится в исходном состоянии (т. е. к нему уже был осуществлён хотя бы один запрос), то оракул $\mathcal{O}_{\text{exec}}$ вернёт специальное значение, сигнализирующее о невозможности выполнить запрос. Во-вторых, такое значение может быть возвращено ещё и в случае, если используется BPR-модель непараллельного типа, а для $A$ или $B$ есть оракул-участник с начатой, но не завершённой сессией.

   Легко видеть, что работу оракула $\mathcal{O}_{\text{exec}}$ противник может реализовать и самостоятельно, ведь для этого нужен только оракул $\mathcal{O}_{\text{send}}$. Выделение отдельного оракула для пассивного прослушивания противником канала связи является основной особенностью модели BPR по сравнению с общей моделью для оценки стойкости AKE-протоколов. За счёт этого удаётся отделить действия противника, связанные с изменением передаваемых сообщений, от пассивного прослушивания. Это особенно важно для PAKE-протоколов, так как их основная задача в том, чтобы самой эффективной атакой был online-перебор, который реализуется именно активным вмешательством в канал связи.

3) $\mathcal{O}_{\text{rev}}$: этот оракул позволяет противнику получить сессионный ключ участника сети в выбранной сессии. Запросом к оракулу является пара $(U, i)$, где $U$ — идентификатор участника; $i$ — номер сессии. Ответом на запрос будет параметр $sk$ оракула $\mathcal{O}_U^i$ (даже если он равен UNDEF).

   Наличие у противника доступа к этому оракулу выглядит менее естественно, чем возможность делать запросы к $\mathcal{O}_{\text{send}}$ и $\mathcal{O}_{\text{exec}}$. Следуя мотивации из оригинальной работы [8], отметим, что этот оракул призван отразить одно из свойств безопасности AKE-протоколов, а именно: потеря сеансовых ключей, полученных в результате выполнения AKE-протокола, не должна привести к возможности вскрытия остальных (ещё не потерянных) сеансовых ключей. Потеря сеансовых ключей участников сети возможна по целому ряду причин: использование нестойких криптографических механизмов в последующих протоколах (которые работают уже на основе выработанного сеансового ключа) или утечка ключа по причине ослабления защиты после завершения сессии (например, ключ может быть извлечён с помощью атаки, описанной в [19]).

   Дополнительно отметим, что наделение противника возможностью узнавать сеансовые ключи позволяет в определённой мере достичь модульности анализа, т. е. изолировать анализ AKE-протокола от условий его использования: даже при неаккуратном дальнейшем использовании результатов его работы протокол сам по себе должен быть стойким.

4) $\mathcal{O}_{\text{test}}$: этот оракул не отражает какой-либо возможности противника, а нужен только для формализации угрозы, т. е. задачи противника по нарушению целевых свойств безопасности AKE-протоколов. Как и для $\mathcal{O}_{\text{rev}}$, запросом к этому оракулу является пара $(U, i)$. Если для оракула-участника $\mathcal{O}_U^i$ параметр $acc$ ра-

вен FALSE, то оракул $\mathcal{O}_{\text{test}}$ вернёт символ ошибки $\perp$. Если $acc = \text{TRUE}$, то для формирования ответа оракул сначала выбирает бит $b$ в соответствии с равномерным распределением на множестве $\{0, 1\}$. Если $b = 1$, то оракул возвращает ключ $sk$ оракула-участника $\mathcal{O}_U^i$; если $b = 0$, то — ключ, выбранный в соответствии с равномерным распределением на множестве всех сеансовых ключей.
Формулировка угрозы в виде задачи отличения некоторого параметра от случайной строки является очень распространённой в математической криптографии [20] и отражает идею о том, что противник не должен получить об этом параметре никакой информации.

Особо отметим, что в BPR-модели противник может делать к оракулу $\mathcal{O}_{\text{test}}$ лишь один запрос, поэтому эту модель в литературе обычно называют моделью с угрозой FtG-типа («Find-then-Guess»). Развитие этой модели с расширением угрозы до так называемого RoR-типа («Real-or-Random») описано в работе [21]. Основное отличие моделей в том, что RoR-тип допускает больше одного запроса к оракулу $\mathcal{O}_{\text{test}}$ и не содержит оракула $\mathcal{O}_{\text{rev}}$.

Модель BPR формализует задачу оценки стойкости для PAKE-протоколов в общем. Для конкретных протоколов модель может быть уточнена путём добавления оракулов, необходимых для анализа конкретного протокола. Например, это может быть случайный оракул, которым заменяется используемая в протоколе хэш-функция, т. е. оракулы-участники тоже обращаются к этому оракулу вместо вычисления хэш-функции. Особенности использования модели со случайным оракулом («random-oracle model» — ROM) подробно рассматриваются в [22].

**Преимущество противника.** Прежде чем определить преимущество противника в модели BPR, необходимо ввести понятия *партнёрства* (*Partnering*) и *доступности* (*Freshness*) для оракулов-участников. Комментарии по поводу смысла этих понятий даны после определения преимущества противника.

Оракулы-участники $\mathcal{O}_A^i$ и $\mathcal{O}_B^j$ называются *партнёрами*, если выполнены следующие условия:

1) $acc_A^i = acc_B^j = \text{TRUE}$;
2) $sid_A^i = sid_B^j$;
3) $pid_A^i = B$, $pid_B^j = A$;
4) $sk_A^i = sk_B^j$;
5) не существует другой пары оракулов-участников с таким же значением $sid$, как у $\mathcal{O}_A^i$ и $\mathcal{O}_B^j$, для каждого из которых выполнено равенство $acc = \text{TRUE}$.

Оракул-участник $\mathcal{O}_A^i$ называется *доступным*, если выполнены следующие условия:

1) противник не делал запрос $(A, i)$ к оракулу $\mathcal{O}_{\text{rev}}$;
2) противник не делал запрос $(B, j)$ к оракулу $\mathcal{O}_{\text{rev}}$, где $\mathcal{O}_B^j$ — партнёр оракула $\mathcal{O}_A^i$.

Определим преимущество противника в BPR-модели для некоторого протокола $\mathcal{P}$. Противник в этой модели имеет доступ к оракулам $\mathcal{O}_{\text{send}}$, $\mathcal{O}_{\text{exec}}$, $\mathcal{O}_{\text{rev}}$, $\mathcal{O}_{\text{test}}$ и, возможно, дополнительным оракулам, необходимым для анализа конкретного протокола. Задача противника состоит в распознавании бита $b$, выбираемого оракулом $\mathcal{O}_{\text{test}}$, поэтому в результате работы противник возвращает свою «догадку» о нём — бит $b'$. Событие $SUCC$ состоит в том, что $b' = b$ и запрос к оракулу $\mathcal{O}_{\text{test}}$ был сделан для доступного оракула-участника (причём доступность должна сохраняться на протяжении всей работы противника, а не только до запроса к $\mathcal{O}_{\text{test}}$). Преимущество противника $\mathcal{A}$ по отношению к протоколу $\mathcal{P}$ в BPR-модели (определяемое стандартно в соответствии с п. 1.1), следуя оригинальной работе, будем обозначать через $\mathbf{Adv}_{\mathcal{P}}^{AKE}(\mathcal{A})$.

Неформально, модель противника можно сравнить с правилами игры между противником и криптосистемой. Хорошая игра не должна допускать выигрышную для одной из сторон стратегию, согласно которой другая сторона вообще не принимает участие в игре. В данном случае понятие доступного оракула необходимо, чтобы исключить следующий сценарий действий противника, который позволяет ему выигрывать, не зная ничего о строении анализируемого протокола. В нём противник делает запрос к оракулу $\mathcal{O}_{\text{exec}}$, после которого два оракула-участника становятся партнёрами, разделяя общий сеансовый ключ. После этого противник делает запросы к оракулам $\mathcal{O}_{\text{test}}$ и $\mathcal{O}_{\text{rev}}$: первый запрос — для одного из этих оракулов-участников, а второй — для другого. Чтобы угадать бит $b$ оракула $\mathcal{O}_{\text{test}}$, противнику достаточно просто сравнить полученные ключи.

**Пример 2.** Опишем противника $\mathcal{A}$, осуществляющего угрозу в модели BPR для протокола, описанного в примере 1. Он выполняет следующую последовательность действий:

1) случайно выбирает $z$ и вычисляет $Z = g^z$;
2) делает запрос $(B, 1, (A, Z))$ к оракулу $\mathcal{O}_{\text{send}}$. В качестве ответа $\mathcal{A}$ получает сообщение $(B, Y)$ и значения внутренних переменных $sid = (A, Z)||(B, Y)$, $pid = A$, $acc = term = \text{TRUE}$;
3) вычисляет $K = Y^z$;
4) делает запрос $(B, 1)$ к оракулу $\mathcal{O}_{\text{test}}$. Оракул возвращает ключ $K'$;
5) если $K = K'$, то $\mathcal{A}$ возвращает в качестве результата 1, иначе 0.

Заметим, что $\mathbf{Adv}^{AKE}(\mathcal{A})$ для данного протокола отличается от 1 на величину вероятности случайно выбрать ключ $K'$ равным $g^{zy}$. Также стоит отметить, что оракул $\mathcal{O}_B^1$ является доступным, так как запросов к оракулу $\mathcal{O}_{\text{rev}}$ противник не делал.

Данный пример нужен исключительно для того, чтобы проиллюстрировать возможности и цели противника в модели $BPR$. Очевидно, что рассмотренный протокол не является в этой модели стойким, так как не использует каких-либо механизмов аутентификации сторон.

### 1.3. Стойкость PAKE-протоколов

Рассмотрим пример того, как в BPR-модели реализуется онлайн-перебор паролей на примере упрощённого PAKE-протокола $\mathcal{P}$ без подтверждения ключа.

**Пример 3.** Протокол $\mathcal{P}$ устроен следующим образом: клиент и сервер хранят в секрете общий пароль $pw$, который является элементом несекретного подмножества $PW$ мощности $N$ группы $G = \langle g \rangle$ простого порядка. Клиент $C$ направляет серверу сообщение $(C, X = pw \cdot g^x)$, а сервер $S$ отвечает сообщением $(S, Y = pw \cdot g^y)$ ($x, y$ выбираются из множества $\{1, \ldots, q-1\}$ случайно и равновероятно). Далее стороны вычисляют общий ключ $K = (X/pw)^y = (Y/pw)^x$.

Опишем противника $\mathcal{A}$, осуществляющего угрозу в модели BPR для описанного протокола. Противник выполняет следующие действия:

1) полагает $T = PW$, $i = 0$;
2) выбирает $pw'$ из $T$ и полагает $T = T \setminus \{pw'\}$;
3) делает запрос $(S, i, (C, pw' \cdot g))$ к оракулу $\mathcal{O}_{\text{send}}$, извлекая из его ответа параметр $resp$, который является элементом группы $G$;
4) делает запрос $(S, i)$ к оракулу $\mathcal{O}_{\text{rev}}$, получая в ответ ключ $K \in G$;
5) если $K \neq resp/pw'$, то полагает $i = i + 1$ и возвращается на шаг 2;
6) делает запрос $(S, i+1, (C, pw' \cdot g))$ к оракулу $\mathcal{O}_{\text{send}}$, извлекая из его ответа $resp \in G$;

7) делает запрос $(S, i+1)$ к оракулу $\mathcal{O}_{\text{test}}$, получая в ответ ключ $K$;
8) заканчивает работу с результатом 1, если $K = resp/pw'$, и 0 в противном случае.

Переход противника $\mathcal{A}$ на шаг 6 происходит в том случае, если $pw' = pw$, где $pw$ — пароль, разделяемый $C$ и $S$. Таким образом, противник просто осуществляет полный перебор паролей. Такой перебор называется онлайн-перебором, так как он предполагает активное взаимодействие противника с участниками сети.

Если противник может делать $N+1$ запрос к $\mathcal{O}_{\text{send}}$, то $\mathbf{Adv}^{AKE}(\mathcal{A}) = 2(1-1/|G|)-1 = 1 - 2/|G|$. Отсюда следует, что

$$\mathbf{Insec}_{\mathcal{P}}^{AKE}(t, q_{\text{send}} = N+1, q_{\text{rev}} = N, q_{\text{exec}} = 0) \geqslant 1 - 2/|G|,$$

где вычислительные ресурсы $t$ равны $c \cdot N$; $c$ — некоторая небольшая константа. Здесь через $q_{\text{send}}, q_{\text{rev}}, q_{\text{exec}}$ обозначено количество запросов к соответствующим оракулам.

Понятно, что описанный противник может быть изменён так, чтобы работать в условиях, когда ему доступно лишь $n \leqslant N+1$ запросов к оракулу $\mathcal{O}_{\text{send}}$, и иметь вероятность успеха порядка $n/N$.

Для любого PAKE-протокола противник может осуществить угрозу отличения ключа так, как показано в примере 3. Стойкий PAKE-протокол — это такой протокол, для которого сценарий с онлайн-перебором является наиболее эффективной атакой, причём ни относительно существенное увеличение вычислительных ресурсов, ни количества запросов к любым оракулам, кроме $\mathcal{O}_{\text{send}}$, не приводят к появлению более эффективных методов реализации угрозы.

Рассмотрим некоторые существующие PAKE-протоколы с точки зрения особенностей их строения, применимости модели BPR для их анализа и оценок их стойкости.

По формату этапа аутентификации протоколы выработки общего ключа на основе пароля можно подразделить на два типа:

1) Выработанный сессионный ключ непосредственно используется в качестве секретного аргумента функции вычисления аутентификационной информации.
2) Из выработанной обеими сторонами величины с помощью одного алгоритма создаётся сессионный ключ, а с помощью другого алгоритма генерируется аутентификационная информация.

Для протоколов первого типа BPR-модель не является корректной, так как существует противник, реализующий угрозу независимо от анализируемого протокола: ключ, выданный оракулом $\mathcal{O}_{\text{test}}$, проверяется на подлинность путём подстановки в известную противнику функцию получения аутентификационной информации и сравнения полученного результата с тем, что передавался по каналу связи. Поэтому BPR-модель не подходит для оценки стойкости протоколов первого типа. Авторам удалось найти доказательство безопасности протоколов такого типа только для этапа генерации ключа.

Для протоколов второго типа определённая выше угроза корректна.

Рассмотрим оценки стойкости, полученные для двух связанных друг с другом протоколов второго типа: One-Encryption Key-Exchange (OEKE) и AuthA. Протокол AuthA предложен M. Bellare и P. Rogaway в 2000 г. в [2] и стандартизирован IEEE (P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques). В 2003 г. в работе [3] E. Bresson, O. Chevassut и D. Pointcheval привели оценку стойкости упрощённой версии этого протокола, которую назвали OEKE, в BPR-модели параллельного типа (concurrent) со случайным оракулом и оракулом

идеального шифра. В этой работе получена также оценка стойкости протокола OAuth в модели с угрозой ложной аутентификации, не основанной на задаче различения.

В терминологии настоящей работы оценка стойкости протокола OEKE в модели BPR, полученная в [3], выглядит следующим образом:

$$\mathbf{Insec}^{AKE}_{OEKE}(t, q_s, q_p, q_h, q_e) \leqslant 3\,\frac{q_s}{N} + 8q_h \cdot \mathbf{Insec}^{CDH}_{G}(t') + \frac{(2q_e + 3q_s + 3q_p)^2}{q-1} + \frac{q_h^2 + 4q_s}{2^{l_1}},$$

где $q_s, q_p, q_h, q_e$ — максимально допустимые количества запросов к оракулам $\mathcal{O}_{\text{send}}$, $\mathcal{O}_{\text{exec}}$, случайному оракулу и оракулу идеального шифра соответственно; $q$ — мощность используемой для вычислений группы $G$; $t' \leqslant t + (q_s + q_p + q_e + 1)\tau_G$; $\tau_G$ — трудоёмкость возведения в степень в группе $G$; $l_1$ — размер выхода хэш-функции, используемой для вычисления аутентификационной информации; $\mathbf{Insec}^{CDH}_{G}(t')$ — вероятность успеха наиболее эффективного алгоритма решения задачи CDH (её определение приведено в п. 4) в группе $G$.

Рассмотрим оценку подробнее. Величины $2^{l_1}$ и $q$ на практике обычно достаточно большие (не меньше $2^{256}$). Величину числителей двух правых слагаемых определяют значения $q_h$ и $q_e$. Действительно, на практике для вычисления хэш-функции и зашифрования противник просто вычисляет эти функции и не обращается ни к каким оракулам, т. е. эти параметры ограничены исключительно его вычислительными ресурсами. Поэтому количество $q_p$ пассивно прослушанных сеансов взаимодействия участников протокола существенно меньше $q_h$ и $q_e$. Тем более меньше этих параметров оказывается величина $q_s$, которая на практике ограничивается организационными мерами (например, использованием счётчиков неуспешных сессий). Таким образом, если вычислительные ресурсы противника существенно меньше $\sqrt{q}$ (для $q \geqslant 2^{256}$ это предположение на сегодняшний день представляется вполне правдоподобным), то два правых слагаемых пренебрежимо малы.

Величина $\mathbf{Insec}^{CDH}_{G}(t')$ для хорошо подобранных групп (например, групп точек эллиптических кривых, выбранных с учётом критериев из работы [23]) оценивается с учётом общих методов дискретного логарифмирования, т. е. $\mathbf{Insec}^{CDH}_{G}(t') \approx (t')^2/q$. Мощность $q$ используемой подгруппы точек эллиптической кривой должна быть выбрана таким образом, чтобы слагаемое $8q_h \cdot \mathbf{Insec}^{CDH}_{G}(t')$ было пренебрежимо малым с учётом предполагаемых вычислительных ресурсов противника. Здесь важно то, что это слагаемое можно сделать пренебрежимо малым с помощью изменения численного параметра протокола, а не принципа его работы.

Таким образом, величину $\mathbf{Insec}^{AKE}_{OEKE}(t, q_s, q_p, q_e, q_h)$ определяет первое слагаемое, которое соответствует атаке онлайн-перебора из примера 3. Оценка показывает, что эта атака является единственно эффективной для данного протокола в указанных предположениях о вычислительных возможностях и параметрах протокола. Именно о PAKE-протоколах с оценками стойкости такого типа говорят, что они являются стойкими в BPR-модели.

В работе [3] в условиях BPR-модели определяется также угроза ложной аутентификации нераспознавательного типа. Через $\mathbf{Adv}^{c-auth}_{oeke}(\mathcal{A})$ обозначена вероятность того, что противнику удалось выдать себя за клиента, которым он не является, т. е. оракул-сервер пришёл в состояние $acc = \text{TRUE}$, но не существует оракула-клиента, для которого этот оракул-сервер является партнёром. Недостатком данной модели является то, что в ней оценивается возможность противника выдавать себя лишь за клиента, но не за сервер.

## 2. Описание протокола SESPAKE

Через $V_n$ будем обозначать множество наборов длины $n$ с элементами из поля GF(2). Далее все операции с точками эллиптической кривой проводятся в подгруппе $E = \langle P \rangle$ простого порядка $q$ группы точек некоторой эллиптической кривой порядка $m$. Через $0_E$ обозначим нейтральный элемент группы $E$; для конечного множества $A$ через $a \xleftarrow{\mathcal{U}} A$ будем обозначать операцию выбора элемента $a$ из $A$ в соответствии с равномерным распределением.

Для конкретной реализации протокола необходимо зафиксировать следующие параметры:

— $l$ — произвольное натуральное число;
— доказуемо псевдослучайные точки $P, Q_1, \ldots, Q_l \in E$. Доказуемая псевдослучайность гарантирует, что кратность любой точки относительно любой другой неизвестна (подробнее об этом см. в [13]);
— строковые константы $T_A$ и $T_B$, которые используются клиентами и серверами соответственно.

Через $F$ далее будем обозначать функцию PBKDF2, определённую в [24], а через $H_{256}$ — хэш-функцию ГОСТ Р 34.11-2012 [25] с длиной выхода равной 256 битам. Через HMAC обозначим алгоритм [26] с длиной выхода 256 бит.

Протокол предполагает взаимодействие двух участников: клиента и сервера. Участники сети обладают фиксированными идентификаторами, которые являются байтовыми строками некоторой фиксированной длины $N$ (длина одна для всех участников протокола), множество всех идентификаторов обозначим $ID$ ($ID = V_8^N$). Если $A$ — участник протокола, то его идентификатор из $ID$ будем обозначать так же $A$.

Клиент в схеме работы протокола обозначается через $A$ и хранит пароль $PW \in V_8^k$.

Сервер в схеме работы протокола обозначается через $B$ и хранит следующие параметры для каждого клиента, с которым может взаимодействовать:

— число $ind \in \{1, \ldots, l\}$;
— строку $salt \in V_{64}$;
— точку $Q_{PW} = F(PW, salt, 2000)Q_{ind}$.

В секрете должны храниться только параметры $PW$ и $Q_{PW}$.

Схема работы протокола приведена в таблице. Для удобства дальнейших рассуждений будем обозначать $\text{SESPAKE}_{KA}$ и называть *протоколом выработки общего ключа* ту часть протокола, которая предшествует первому вычислению участником $A$ значения функции HMAC. Оставшуюся часть протокола будем называть *протоколом подтверждения ключа с аутентификацией* и обозначать $\text{SESPAKE}_{KC}$. Это деление наглядно продемонстрировано в таблице. Под протоколом SESPAKE будем понимать аутентифицированную выработку общего ключа по протоколу $\text{SESPAKE}_{KA}$ с последующим его подтверждением в соответствии с протоколом $\text{SESPAKE}_{KC}$.

**Описание протокола SESPAKE**

| Фиксированные открытые параметры: $l, P, Q_1, \ldots, Q_l, m, q$ | | | |
|---|---|---|---|
| $A\ [PW]$ | | $B\ [Q_{PW}, ind, salt]$ | |
| | $\xrightarrow{A}$ | | SESPAKE$_{KA}$ |
| $Q_{PW}^A = F(PW, salt, 2000)Q_{ind}$ | $\xleftarrow{ind,\ salt}$ | | |
| $z_A = 0$, $\alpha \xleftarrow{\mathcal{U}} \{1, \ldots, q-1\}$ | | | |
| $u_1 = \alpha \cdot P - Q_{PW}^A$ | $\xrightarrow{u_1}$ | if $u_1 \notin E \Rightarrow$ FINISH | |
| | | $Q_B = u_1 + Q_{PW}$ | |
| | | $z_B = 0$, $\beta \xleftarrow{\mathcal{U}} \{1, \ldots, q-1\}$ | |
| | | if $\frac{m}{q}Q_B = 0_E \Rightarrow Q_B = P$, $z_B = 1$ | |
| | | $src = \left(\frac{m}{q} \cdot \beta\right) Q_B$ | |
| | | $K_B = H_{256}(src)$ | |
| if $u_2 \notin E \Rightarrow$ FINISH | $\xleftarrow{u_2}$ | $u_2 = \beta \cdot P + Q_{PW}$ | |
| $Q_A = u_2 - Q_{PW}^A$ | | | |
| if $\frac{m}{q}Q_A = 0_E \Rightarrow Q_A = P$, $z_A = 1$ | | | |
| $src = \left(\frac{m}{q} \cdot \alpha\right) Q_A$ | | | |
| $K_A = H_{256}(src)$ | | | |
| $tag_A = T_A\|\|A\|\|ind\|\|salt\|\|u_1\|\|u_2$<br>$M_A = \text{HMAC}_{K_A}(tag_A)$ | $\xrightarrow{M_A}$ | $tag = T_A\|\|A\|\|ind\|\|salt\|\|u_1\|\|u_2$<br>$M = \text{HMAC}_{K_B}(tag)$ | SESPAKE$_{KC}$ |
| | | If $M \neq M_A$ or $z_B \neq 0 \Rightarrow$ FINISH | |
| $tag = T_B\|\|B\|\|ind\|\|salt\|\|u_1\|\|u_2$<br>$M = \text{HMAC}_{K_A}(tag)$ | $\xleftarrow{M_B}$ | $tag_B = T_B\|\|B\|\|ind\|\|salt\|\|u_1\|\|u_2$<br>$M_B = \text{HMAC}_{K_B}(tag_B)$ | |
| If $M \neq M_B$ or $z_A \neq 0 \Rightarrow$ FINISH | | | |

## 3. Модели противника для оценки стойкости SESPAKE

В данной работе оценка стойкости протокола SESPAKE проводится в два этапа. Сначала оценивается стойкость протокола SESPAKE$_{KA}$ в непараллельной модели BPR. Далее оценивается стойкость протокола SESPAKE в модели с угрозой ложной аутентификации распознавательного типа, описываемой ниже. Обоснование стойкости протокола SESPAKE существенно основывается на стойкости протокола SESPAKE$_{KA}$.

Уточним, что стойкость протокола SESPAKE$_{KA}$ оценивается в BPR-модели непараллельного типа с дополнительным случайным оракулом $\mathcal{O}_H$, которым заменяется хэш-функция $H_{256}$, используемая для вычисления ключей $K_A$ и $K_B$.

Для протокола SESPAKE рассматривается модель противника с некоторым вариантом угрозы ложной аутентификации. Более точно, рассматривается угроза отличения строки $M$, полученной с помощью имеющегося у участника ключа, от строки $M'$, полученной с помощью случайно выбранного ключа.

Опишем возможности противника $\mathcal{A}$ для протокола SESPAKE. Помимо возможностей, которые есть у противника, рассматривавшегося для протокола SESPAKE$_{KA}$, противник $\mathcal{A}$ может делать запросы к оракулам $\mathcal{O}_{\text{HMAC}}$, $\mathcal{O}_{\text{check}}$ и $\mathcal{O}_{\text{auth}}$. Через $IDS_{A,i}$ обозначим конкатенацию тех данных, которые были пересланы по каналу связи с точки зрения оракула $\mathcal{O}_A^i$ до этапа подтверждения ключа. Эти оракулы работают по следующим правилам:

— $\mathcal{O}_{\text{HMAC}}$: реализует семейство случайных отображений $\{G_K : V^* \to V_{256}\}_{K \in V_{256}}$, которые выбираются случайно, независимо и равновероятно перед началом работы; на запрос $(K, T)$ оракул $\mathcal{O}_{\text{HMAC}}$ возвращает значение $G_K(T)$;
— $\mathcal{O}_{\text{check}}$: в качестве ответа на запрос $(A, i)$ этот оракул возвращает значение $G_{K_{A,i}}(T_A||IDS_{A,i})$, где $K_{A,i}$ — сессионный ключ, выработанный оракулом $\mathcal{O}_A^i$;
— $\mathcal{O}_{\text{auth}}$: с помощью этого оракула определяется угроза. Оракул $\mathcal{O}_{\text{auth}}$ принимает на вход пару $(A, i)$, случайно и равновероятно выбирает бит $b \in \{0, 1\}$ и в зависимости от $b$ возвращает либо пару $(M_A, M')$ (если $b = 0$), либо пару $(M_A, M)$ (если $b = 1$). При этом $M' = G_{K'}(T_B||IDS_{A,i})$, где $K' \xleftarrow{\mathcal{U}} V_{256}$; $M_A = G_{K_{A,i}}(T_A||IDS_{A,i})$; $M = G_{K_{A,i}}(T_B||IDS_{A,i})$.

Как и в модели для протокола $\text{SESPAKE}_{KA}$, противник может делать запрос к оракулу $\mathcal{O}_{\text{auth}}$ только для тех оракулов, которые допускают тестирование. Оракулом, допускающим тестирование, называется такой оракул $\mathcal{O}_A^i$, по отношению к которому не было запросов к $\mathcal{O}_{\text{check}}$ и для которого противник не получил значение $M$ тривиальным образом, то есть не сделал запрос к $\mathcal{O}_{\text{check}}$ для оракула, разделяющего с $\mathcal{O}_A^i$ общий ключ. Говоря неформально, в такой модели противник перед тем, как попытаться осуществить угрозу, предупреждает, что он будет осуществлять угрозу по отношению именно к этому участнику и сеансу. Подчеркнем, что противник выбирает «цель» не в начале, а в процессе работы.

В качестве результата противник возвращает значение $a \in \{0, 1\}$. Через $\text{SUCC}_{\text{auth}}$ обозначим событие, состоящее в том, что $a = b$, где $b$ — значение, которое выбрал оракул $\mathcal{O}_{\text{auth}}$.

## 4. Сопутствующие задачи и соотношения между ними

Рассмотрим задачи, которые используются при обосновании стойкости протокола SESPAKE. Все задачи формулируются для подгруппы $E$ простого порядка $q$ группы точек эллиптической кривой порядка $m$ с образующим элементом $P$. Через $\tau$ будем обозначать трудоёмкость вычисления кратной точки в этой подгруппе.

Преимущества противников во всех задачах определяются как вероятность того, что результат соответствующего задаче эксперимента будет равен 1 (обозначается $\mathbf{Exp}(\mathcal{A}) = 1$). Трудоёмкость решения задач определяется величиной **Insec**, задаваемой так, как описано в п. 1.1.

### 4.1. Определения

**Задача CDH** (вычислительная задача Диффи — Хеллмана) определяется для противника $\mathcal{A}$ экспериментом $\mathbf{Exp}^{\text{CDH}}$:

— противнику $\mathcal{A}$ передаются точки $X = xP$, $Y = yP$, где $x, y \xleftarrow{\mathcal{U}} \mathbb{Z}_q$;
— противник $\mathcal{A}$ возвращает точку $Z$;
— если $Z = xyP$, то в качестве результата $\mathbf{Exp}^{\text{CDH}}(\mathcal{A})$ данного эксперимента выдаётся 1, иначе 0.

Для данных $X = xP$ и $Y = yP$ через $\text{CDH}_P(X, Y)$ будем обозначать точку $xyP$. Там, где из контекста ясно, о каком $P$ идёт речь, будем писать просто $\text{CDH}(X, Y)$. Заметим, что справедливы следующие соотношения:

— $\text{CDH}_P(P, Y) = Y$;
— $\text{CDH}(aX, Y) = a \cdot \text{CDH}(X, Y)$;
— $\text{CDH}(X + Y, Z) = \text{CDH}(X, Z) + \text{CDH}(Y, Z)$.

**Задача SCDH** [27] отличается от CDH тем, что противнику передаётся только точка $X$, а его задача состоит в том, чтобы вычислить $\mathrm{CDH}(X, X)$. Через $\mathrm{SCDH}(X)$, где $X = xP$, будем обозначать точку $x^2P$.

**Задача QCCDH** определяется для противника $\mathcal{A}$ экспериментом $\mathbf{Exp}^{\mathrm{QCCDH}}$:

— противнику передаются точки $X, Q \xleftarrow{\mathcal{U}} E$;
— противник возвращает точки $Y$, $U$ и $V$;
— если $U = \mathrm{CDH}(X, Y)$ и $V = \mathrm{CDH}(X - Q, Y - Q)$, то в качестве результата возвращается 1, иначе 0.

**Задача $\mathbf{QPCCDH}_{\mathcal{D}}$.**
Пусть $\mathcal{D} \subset \mathbb{Z}_q$ — множество паролей. Эксперимент $\mathbf{Exp}^{\mathrm{QPCCDH}}(\mathcal{A})$, определяющий задачу $\mathrm{QPCCDH}_{\mathcal{D}}$, состоит из двух этапов, а противник $\mathcal{A}$, участвующий в нём, использует при работе на разных этапах две вероятностные ленты, заполнение первой из которых возвращается вместе со значением $Y$. Заполнение случайной ленты для первого этапа может быть задано перед началом эксперимента произвольным образом. Если оно не задано, то выбирается случайно и равновероятно.

Эксперимент $\mathbf{Exp}^{\mathrm{QPCCDH}}(\mathcal{A})$:

— противнику $\mathcal{A}$ передаются точки $Q, X \xleftarrow{\mathcal{U}} E$;
— противник возвращает $Y \in E$ и $u$ — заполнение своей случайной ленты, которая использовалась при вычислениях на первом этапе;
— противнику передаётся пароль $r \xleftarrow{\mathcal{U}} \mathcal{D}$;
— противник возвращает $K \in E$;
— если $K = \mathrm{CDH}(X - rQ, Y - rQ)$, то в качестве результата возвращается 1, иначе 0.

**Замечание 1.** Противник, угадавший перед возвращением $Y$ пароль $r$, гарантированно решает задачу, возвращая $Y = P + rQ$ и $K = X - rQ$.

**Замечание 2.** Возврат заполнения случайной ленты обеспечивает возможность «запускать» противника $\mathcal{A}$ так, чтобы результат его работы на первом этапе (точка $Y$) для одних и тех же входных данных не менялся. Фактически это означает, что, получив от $\mathcal{A}$ точку $Y$, можно многократно «запускать» $\mathcal{A}$ для работы только на втором этапе.

**Задача $\mathbf{S\text{-}QPCCDH}_{\mathcal{D},s}$** отличается от задачи $\mathrm{QPCCDH}_{\mathcal{D}}$ тем, что вместо одного ключа $K$ противник $\mathcal{A}$ может выдать в качестве ответа множество ключей $\mathcal{K}$ мощности $s \in \mathbb{N}$. В эксперименте $\mathbf{Exp}^{\text{S-QPCCDH}}(\mathcal{A})$ в качестве результата возвращается 1, если точка $\mathrm{CDH}_P(X - rQ, Y - rQ)$ есть в множестве $\mathcal{K}$.

Везде далее, если из контекста ясно, для каких параметров $s$ и $\mathcal{D}$ рассматриваются описанные задачи, их названия приводятся без индексов.

### 4.2. Соотношения между задачами

Докажем утверждения, итоговая цель которых — оценить сверху нестойкость задачи S-QPCCDH через нестойкость задачи SCDH. Согласно [27], наиболее эффективным методом её решения является дискретное логарифмирование.

**Теорема 1.** Справедливо неравенство

$$\mathbf{Insec}^{\mathrm{QCCDH}}(t) \leqslant \mathbf{Insec}^{\mathrm{SCDH}}(t + 4\tau).$$

***Доказательство.*** Пусть $\mathcal{A} \in \mathcal{A}(t)$ — противник, решающий задачу QCCDH с вероятностью успеха $\varepsilon$. Построим противника $\mathcal{A}'$, решающего задачу SCDH с помощью $\mathcal{A}$.

Точку, полученную противником $\mathcal{A}'$ в рамках эксперимента $\mathbf{Exp}^{\mathrm{SCDH}}$, обозначим через $Q$. Противник $\mathcal{A}'$ осуществляет следующие действия: $b \xleftarrow{U} \mathbb{Z}_q$; подаёт $\mathcal{A}$ на вход пару $(X = 2Q + bP, Q)$, получая в ответ тройку $Y, U, V$. Заметим, что пара $(X, Q)$, сформированная таким образом, принимает каждое значение из $E \times E$ с одинаковой вероятностью.

Если $\mathcal{A}$ успешно решил задачу QCCDH, то

$$U = \mathrm{CDH}(2Q + bP, Y) = 2 \cdot \mathrm{CDH}(Q, Y) + bY,$$
$$V = \mathrm{CDH}(Q + bP, Y - Q) = \mathrm{CDH}(Q, Y) - \mathrm{SCDH}(Q) + bY - bQ.$$

Таким образом, точка

$$R = 1/2 \cdot U - V + b/2 \cdot Y - bQ$$

является решением задачи SCDH для точки $Q$ с вероятностью $\varepsilon$. Поэтому в силу произвольности $\mathcal{A}$ получаем следующую цепочку неравенств:

$$\mathbf{Insec}^{\mathrm{QCCDH}}(t) = \mathbf{Adv}^{\mathrm{QCCDH}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{SCDH}}(\mathcal{A}') \leqslant \mathbf{Insec}^{\mathrm{SCDH}}(t + 4\tau).$$

Теорема 1 доказана. ■

Заметим, что в работе [9] доказательство аналогичного утверждения требует корректировки. Неточность заключается в том, что итоговая оценка не учитывает, что построенный в [9] противник не может решить задачу CDH при $b = 0$ или $b = 1$, а также то, что входные аргументы для $\mathcal{A}$ выбираются не в соответствии с равномерным распределением.

Приведём формулировку леммы из работы [9], которая понадобится для доказательства теоремы 2.

**Лемма 1** [9]. Для конечных множеств $X$ и $Y$ пусть $A \subset X \times Y$ таково, что $\Pr[(x, y) \in A] \geqslant \varepsilon$. Пусть также для произвольного $\alpha < \varepsilon$

$$B = \{x \in X : \Pr_{y' \in Y}[(x, y') \in A] \geqslant \varepsilon - \alpha\}.$$

Тогда $\Pr[B] \geqslant \alpha$ и $\Pr[B|A] \geqslant \alpha/\varepsilon$.

**Теорема 2.** Пусть $|\mathcal{D}| = n$. Для $t$, такого, что

$$\frac{2}{n} \geqslant \mathbf{Insec}^{\mathrm{QPCCDH}_{\mathcal{D}}}(t) \geqslant \frac{1}{n} + \varepsilon$$

для некоторого $\varepsilon \leqslant 1/n$, справедливо неравенство

$$\mathbf{Insec}^{\mathrm{QCCDH}}(2t + 2\tau) \geqslant \frac{n\varepsilon^3}{64}.$$

***Доказательство.*** Пусть $\mathcal{A} \in \mathcal{A}(t)$ — противник, решающий задачу $\mathrm{QPCCDH}_{\mathcal{D}}$ с вероятностью успеха $\dfrac{2}{n} \geqslant \mathbf{Adv}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) \geqslant \dfrac{1}{n} + \varepsilon$.

Пусть $Q, X \in E$ — случайно и независимо выбранные точки, для которых требуется решить задачу QCCDH. Будем использовать противника $\mathcal{A}$ для того, чтобы решить эту задачу.

Выберем случайно и равновероятно заполнение $u$ случайной ленты для первого шага работы противника $\mathcal{A}$ (заметим, что достаточно рассматривать ленту конечной длины $t$). Выберем случайно $r \neq r' \in \mathcal{D}$ и вычислим $\gamma = r' - r$.

Вычислим $Q' = \frac{1}{\gamma}Q$ и $X' = X + rQ'$. Далее дважды используем $\mathcal{A}$, чтобы решить задачу QPCCDH для одинаковых входных параметров первого этапа и разных параметров второго этапа.

Подадим на вход противнику $\mathcal{A}$ пару $Q', X'$, записав на его вероятностную ленту первого этапа значение $u$. Пусть $Y'$ — то, что противник выдал в качестве результата первого этапа (заполнение случайной ленты первого этапа, которое возвратит $\mathcal{A}$, очевидно, равно $u$). В качестве входного параметра второго этапа передадим $\mathcal{A}$ число $r'$. Пусть $K'$ — значение, которое $\mathcal{A}$ возвратит в качестве результата.

Снова используем $\mathcal{A}$, подав на вход первого этапа ту же пару $Q', X'$ и записав на ленту первого этапа $u$. Противник $\mathcal{A}$ возвратит то же самое значение $Y'$. В качестве входного параметра второго этапа теперь передадим число $r$. Пусть $K$ — то, что выдал $\mathcal{A}$ в качестве результата вычислений на втором этапе.

Если противник $\mathcal{A}$ правильно решил обе задачи, то $K' = \mathrm{CDH}(X' - r'Q', Y' - r'Q')$, а $K = \mathrm{CDH}(X' - rQ', Y' - rQ')$. Пусть $Y = Y' - rQ'$, тогда тройка $Y, K, K'$ будет решением задачи QCCDH для входных параметров $X, Q$. Это объясняется тем, что

$$K = \mathrm{CDH}(\underbrace{X'}_{X+rQ'} - rQ', \underbrace{Y' - rQ'}_{Y}) = \mathrm{CDH}(X, Y),$$

$$K' = \mathrm{CDH}(X' - r'Q', Y' - r'Q') = \mathrm{CDH}(X + rQ' - r'Q', Y + rQ' - r'Q) =$$
$$= \mathrm{CDH}(X - (r' - r)Q', Y - (r' - r)Q') = \mathrm{CDH}(X - Q, Y - Q),$$

так как $Q' = \dfrac{1}{r' - r}Q$.

Вероятность решить задачу QCCDH для случайной пары $X, Q$ у противника, действующего описанным образом, не меньше, чем вероятность того, что противник $\mathcal{A}$ решит одновременно задачи для входов $X', Q', r$ и $X', Q', r'$, где $X', Q', r, r'$ случайны и независимы, при условии, что заполнение случайной ленты $u$ первого этапа для обеих задач одинаково. Обозначим это событие через SUCC.

Множество $\Omega_0 = \{(X', Q', u, v, r) : X', Q' \in E,\ u, v \in \{0,1\}^t,\ r \in \mathcal{D}\}$ является пространством элементарных событий случайной величины, которой является результат эксперимента $\mathbf{Exp}^{\mathrm{QPCCDH}}$. По условию теоремы $\Pr_{\Omega_0}\{\mathbf{Exp}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) = 1\} \geqslant \dfrac{1}{n} + \varepsilon$.

Представим $\Omega_0$ в виде декартова произведения $\Omega_1' \times \Omega_1$:

$$\Omega_1' = \{u : u \in \{0,1\}^t\},$$
$$\Omega_1 = \{(X', Q', v, r) : X', Q' \in E,\ v \in \{0,1\}^t,\ r \in \mathcal{D}\}.$$

Пусть $S_1 = \left\{u \in \Omega_1' : \Pr_{\Omega_1}[\mathbf{Exp}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) = 1] \geqslant \dfrac{1}{n} + \dfrac{\varepsilon}{2}\right\}$. Тогда из леммы 1 следует, что

$$\Pr_{\Omega_0}[u \in S_1 : \mathbf{Exp}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) = 1] \geqslant \frac{\varepsilon/2}{1/n + \varepsilon} = \frac{n\varepsilon}{2 + 2n\varepsilon}.$$

Теперь представим $\Omega_1$ в виде декартова произведения $\Omega_2' \times \Omega_2$:

$$\Omega_2' = \{(X', Q') : X', Q' \in E\}, \quad \Omega_2 = \{(v, r) : v \in \{0,1\}^t,\ r \in \mathcal{D}\}.$$

Для любого $u \in S_1$ пусть

$$S_2(u) = \left\{(X', Q') \in E \times E : \Pr_{\Omega_2}[\mathbf{Exp}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) = 1] \geqslant \frac{1}{n} + \frac{\varepsilon}{4}\right\},$$

тогда по лемме 1 справедливо неравенство

$$\Pr_{\Omega_1}[(X',Q') \in S_2(u) : \mathbf{Exp}^{\mathrm{QPCCDH}_{\mathcal{D}}}(\mathcal{A}) = 1] \geqslant \frac{\varepsilon/4}{1/n + \varepsilon/2} = \frac{n\varepsilon}{4 + 2n\varepsilon}.$$

Вероятность успеха противника $\mathcal{A}$ при решении задачи для случайного входа $X', Q', r$ не меньше чем $\dfrac{1}{n} + \varepsilon$, причём вероятность того, что выполнены условия $u \in S_1$, $(X', Q') \in S_2(u)$, не меньше чем

$$\frac{n\varepsilon}{2 + 2n\varepsilon} \frac{n\varepsilon}{4 + 2n\varepsilon} \geqslant \frac{1}{8} \left( \frac{n\varepsilon}{1 + n\varepsilon} \right)^2.$$

Противник $\mathcal{A}$ решает задачу для входа $X', Q', r'$ при фиксированных $u \in S_1$, $(X', Q') \in$ $\in S_2(u)$ с вероятностью не меньше чем $\dfrac{1}{n} + \dfrac{\varepsilon}{4} \geqslant \dfrac{\varepsilon}{4}$. Следовательно, выполняется следующее неравенство:

$$\Pr[\mathrm{SUCC}] \geqslant \left( \frac{1}{n} + \varepsilon \right) \frac{1}{8} \left( \frac{n\varepsilon}{1 + n\varepsilon} \right)^2 \frac{\varepsilon}{4}.$$

Из условия $\dfrac{2}{n} \geqslant \dfrac{1}{n} + \varepsilon$ получаем требуемое неравенство:

$$\mathbf{Insec}^{\mathrm{QCCDH}}(2t + 2\tau) \geqslant \Pr[\mathrm{SUCC}] \geqslant \frac{n\varepsilon^3}{64}.$$

Теорема 2 доказана. ■

**Теорема 3.** Справедливо неравенство

$$\mathbf{Insec}^{\text{S-QPCCDH}_{s,\mathcal{D}}}(t) \leqslant \frac{1}{|\mathcal{D}|} + \sqrt[6]{\frac{\mathbf{Insec}^{\mathrm{SCDH}}(4t + \Theta + O(s\tau))}{|\mathcal{D}|^2} + \frac{2^{13}s^4}{|\mathcal{D}|^2 q}},$$

где $\Theta$ — трудоёмкость поиска пары одинаковых элементов в двух множествах размера $s^2$.

***Доказательство.*** Пусть $\mathcal{A} \in \mathcal{A}(t)$ — противник, решающий задачу $\text{S-QPCCDH}_{s,\mathcal{D}}$ с вероятностью успеха $\dfrac{1}{n} + \varepsilon$, где $n = |\mathcal{D}|$; $R \in E$ — точка, для которой требуется решить задачу SCDH. Будем использовать для решения противника $\mathcal{A}$.

Выберем случайно и независимо друг от друга числа $x_1, x_2, q_1, q_2 \in \mathbb{F}_p$. Вычислим точки $X_i = 2R + x_i P$ и $Q_i = R + q_i P$. Выберем случайно и независимо друг от друга два заполнения $u_1$ и $u_2$ случайной ленты первого этапа противника $\mathcal{A}$. Далее используем противника $\mathcal{A}$ в качестве чёрного ящика таким же образом, как описано в доказательстве теоремы 2, сначала для пары $X_1, Q_1$ и заполнения $u_1$, а потом для пары $X_2, Q_2$ и заполнения $u_2$. С учётом порядка построения противника в теореме 2, в последующих выкладках используем следующие дополнительные обозначения: $r_i, r_i'$, $Q_i' = \dfrac{1}{r_i' - r_i} Q_i$, $X_i' = X_i + r_i Q_i'$ и $Y_i$. Два этих эксперимента независимы в силу независимости выбора параметров $x_i$, $q_i$ и $u_i$. В результате такой процедуры получим четыре множества по $s$ точек в каждом.

Вероятность того, что в обоих множествах, полученных в результате проведения одной процедуры, найдутся решения поставленной перед противником $\mathcal{A}$ задачи S-QPCCDH$_{s,\mathcal{D}}$, оценивается снизу величиной $n\varepsilon^3/64$ (рассуждения совпадают с теми, что приведены в доказательстве теоремы 2). Поскольку при проведении процедур параметры выбираются независимо, вероятность того, что в каждом из четырёх полученных множеств найдётся правильное решение, не меньше $(n\varepsilon^3/64)^2$. Полученные множества обозначим через $S_1$, $S_1'$, $S_2$ и $S_2'$.

Если противник $\mathcal{A}$ успешно решил обе задачи при выполнении одной из процедур, то в множествах $S_i$ и $S_i'$ присутствуют точки $K_i$ и $K_i'$, для которых выполнены следующие соотношения:

$$K_i = \mathrm{CDH}(X_i' - r_iQ_i', \underbrace{Y_i - r_iQ_i'}_{Y_i'}) = \mathrm{CDH}(X_i, Y_i') = 2\mathrm{CDH}(R, Y_i') + x_iY_i',$$

$$\begin{gathered} K_i' = \mathrm{CDH}(X_i' - r_i'Q_i', Y_i - r_i'Q_i') = \mathrm{CDH}(X_i - (r_i' - r_i)Q_i', Y_i' - (r_i' - r_i)Q_i') = \\ = \mathrm{CDH}(X_i - Q_i, Y_i' - Q_i) = \mathrm{CDH}(R + x_iP - q_iP, Y_i' - R - q_iP) = \\ = \mathrm{CDH}(R, Y_i') - \mathrm{SCDH}(R) - q_iR + x_iY_i' - x_iR - x_iq_iP - q_iY_i' + qR + q^2P. \end{gathered}$$

Отсюда заключаем, что

$$\mathrm{SCDH}(R) = 1/2 \cdot K_i - K_i' + \underbrace{q_iR - x_i/2 \cdot Y_i' + x_iR + x_iq_iP + q_iY_i' - qR - q^2P}_{C}. \qquad (1)$$

Точка $C$ вычисляется один раз для каждой из процедур. Для каждого из четырёх множеств вычисляется $s$ точек $1/2 \cdot K_i$ или $K_i'$ (в зависимости от рассматриваемого множества). После этого с использованием соотношения 1 формируются два множества по $s^2$ элементов в каждом. Первый элемент из первого такого множества, для которого нашёлся совпадающий с ним элемент во втором множестве, выдаётся в качестве предполагаемого значения $\mathrm{SCDH}(Q)$. Вероятность случайного совпадения оценим сверху величиной $2s^4/q$.

Таким образом, алгоритм решает задачу SCDH с вероятностью успеха не меньше чем $\dfrac{n^2\varepsilon^6}{2^{12}} - \dfrac{2s^4}{q}$. Поэтому

$$\mathbf{Insec}^{\mathrm{SCDH}}(4t + \Theta + O(s\tau)) \geqslant \frac{n^2\varepsilon^6}{2^{12}} - \frac{2s^4}{q}.$$

Отсюда получаем

$$\mathbf{Insec}^{\text{S-QPCCDH}_{s,\mathcal{D}}}(t) \leqslant \frac{1}{|\mathcal{D}|} + \sqrt[6]{\frac{\mathbf{Insec}^{\mathrm{SCDH}}(4t + \Theta + O(s\tau))}{|\mathcal{D}|^2} + \frac{2^{13}s^4}{|\mathcal{D}|^2q}}.$$

Теорема 3 доказана. ■

Отметим, что в оригинальной работе [9] доказательство схожей теоремы также требует корректировки.

## 5. Оценка стойкости SESPAKE$_{KA}$

Докажем нижнюю оценку преобладания в задаче отличения ключа от случайной строки для протокола SESPAKE$_{KA}$.

**Теорема 4.** При $q_{\text{send}} \geqslant 2$ и некотором $t$, достаточном для осуществления одним клиентом одного взаимодействия в соответствии с протоколом, справедливо неравенство

$$\mathbf{Insec}^{AKE}_{\text{SESPAKE}_{KA}}(t, q_{\text{send}}) \geqslant \frac{1}{|\mathcal{D}|} - \frac{1}{q}.$$

***Доказательство.*** Приведём пример противника, решающего задачу отличения ключа от случайной строки для протокола $\text{SESPAKE}_{KA}$. Противник с помощью $\mathcal{O}_{\text{send}}$ посылает некоторый $A_{ID}$ участнику $B$, получая в ответ $(ind, salt)$. Противник выбирает пароль $PW'$ из словаря $\mathcal{D}$ в соответствии с равномерным распределением и вычисляет $Q_{PW'}$, а также элемент $\alpha$ из множества $\{1, \ldots, q-1\}$ в соответствии с равномерным распределением. После этого он вычисляет $u_1$ в соответствии со спецификацией протокола (как если бы он был пользователем $A$) и с помощью запроса к $\mathcal{O}_{\text{send}}$ посылает $u_1$ участнику $B$. Получив его ответ, противник вычисляет ключ сеанса $k'$ в соответствии со спецификацией протокола, используя $PW'$ в качестве пароля.

После этого противник делает запрос к оракулу $\mathcal{O}_{\text{test}}$ на отличение того ключа, который был только что установлен участником $B$. Получая некоторую строку $k$, противник сравнивает её с $k'$, полагает $b' = 1$, если они равны, и $b' = 0$ в противном случае и возвращает $b'$ в качестве результата.

Если $k$ является ключом, установленным участником $B$ в ходе описанного взаимодействия с противником, то $\Pr[k' = k] \geqslant \dfrac{1}{|\mathcal{D}|}$ (не учитываем вероятность коллизии хэш-функции при выработке ключа и функции $F$ при выработке $Q_{PW'}$). Пусть $b$ — случайный бит, который выбрал оракул $\mathcal{O}_{\text{test}}$. Справедливо соотношение

$$\Pr[b' = b] = \Pr[b' = 0 | b = 0] \Pr[b = 0] + \Pr[b' = 1 | b = 1] \Pr[b = 1] =$$
$$= \frac{1}{2}\left((1 - 1/q) + \Pr[b = 1 | b' = 1]\right) \geqslant \frac{1}{2}\left((1 - 1/q) + \frac{1}{|\mathcal{D}|}\right) = \frac{1}{2} + \frac{1}{2|\mathcal{D}|} - \frac{1}{2q}.$$

Таким образом, $\mathbf{Insec}^{AKE}_{\text{SESPAKE}_{KA}}(t, q_{\text{send}}) \geqslant \dfrac{1}{|\mathcal{D}|} - \dfrac{1}{q}$. ■

Учитывая, что $|\mathcal{D}| \ll q$, преобладание описанного противника по порядку соответствует $\dfrac{1}{|\mathcal{D}|}$.

Доказательство стойкости (с некоторыми параметрами) протокола $\text{SESPAKE}_{KA}$ проводится в модели со случайным оракулом $\mathcal{O}_H$ (используемую в протоколе хэш-функцию $H$ считаем случайной функцией). Стойкость в конечном итоге базируется на сложности вычислительной задачи Диффи — Хеллмана (CDH) в соответствующей группе.

**Замечание 3.** Приведём комментарии по поводу случайного оракула, которые призваны облегчить понимание некоторых шагов доказательств последующих утверждений. Термин «случайный оракул» широко используется в работах по математической криптографии. В нашем случае моделирование некоторой функции $H : A \to B$ с помощью случайного оракула означает, что в рассуждениях функцию $H$ реализует некоторый вычислитель $\mathcal{O}_H$, который принимает на вход элементы множества $A$, возвращая значения из множества $B$ (этот вычислитель принято называть оракулом). «Случайность» оракула $\mathcal{O}_H$ заключается в том, каким образом он выбирает значение из $B$, которое возвратит в ответ на значение-запрос из $A$. При первом запросе оракул $\mathcal{O}_H$ случайно и равновероятно выбирает функцию из множества всех функций,

определённых на $A$ и принимающих значения из $B$, после чего возвращает значение выбранной функции, вычисленное для поданного ему на вход аргумента. Для всех последующих запросов оракул $\mathcal{O}_H$ просто возвращает соответствующие значения выбранной функции.

Приведённое определение поведения оракула $\mathcal{O}_H$ удобнее интерпретировать следующим эквивалентным образом. Оракул хранит таблицу $T$ размера $|A|$, индексируемую элементами множества $A$, все значения в которой перед первым запросом не определены. При поступлении на вход значения $a \in A$ оракул $\mathcal{O}_H$ либо возвращает значение $T(a)$, если оно определено, либо выбирает случайно и равновероятно значение $b \in B$, устанавливает $T(a) = b$ и возвращает $b$ в качестве ответа. Таким образом, оракул фактически определяет реализуемую им функцию не единовременно, а в процессе ответов на запросы.

Рассмотрим задачу распознавания двух сценариев в следующем эксперименте. Пусть $\mathcal{O}_1$ и $\mathcal{O}_2$ — случайные оракулы, реализующие отображения $A \to B$. Противник может делать запросы вида $(i, a)$, $i \in \{1, 2\}$, $a \in A$, к оракулу $\mathcal{O}$, который возвращает элемент множества $B$, используя при этом оракулов $\mathcal{O}_1$ и $\mathcal{O}_2$. Перед началом работы оракул $\mathcal{O}$ случайно и равновероятно выбирает значение $\beta \in \{1, 2\}$. Оракул $\mathcal{O}$ хранит таблицу $T$, которая изначально пуста. При поступлении на вход запроса $(i, a)$ он действует следующим образом:

— если $\beta = 1$, то $\mathcal{O}(i, a) = \mathcal{O}_1(a)$;
— если $\beta = 2$, то если $T(a)$ не определено, то $\mathcal{O}(i, a) = \mathcal{O}_i(a)$ и $T(a) = \mathcal{O}_i(a)$, иначе $\mathcal{O}(i, a) = T(a)$.

Перед противником стоит задача различения двух сценариев: $\beta = 1$ или $\beta = 2$.

Такое подробное описание приведено ради строгости. Неформально, противник должен определить, получает ли он ответы от двух случайных оракулов или от одного, причём он не имеет возможности делать запросы с одинаковым вторым аргументом $a$ к двум разным оракулам (чтобы сравнить реализуемые оракулами функции покоординатно).

Ясно, что при каждом значении $\beta$ противник фактически осуществляет запросы к одному случайному оракулу (во втором случае этот оракул просто определяется несколько нестандартно). Таким образом, он не может получить никакой информации о значении $\beta$. Преобладание противника при решении такой задачи равно нулю независимо от вычислительных ресурсов.

Далее $q_{\text{exec}}$, $q_{\text{send}}$, $q_H$ — количество запросов к оракулам $\mathcal{O}_{\text{exec}}$, $\mathcal{O}_{\text{send}}$, $\mathcal{O}_H$ соответственно, которые совершает противник $\mathcal{A}_{\text{SESPAKE}_{KA}}$. В теореме 5 значения этих параметров могут быть произвольными. На практике ограничение их значений является важнейшим условием для обеспечения стойкости протокола. Подробные комментарии даны в п. 7.1.

**Теорема 5.** Справедливо неравенство

$$\mathbf{Insec}^{AKE}_{\text{SESPAKE}_{KA}}(t, q_H, q_{\text{send}}, q_{\text{exec}}, q_{\text{rev}}) \leqslant$$

$$\leqslant \frac{(2q_{\text{exec}} + q_{\text{send}})^2}{q} + 2q_H \mathbf{Adv}^{\text{CDH}}(t + 2\tau q_{\text{exec}}) + 2q_{\text{send}} \mathbf{Adv}^{\text{S-QPCCDH}_{\mathcal{D},2q_H}}(t + q_{S1}\tau + C),$$

где $C$ — некоторая константа; $\tau$ — трудоёмкость вычисления кратной точки в группе $E$; $q_{S1}$ — количество запросов к оракулу $\mathcal{O}_{\text{send}}$ вида $(ind, salt)$.

***Структура доказательства.*** Пусть $\mathcal{A}_{\text{SESPAKE}_{KA}}$ — противник, для которого $\Pr[\text{SUCC}]$ — вероятность успеха в процедуре, описанной в конце п. 3. Будем исполь-

зовать противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в качестве чёрного ящика, перехватывая его запросы к оракулам $\mathcal{O}_{\text{exec}}$, $\mathcal{O}_{\text{send}}$, $\mathcal{O}_{\text{rev}}$ и $\mathcal{O}_{\text{test}}$, моделируя работу участников в соответствии с протоколом $\text{SESPAKE}_{KA}$. Далее постепенно будем «лишать» этого противника возможности получать истинные ответы оракулов. При этом будем показывать, насколько изменяются от шага к шагу вероятности успеха противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в модифицированных экспериментах. После некоторого количества модификаций получим эксперимент, вероятность успеха в котором равна $1/2$. Таким образом, отклонение вероятности успеха в «чистом» эксперименте от $1/2$ можно будет оценить сверху суммой оценок, полученных при каждой модификации эксперимента.

Опишем качественную структуру получаемой оценки. Модифицируя условия эксперимента, мы фактически лишаем противников определённых возможностей и тем самым исключаем из рассмотрения некоторый класс противников, которые не могут быть успешны в новых условиях (т. е. тех, которые существенно используют заблокированные возможности). Так, класс, который выводится из рассмотрения при рандомизации ответов оракула $\mathcal{O}_{\text{exec}}$, состоит из противников, пассивно прослушивающих канал связи. Основные рассуждения при этом направлены на оценку преобладания «исключаемых» противников. Далее полученная оценка включается в итоговую в качестве одного из слагаемых. Итого, если всё множество $A$ противников подходящим образом разбито, например, на классы $B_1$, $B_2$ и $B_3$, то оценка имеет следующий вид:

$$\mathbf{Adv}(A) = \max\{\mathbf{Adv}(B_1), \mathbf{Adv}(B_2), \mathbf{Adv}(B_3)\} \leqslant C_1 + C_2 + C_3,$$

где $C_i$ — верхняя оценка для $\mathbf{Adv}(B_i)$.

***Доказательство.*** Пусть вероятность успеха противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ равна $\Pr[\text{SUCC}_0]$. Будем использовать противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в качестве чёрного ящика, перехватывая его запросы к оракулам, при этом нас интересует вероятность успеха противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в модифицированном эксперименте. Для обработки его запросов к оракулам будем моделировать подразумеваемое поведение всех возможных участников протокола: случайным образом выбирать для каждого пароль, генерировать случайные точки на этапе выработки ключа и т. п. В этом случае смоделированное множество участников с точки зрения противника не отличается от того, которое используется в «чистом» эксперименте. Поэтому вероятность $\Pr[\text{SUCC}_1]$ равна вероятности $\Pr[\text{SUCC}_0]$.

Далее используем противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в качестве чёрного ящика в эксперименте, отличающемся от предыдущего эксперимента с полноценно смоделированной сетью абонентов. Отличие заключается в том, что будем заканчивать всякую работу и выдавать на выход случайный бит в том случае, если среди точек $\{u_1\}$ и $\{u_2\}$, полученных в результате выполнения противником $\mathcal{A}_{\text{SESPAKE}_{KA}}$ запросов к оракулам $\mathcal{O}_{\text{send}}$ и $\mathcal{O}_{\text{exec}}$ (смоделированных нами), появились две одинаковые точки. Вероятность такого события не превышает $(2q_{\text{exec}} + q_{\text{send}})^2/(2q)$. Поэтому вероятность успеха $\Pr[\text{SUCC}_2]$ в этом эксперименте отличается от $\Pr[\text{SUCC}_1]$ не более чем на $(2q_{\text{exec}} + q_{\text{send}})^2/(2q)$. Данный эксперимент призван освободить дальнейшие рассуждения от противников, которые, например, посылают запросы с одинаковым $u_1$ участнику $B$, восстанавливая потом (с помощью $\mathcal{O}_{\text{rev}}$) установленный им ключ $K_B$, до тех пор, пока не появится ранее передававшийся $u_2$. Если такое событие случается, то ключ для $K_A$ следует установить равным тому $K_B$, который соответствует ранее встреченному $u_2$.

В следующей модификации эксперимента (назовём её $\mathbf{Exp}_3$) будем при запросе, поступившем от $\mathcal{A}_{\text{SESPAKE}_{KA}}$ к оракулу $\mathcal{O}_{\text{exec}}$, в качестве сессионного ключа выдавать значение произвольной неизвестной и недоступной для вычисления противнику

$\mathcal{A}_{\text{SESPAKE}_{KA}}$ случайной функции $H'(u_1, u_2)$. Пусть вероятность успеха в этом эксперименте отличается от вероятности успеха в предыдущем эксперименте на $\triangle P$, то есть $\triangle P = |\Pr[\text{SUCC}_2] - \Pr[\text{SUCC}_1]|$. Поскольку мы считаем, что функция $H$, с помощью которой формируется реальный ключ, является случайной, то различия в поведении противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в этом и предыдущем экспериментах могут наблюдаться лишь в том случае, если он осуществил запрос $\dfrac{m}{q}\,\text{CDH}\,(u_1 + Q_{PW}, u_2 - Q_{PW})$ к оракулу $\mathcal{O}_H$. То есть с вероятностью $\Delta P$ среди $q_H$ запросов к оракулу $\mathcal{O}_H$ фактически присутствует решение задачи CDH для пары $(u_1 + Q_{PW}, u_2 - Q_{PW})$. Этим можно воспользоваться, чтобы решить задачу CDH для произвольных точек $Q_1$ и $Q_2$. Для этого в ответ на запрос к оракулу $\mathcal{O}_{\text{exec}}$ будем в качестве $u_1$ и $u_2$ использовать точки $(Q_1 + \alpha P) - Q_{PW}$ и $(Q_2 + \beta P) + Q_{PW}$. Если в запросах к оракулу $\mathcal{O}_H$ (их всего $q_H$), которые осуществлял противник, есть значение $\text{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$, то

$$\text{CDH}(Q_1, Q_2) = \text{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW}) - \beta Q_1 - \alpha Q_2 - \alpha\beta P.$$

Из этого можно сделать вывод, что $\Delta P$ не превосходит значения $q_H \cdot \mathbf{Insec}^{\text{CDH}}(t + {} + 2\tau q_{\text{exec}} + 3\tau)$. Это означает, что для пассивно прослушивающего канал связи противника задача отличения сессионных ключей от случайных данных не легче задачи CDH.

На следующем шаге эксперимент $\mathbf{Exp}_3$ модифицируется путём рандомизации ответов оракула $\mathcal{O}_{\text{send}}$: на запросы, подразумевающие содержательные с точки зрения криптографии ответы, будем возвращать случайные точки и формировать сессионный ключ с помощью функции $H'$ (новый эксперимент назовём $\mathbf{Exp}_4$). Таким образом, в новом эксперименте противник не получает никакой информации от запросов к оракулам. Поэтому вероятность успеха $\Pr[\text{SUCC}_4]$ в таком эксперименте равна $1/2$.

**Лемма 2.** Справедливо неравенство

$$\left|\Pr[\text{SUCC}_3] - \underbrace{\Pr[\text{SUCC}_4]}_{=1/2}\right| \leqslant q_{\text{send}} \cdot \mathbf{Insec}^{\text{S-QPCCDH}_{|\mathcal{D}|,2q_H}}(t + q_{S1}\tau + C).$$

***Доказательство.*** Чтобы оценить разницу между вероятностями успехов в $\mathbf{Exp}_3$ и $\mathbf{Exp}_4$, построим специальную цепочку из $q_u = q_{u_1} + q_{u_2} + 1$ ($q_{u_1}$, $q_{u_2}$ — количества запросов к $\mathcal{O}_{\text{send}}$, аргументами которых являются точки $u_1$ и $u_2$ соответственно) экспериментов $\mathbf{Hyb}_0, \ldots, \mathbf{Hyb}_{q_{u_1}+q_{u_2}}$, каждый из которых будет чуть больше отличаться от $\mathbf{Exp}_3$ и будет чуть больше похож на $\mathbf{Exp}_4$:

$$\mathbf{Exp}_3 = \mathbf{Hyb}_0 \rightsquigarrow \mathbf{Hyb}_1 \rightsquigarrow \cdots \rightsquigarrow \mathbf{Hyb}_{q_{u_1}+q_{u_2}} = \mathbf{Exp}_4. \tag{2}$$

Далее в экспериментах $\mathbf{Hyb}_j$ будем через $i$ обозначать номер того запроса к оракулу $\mathcal{O}_{\text{send}}$, параметром которого является либо $u_1$, либо $u_2$ (то есть этот номер учитывает только такие запросы).

*Описание эксперимента* $\boldsymbol{Hyb}_j$*:* запросы ко всем оракулам, кроме $\mathcal{O}_{\text{send}}$, обрабатываются так же, как в $\mathbf{Exp}_3$. Запросы к оракулу $\mathcal{O}_{\text{send}}$ обрабатываются следующим образом:

— запрос с номером $1 \leqslant i \leqslant j$ к оракулу $\mathcal{O}_{\text{send}}$ обрабатываем, как в $\mathbf{Exp}_4$;
— запрос с номером $i > j$ к оракулу $\mathcal{O}_{\text{send}}$ обрабатываем, как в $\mathbf{Exp}_3$.

Если $P_j$ — вероятность успеха противника $\mathcal{A}_{\text{SESPAKE}_{KA}}$ в эксперименте $\mathbf{Hyb}_j$, то

$$\left|\Pr[\text{SUCC}_3] - \Pr[\text{SUCC}_4]\right| \leqslant \sum_{j=1}^{q_u} |P_j - P_{j-1}|.$$

Опишем набор противников $D_j$ и $D'_j$, $j = 1, \ldots, q_u$, с помощью которых оценим разность $|P_j - P_{j-1}|$. Будем использовать этих противников для решения задачи S-QPCCDH с некоторыми модификациями, а именно: при описании $D_j$ будем считать, что для полученных на вход точек $W$ и $Q_s$ известны их кратности относительно $P$: $W = wP$ и $Q_s = zP$. Для $D'_j$ такая модификация не рассматривается (в описании этого алгоритма $w$ и $z$ не участвуют).

Противник $D_j$

Через $i$ будем обозначать номер следующего запроса к оракулу $\mathcal{O}_{\text{send}}$ с параметром $u_1$ (пересылка от $A$ к $B$, в ответ на которую при правильном состоянии выдаётся точка $u_2$) или с параметром $u_2$ (пересылка от $B$ к $A$, после которой при правильном состоянии $A$ формирует ключ $K_A$). Эти запросы будем обозначать $S2$ и $S3$. Кроме того, $D_j$ особым образом обрабатывает запросы с параметрами $(ind, salt)$, которые будем обозначать $S1$.

Противник $D_j$ обрабатывает запросы ко всем оракулам, кроме $\mathcal{O}_{\text{send}}$, так же, как в $\mathbf{Exp}_3$. Запросы к оракулам $\mathcal{O}_{\text{send}}$ обрабатываются следующим образом:

— Запрос $S1$:
  1) Если $i \leqslant j$, то $D_j$ выбирает случайный $x^*$, в качестве ответа возвращает $W + x^*P$.
  2) Если $i > j$, то $D_j$ обрабатывает запрос так же, как в $\mathbf{Exp}_3$.

— Запрос $S2$ (параметр $u_1$):
  1) Если $i < j$, то запрос обрабатывается так же, как в $\mathbf{Exp}_4$.
  2) Если $i = j$, то в ответ на запрос возвращается $(B, u_2 = W)$. В качестве ответа на первом шаге задачи S-QPCCDH возвращает $(st, Y = u_1)$, получая в ответ пароль $(st, PW')$. Устанавливает пароль, разделяемый участниками $A$ и $B$, равным $PW'$, и устанавливает ключ

$$K_B = H_{256}((m/q(w - zr') \bmod q)(u_1 - r'Q_s)),$$

     где $r' = F(PW', salt, 2000)$.
  3) Если $i > j$, то запрос обрабатывается так же, как в $\mathbf{Exp}_3$.

— Запрос $S3$ (параметр $u2$):
  1) Если $i < j$, то запрос обрабатывается так же, как в $\mathbf{Exp}_4$.
  2) Если $i = j$, то возвращает $(st, Y = u_2)$ в качестве ответа на первом шаге задачи S-QPCCDH. Получает в ответ $(st, PW')$. Устанавливает $PW'$ в качестве пароля, разделяемого между $A$ и $B$. Устанавливает ключ

$$K_A = H_{256}((m/q(w + x^* - zr') \bmod q)(u_2 - r'Q_s)),$$

     где $r' = F(PW', salt, 2000)$.
  3) Если $i > j$, то запрос обрабатывается так же, как в $\mathbf{Exp}_3$.

Из запросов к оракулу $\mathcal{O}_H$ противник $D_j$ извлекает ключи $K$ (то есть точки, которые умножаются на $m/q$). Для каждой точки $K$ он формирует две точки $K' = K$ и $K'' = K - x^*(Y - rQ_s)$. В качестве ответа $D_j$ выдаёт список из $2q_H$ точек $\{K'_1, K''_1, K'_2, K''_2, \ldots, K'_{q_H}, K''_{q_H}\}$.

Противник $D'_j$

Этот противник отличается от $D_j$ тем, что при обработке запросов $S2$ и $S3$ для установки ключа он использует функцию $H'$, к которой у $\mathcal{A}_{\text{SESPAKE}_{KA}}$ нет доступа.

Отметим два важных обстоятельства, которые позволяют с помощью описанных противников оценить разность $|P_j - P_{j-1}|$.

С точки зрения $\mathcal{A}_{\mathrm{SESPAKE}_{KA}}$ поведение противника $D'_j$, то есть то, как он обрабатывает запросы к оракулам $\mathcal{O}_{\mathrm{send}}$, $\mathcal{O}_{\mathrm{exec}}$ и т. д., полностью совпадает с экспериментом $\mathbf{Hyb}_j$, поскольку «честная» обработка запросов к $\mathcal{O}_{\mathrm{send}}$ начинается лишь с $j+1$-го запроса. Для противника $D_j$ «честная обработка» запросов к $\mathcal{O}_{\mathrm{send}}$ начинается уже с $j$-го запроса. Например, чему бы ни был равен параметр $u_1$ в запросе $S2$, множитель, на который умножается точка $u_1 - r'Q_s$, равен кратности точки, которая была возвращена в качестве ответа, то есть $W$, после снятия «маски» $r'Q_s$. Таким образом, $A$, получив данный ответ, получит тот же ключ, что и $B$, если $A$ изначально (то есть на запрос $S1$) моделировался в соответствии с протоколом.

Поведение противника в экспериментах $D_j$ и $D'_j$ будет отличаться лишь в том случае, если он сделает запрос к оракулу $\mathcal{O}_H$ с параметрами, среди которых будет точка $\mathrm{CDH}(u_1 - r'Q_s, u_2 - r'Q_s)$, где $u_1$ или $u_2$ присутствовали в $j$-м запросе $S2$ или $S3$. Если такой параметр в запросах к $\mathcal{O}_H$ был, то найти $\mathrm{CDH}(W - r'Q_s, Y - r'Q_s)$ можно, так как

$$K_A = \mathrm{CDH}(u_2 - r'Q_s, W + x^*P - r'Q_s) = \mathrm{CDH}(W - r'Q_s, Y - r'Q_s) + x^*(Y - r'Q_s),$$
$$K_B = \mathrm{CDH}(u_1 - r'Q_s, W - r'Q_s) = \mathrm{CDH}(W - r'Q_s, Y - r'Q_s).$$

Учитывая способ формирования выходного списка из $2q_H$ точек противника $D'_j$, можно заключить, что искомое решение в нём найдётся. Таким образом, вероятность того, что среди параметров запросов к $\mathcal{O}_H$ найдётся $\mathrm{CDH}(u_1 - r'Q_s, u_2 - r'Q_s)$ с $u_1, u_2$, фигурировавшими в $j$-м запросе $S2$ или $S3$, можно оценить сверху наибольшей вероятностью успеха при решения задачи S-QPCCDH противником, работающим за время $t + q_{S1}\tau + C$, где $q_{S1}$ — количество запросов $S1$ к участнику $A$; $C$ — некоторая фиксированная константа. Таким образом, выполнено следующее неравенство:

$$\left|\Pr[\mathrm{SUCC}_3] - \underbrace{\Pr[\mathrm{SUCC}_4]}_{=1/2}\right| \leqslant \sum_{j=1}^{q_u} |P_j - P_{j-1}| \leqslant q_{\mathrm{send}} \cdot \mathbf{Insec}^{\text{S-QPCCDH}_{|\mathcal{D}|,2q_H}}(t + q_{S1}\tau + C).$$

Лемма 2 доказана. ■

Учитывая все полученные оценки, можно заключить, что выполнено неравенство

$$\left|\Pr[\mathrm{SUCC}_0] - \frac{1}{2}\right| \leqslant \frac{(2q_{\mathrm{exec}} + q_{\mathrm{send}})^2}{2q} + q_H \mathbf{Insec}^{\mathrm{CDH}}(t + 2\tau q_{\mathrm{exec}}) +$$
$$+ q_{\mathrm{send}} \cdot \mathbf{Insec}^{\text{S-QPCCDH}_{|\mathcal{D}|,2q_H}}(t + q_{S1}\tau + C).$$

Из этого соотношения непосредственно следует доказываемое неравенство для $\mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t, q_H, q_{\mathrm{send}}, q_{\mathrm{exec}}, q_{\mathrm{rev}})$. Теорема 5 доказана. ■

## 6. Оценка стойкости SESPAKE

### 6.1. Используемые подзадачи

Для обоснования стойкости протокола SESPAKE помимо модели, описанной в п. 3, введём несколько дополнительных задач, которые выступят в роли промежуточных. Отличия между всеми используемыми моделями заключаются в том, как работает оракул $\mathcal{O}_{\mathrm{auth}}$.

Отметим особенность промежуточных задач. Оракулы, с помощью которых формулируется угроза для $\mathrm{SESPAKE}_{KA}$ и SESPAKE, возвращают «честные» данные при $b = 1$ и случайные при $b = 0$. В задачах 1HMAC и 2HMAC мы поменяли смысловую

нагрузку значений, возвращаемых оракулом $\mathcal{O}_{\text{auth}}$: в этих задачах «более честные» данные возвращаются при $b = 0$, а не при $b = 1$. Это сделано для удобства оценки основных параметров.

### 6.2. Задача 1HMAC

В этой задаче противник имеет доступ ко всем оракулам, к которым имеет доступ противник в модели для протокола SESPAKE. Отличие состоит в том, что оракул $\mathcal{O}_{\text{auth}}$ возвращает не пару строк, а либо $M'_A$ (при $b = 1$), либо $M_A$ (при $b = 0$).

### 6.3. Задача 2HMAC

В задаче 2HMAC оракул $\mathcal{O}_{\text{auth}}$ в ответ на запрос возвращает противнику либо строки $(M'_A, M')$ (при $b = 1$), либо строки $(M_A, M'')$ (при $b = 0$). Строки определяются следующим образом:

1) $M_A = \text{HMAC}_{K_{A,i}}(T_A||IDS_{A,i})$;
2) $M'_A = \text{HMAC}_{K'}(T_A||IDS_{A,i})$ и $M' = \text{HMAC}_{K'}(T_B||IDS_{A,i})$, где $K' \xleftarrow{\mathcal{U}} V_{256}$;
3) $M'' = \text{HMAC}_{K''}(T_B||IDS_{A,i})$, где $K'' \xleftarrow{\mathcal{U}} V_{256}$.

### 6.4. Модель противника «с предупреждением»

Заметим, что преобладание $\mathbf{Adv}(\mathcal{A})$ некоторого противника, решающего задачу распознавания для бита $b$, можно преобразовать следующим образом:

$$\mathbf{Adv}(\mathcal{A}) = 2\Pr[\text{SUCC}] - 1 = 2\left[\frac{1}{2}\Pr[\mathcal{A} = 1|b = 1] + \frac{1}{2}(1 - \Pr[\mathcal{A} = 1|b = 0])\right] - 1 =$$
$$= \Pr[\mathcal{A} = 1|b = 1] - \Pr[\mathcal{A} = 1|b = 0].$$

Далее в основном будем оценивать именно таким образом преобразованную величину.

**Теорема 6.** Справедливо соотношение

$$\mathbf{Insec}^{\text{1HMAC}}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}, q_{\text{check}}) \leqslant$$
$$\leqslant \mathbf{Insec}^{AKE}_{\text{SESPAKE}_{KA}}(t + q_{\text{HMAC}} + q_{\text{check}}, q_{\text{exec}}, q_{\text{send}}, q_H) + O\left(\frac{1}{2^n}\right).$$

***Доказательство.*** Пусть $\mathcal{A}$ — противник, решающий задачу 1HMAC. Опишем противника $\mathcal{A}'$, реализующего угрозу отличения ключа для протокола $\text{SESPAKE}_{KA}$ и использующего $\mathcal{A}$ в качестве чёрного ящика. $\mathcal{A}'$ транслирует все запросы к имеющимся в его распоряжении оракулам ($\mathcal{O}_{\text{exec}}$, $\mathcal{O}_{\text{send}}$, $\mathcal{O}_{\text{rev}}$ и $\mathcal{O}_H$) и их ответы без изменений. При запросе к оракулу $\mathcal{O}_{\text{check}}$ противник $\mathcal{A}'$ делает запрос к $\mathcal{O}_{\text{rev}}$, получая ключ того оракула, для которого $\mathcal{A}$ хочет получить значение аутентификационных данных. После этого $\mathcal{A}'$ возвращает $\mathcal{A}$ «честное» значение запрашиваемых им данных, используя полученный ключ и оракул $\mathcal{O}_{\text{HMAC}}$. Если $\mathcal{A}$ делает запрос к оракулу $\mathcal{O}_{\text{auth}}$, то $\mathcal{A}'$ делает запрос к $\mathcal{O}_{\text{test}}$, получая либо случайный ключ $K'$, либо истинный ключ $K$, который хранит тестируемый оракул. После этого $\mathcal{A}'$ передает $\mathcal{A}$ значение HMAC, вычисленное на полученном ключе. В качестве ответа $\mathcal{A}'$ возвращает значение, противоположное тому, которое вернул $\mathcal{A}$.

Из описания видно, что $\mathcal{A}'$ полностью моделирует эксперимент, в условиях которого работает противник $\mathcal{A}$. Преобладание $\mathbf{Adv}^{AKE}_{\text{SESPAKE}_{KA}}(\mathcal{A}')$ отличается от преобладания $\mathbf{Adv}^{\text{1HMAC}}(\mathcal{A})$ не более чем на величину порядка $2^{-n}$, что объясняется тем, что значение HMAC на случайно выбранном ключе может совпасть со значением HMAC на сессионном ключе атакуемого оракула. Поэтому

$$\mathbf{Adv}^{AKE}_{\text{SESPAKE}_{KA}}(\mathcal{A}') \geqslant \mathbf{Adv}^{\text{1HMAC}}(\mathcal{A}) - O\left(\frac{1}{2^n}\right).$$

Поскольку $\mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}}, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) \geqslant \mathbf{Adv}^{AKE}_{\mathrm{SESPAKE}_{KA}}(\mathcal{A}')$, то

$$\mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A}) = \mathbf{Insec}^{1\mathrm{HMAC}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}}) \leqslant$$
$$\leqslant \mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}}, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) + O\left(\frac{1}{2^n}\right).$$

Теорема 6 доказана. ■

**Теорема 7.** Справедливо соотношение

$$\mathbf{Insec}^{2\mathrm{HMAC}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}}) \leqslant$$
$$\leqslant \mathbf{Insec}^{1\mathrm{HMAC}}(t + 1, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}} + 1) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}}.$$

***Доказательство.*** Пусть $\mathcal{A} \in \mathcal{A}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}})$ — противник, решающий задачу 2HMAC с преобладанием $\mathbf{Insec}^{2\mathrm{HMAC}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}})$. Построим противника $\mathcal{A}'$, использующего $\mathcal{A}$ в качестве чёрного ящика и решающего задачу 1HMAC.

$\mathcal{A}'$ ничего не меняет во взаимодействии $\mathcal{A}$ со всеми доступными ему оракулами, кроме $\mathcal{O}_{\mathrm{auth}}$. Если $\mathcal{A}$ делает запрос к оракулу $\mathcal{O}_{\mathrm{auth}}$, то $\mathcal{A}'$ делает запрос к своему оракулу $\mathcal{O}_{\mathrm{auth}}$ и пересылает противнику $\mathcal{A}$ его ответ и значение $M''$ функции HMAC, вычисленное на случайно выбранном ключе $K''$. В итоге противник $\mathcal{A}$ фактически отличает следующие ситуации: он получил либо пару $(M_A, M'')$, либо пару $(M'_A, M'')$. Отличие от «честного» с точки зрения $\mathcal{A}$ эксперимента состоит в том, что пара $(M'_A, M'')$ порождена с помощью двух независимо выбранных случайных ключей.

Справедливо соотношение

$$\mathbf{Adv}^{1\mathrm{HMAC}}(\mathcal{A}') = \Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M'_A, M'')] - \Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M_A, M'')] =$$
$$= \Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M'_A, M'')] - \Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M'_A, M')] +$$
$$+ \underbrace{\Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M'_A, M')] - \Pr[\mathcal{A} = 1 | \mathcal{A} \leftarrow (M_A, M'')]}_{=\mathbf{Adv}^{2\mathrm{HMAC}}(\mathcal{A})} \geqslant -\frac{2q_{\mathrm{HMAC}}}{2^n} + \mathbf{Adv}^{2\mathrm{HMAC}}(\mathcal{A}).$$

Действительно, первая из двух оцениваемых разностей не превосходит преобладания наилучшего противника, работающего с параметрами противника $\mathcal{A}$ и решающего задачу отличения пары выходов случайной функции, полученных на одном случайном ключе, от пары выходов случайной функции, полученных на двух независимых случайных ключах. Противник $\mathcal{A}$ сможет это сделать лишь в том случае, если среди его запросов к $\mathcal{O}_{\mathrm{HMAC}}$ присутствовал либо ключ $K'$, либо ключ $K''$, один из которых $\mathcal{O}_{\mathrm{auth}}$ использовал в процессе работы. Поскольку оба ключа выбираются случайно, то вероятность такого события не превосходит $2q_{\mathrm{HMAC}}/2^n$.

Отметим также, что противник $\mathcal{A}'$ в процессе работы дополнительно порождает случайный ключ $K''$ и делает дополнительный запрос к оракулу $\mathcal{O}_{\mathrm{HMAC}}$. В итоге получаем

$$\mathbf{Adv}^{2\mathrm{HMAC}}(\mathcal{A}') \leqslant \mathbf{Adv}^{1\mathrm{HMAC}}(\mathcal{A}') + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} =$$
$$= \mathbf{Insec}^{1\mathrm{HMAC}}(t + 1, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H, q_{\mathrm{HMAC}}, q_{\mathrm{check}} + 1) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}}.$$

Теорема 7 доказана. ■

**Теорема 8.** Справедливо соотношение

$$\mathbf{Insec}_{\mathrm{SESPAKE}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_{\mathrm{check}}, q_H, q_{\mathrm{HMAC}}) \leqslant$$
$$\leqslant 2 \cdot \mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}} + 2, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} + O\left(\frac{1}{2^n}\right).$$

***Доказательство.*** Пусть $\mathcal{A} \in \mathcal{A}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_{\mathrm{check}}, q_H, q_{\mathrm{HMAC}})$ — произвольный противник, реализующий угрозу для протокола SESPAKE с преобладанием $\mathbf{Adv}_{\mathrm{SESPAKE}}(\mathcal{A})$. Опишем противника $\mathcal{A}'$, который использует $\mathcal{A}$ для реализации угрозы для протокола $\mathrm{SESPAKE}_{KA}$. $\mathcal{A}'$ использует $\mathcal{A}$ так же, как это описано в теореме 6. Таким образом, если оракул $\mathcal{O}_{\mathrm{test}}$ вернул случайный ключ $K'$, то противник $\mathcal{A}$ в ответ на запрос к $\mathcal{O}_{\mathrm{auth}}$ получит пару $(M'_A, M') = (\mathrm{HMAC}_{K'}(T_A||IDS_{A,i}), \mathrm{HMAC}_{K'}(T_B||IDS_{A,i}))$, иначе — пару $(M_A, M) = (\mathrm{HMAC}_{K_{A,i}}(T_A||IDS_{A,i}), \mathrm{HMAC}_{K_{A,i}}(T_B||IDS_{A,i}))$. Поскольку в качестве ответа $\mathcal{A}'$ возвращает то, что вернул $\mathcal{A}$, то для преобладания $\mathbf{Adv}(\mathcal{A}')$ выполнено соотношение

$$\mathbf{Adv}(\mathcal{A}') = \Pr[\mathcal{A} = 1|(M_A, M)] - \Pr[\mathcal{A} = 1|(M'_A, M')] =$$
$$= \underbrace{\Pr[\mathcal{A} = 1|(M_A, M)] - \Pr[\mathcal{A} = 1|(M_A, M'')]}_{S} +$$
$$+ \underbrace{\Pr[\mathcal{A} = 1|(M_A, M'')] - \Pr[\mathcal{A} = 1|(M'_A, M')]}_{T},$$

где $M'' = \mathrm{HMAC}_{K''}(T_B||IDS_{A,i})$ и $K'' \xleftarrow{\mathcal{U}} V_{256}$. Заметим, что $S$ равно $\mathbf{Adv}_{\mathrm{SESPAKE}}(\mathcal{A})$.

Оценим величину $T$, то есть преобладание противника $\mathcal{A}$ в задаче отличения следующих ответов $(M_A, M'')$ и $(M'_A, M')$ оракула $\mathcal{O}_{\mathrm{auth}}$. Оценим сверху величину $-T$, получив тем самым нижнюю оценку для $T$. Для $-T$ справедливо соотношение $-T = \Pr[\mathcal{A} = 1|(M'_A, M')] - \Pr[\mathcal{A} = 1|(M_A, M'')] \leqslant \mathbf{Insec}^{\mathrm{2HMAC}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_{\mathrm{check}}, q_H, q_{\mathrm{HMAC}})$, так как $\mathcal{A}$ — конкретный противник из множества $\mathcal{A}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_{\mathrm{check}}, q_H, q_{\mathrm{HMAC}})$, а стоящее справа в неравенстве преобладание характеризует наиболее успешного при решении задачи 2HMAC противника из этого множества. Используя теоремы 6 и 7, получаем неравенство

$$T \geqslant -\mathbf{Insec}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}} + 2, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) - \frac{q_{\mathrm{HMAC}}}{2^{n-1}} - O\left(\frac{1}{2^n}\right).$$

В итоге имеем

$$\mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}}, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) \geqslant \mathbf{Adv}^{AKE}_{\mathrm{SESPAKE}_{KA}}(\mathcal{A}') \geqslant$$
$$\geqslant \mathbf{Adv}_{\mathrm{SESPAKE}}(\mathcal{A}) - \mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}} + 2, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) - \frac{q_{\mathrm{HMAC}}}{2^{n-1}} - O\left(\frac{1}{2^n}\right).$$

Следовательно,

$$\mathbf{Adv}_{\mathrm{SESPAKE}}(\mathcal{A}) = \mathbf{Insec}_{\mathrm{SESPAKE}}(t, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_{\mathrm{check}}, q_H, q_{\mathrm{HMAC}}) \leqslant$$
$$\leqslant 2 \cdot \mathbf{Insec}^{AKE}_{\mathrm{SESPAKE}_{KA}}(t + q_{\mathrm{HMAC}} + q_{\mathrm{check}} + 2, q_{\mathrm{exec}}, q_{\mathrm{send}}, q_H) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} + O\left(\frac{1}{2^n}\right).$$

Теорема 8 доказана. ■

Суть оценки, доказанной в теореме 8, состоит в том, что добавление к протоколу $\text{SESPAKE}_{KA}$ этапа подтверждения ключа даёт противнику лишь критерий для проверки сеансового ключа, но не даёт никаких существенно новых возможностей для атаки на пароль. Это объясняется тем, что слагаемые $q_{\text{HMAC}}/2^{n-1}$ и $O(1/2^n)$ пренебрежимо малы по сравнению с первым слагаемым. Таким образом, наилучшим способом для атаки на пароль является его угадывание с активным вмешательством во взаимодействие по каналу, о чём говорит то, что первое слагаемое по порядку совпадает с $q_{\text{send}}/|\mathcal{D}|$.

### 6.5. Модель противника «без предупреждения»

Ранее рассмотрено преобладание $\mathbf{Adv}_{\text{SESPAKE}}$ противника $\mathcal{A}_{\text{SESPAKE}}$, который с помощью запросов к оракулу $\mathcal{O}_{\text{check}}$ предупреждает, по отношению к каким сессиям он не будет пытаться осуществить угрозу. Теорема 8 относится именно к этой модели противника. Переобозначим его преобладание через $\mathbf{Adv}_{\text{SESPAKE},w}$.

Теперь рассмотрим преобладание $\mathbf{Adv}_{\text{SESPAKE}}$ противника $\mathcal{A}_{\text{SESPAKE}}$, который может делать запросы к оракулу $\mathcal{O}_{\text{auth}}$ для любого участника протокола, независимо от того, сделан ли по отношению к нему запрос к оракулу $\mathcal{O}_{\text{check}}$ или нет. Условия, в которых работает противник, ничем не отличаются, кроме формата ответа на запрос к оракулу $\mathcal{O}_{\text{auth}}$:

— оракул $\mathcal{O}_{\text{auth}}$ принимает на вход пару $(A, i)$, случайно и равновероятно выбирает значение $b \in \{0, 1\}$ и в зависимости от $b$ возвращает либо $M'$ (если $b = 0$), либо $M_B$ (если $b = 1$). При этом $M' = \text{HMAC}_{K'}(T_B||IDS_{A,i})$, где $K' \xleftarrow{U} V_{256}$, $M_B = \text{HMAC}_{K_{A,i}}(T_B||IDS_{A,i})$.

Противник может делать запрос к оракулу $\mathcal{O}_{\text{auth}}$ только для тех оракулов, которые допускают тестирование. Оракулом, допускающим тестирование, называется такой оракул $\mathcal{O}_A^i$, для которого противник не получил значение $M_B$ тривиальным образом, то есть не сделал запрос к $\mathcal{O}_{\text{check}}$ для оракула, разделяющего с $\mathcal{O}_A^i$ общий ключ. Оценим преобладание $\mathbf{Insec}_{\text{SESPAKE}}$.

**Теорема 9.** Справедливо соотношение

$$\begin{gathered}\mathbf{Insec}_{\text{SESPAKE}}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}) \leqslant \\ \leqslant q_{\text{send}} \cdot \mathbf{Insec}_{\text{SESPAKE},w}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}) + \\ +2q_H \cdot \mathbf{Insec}^{\text{CDH}}(t + \tau(2q_{\text{exec}} + q_{\text{send}})) + O\left(\frac{q_H + q_{\text{HMAC}}}{2^n}\right).\end{gathered}$$

***Доказательство.*** Пусть $\mathcal{A}_{SESPAKE}$ — противник, действующий в описанной модели, с параметрами $t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}$ и преобладанием $\mathbf{Insec}_{\text{SESPAKE}}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}})$. Введём на его основе двух вспомогательных противников $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ и $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$.

Будем называть сессию Send-сессией, если она инициирована посредством запроса к оракулу $\mathcal{O}_{\text{send}}$, и Exec-сессией — к оракулу $\mathcal{O}_{\text{exec}}$. Через $s_{\text{auth}}$ будем обозначать ту сессию, по отношению к участнику которой противник осуществил запрос к оракулу $\mathcal{O}_{\text{auth}}$. Тот факт, что сессия $s_{\text{auth}}$ является Send-сессией (Exec-сессией), будем обозначать через $s_{\text{auth}} \in \text{Send}$ ($s_{\text{auth}} \in \text{Exec}$).

Противник $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ использует противника $\mathcal{A}_{\text{SESPAKE}}$ в качестве чёрного ящика и работает как транслятор между ним и оракулами, вплоть до обращения к оракулу $\mathcal{O}_{\text{auth}}$. Если зафиксировано обращение противника $\mathcal{A}_{\text{SESPAKE}}$ к оракулу $\mathcal{O}_{\text{auth}}$ для

Send-сессии ($s_{\text{auth}} \in \text{Send}$), то $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ выдаёт тот же бит, что и $\mathcal{A}_{\text{SESPAKE}}$; если сделано обращение к Exec-сессии ($s_{\text{auth}} \in \text{Exec}$), то $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ независимо от выхода оракула $\mathcal{O}_{\text{auth}}$ выдаёт случайный бит. Через $\text{SUCC}_{\text{SESPAKE}}$, $\text{SUCC}_{\text{SESPAKE}_{\text{send}}}$ и $\text{SUCC}_{\text{SESPAKE}_{\text{exec}}}$ будем обозначать событие, заключающееся в успешности решения задачи ложной аутентификации противником $\mathcal{A}_{\text{SESPAKE}}$, $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ и $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ соответственно.

Верно следующее равенство:

$$\begin{gathered}\Pr\left[\,\text{SUCC}_{\text{SESPAKE}_{\text{send}}}\,\right] = \\ = \Pr\left[\,\text{SUCC}_{\text{SESPAKE}} \mid s_{\text{auth}} \in \text{Send}\,\right] \cdot \Pr\left[\,s_{\text{auth}} \in \text{Send}\,\right] + \frac{1}{2} \cdot \Pr\left[\,s_{\text{auth}} \in \text{Exec}\,\right],\end{gathered} \tag{3}$$

где $\Pr\left[\,s_{\text{auth}} \in \text{Send}\,\right]$ ($\Pr\left[\,s_{\text{auth}} \in \text{Exec}\,\right]$) — вероятность того, что противник сделал запрос к оракулу $\mathcal{O}_{\text{auth}}$ по отношению к Send-сессии (Exec-сессии). Имеем в виду, что $\Pr\left[\,s_{\text{auth}} \in \text{Send}\,\right] + \Pr\left[\,s_{\text{auth}} \in \text{Exec}\,\right] = 1$.

Противник $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ работает аналогичным образом, только при запросе $\mathcal{A}_{\text{SESPAKE}}$ для Exec-сессии он выдаёт тот же бит, что и $\mathcal{A}_{\text{SESPAKE}}$, а при запросе к Send-сессии — случайный бит:

$$\begin{gathered}\Pr\left[\,\text{SUCC}_{\text{SESPAKE}_{\text{exec}}}\,\right] = \\ = \Pr\left[\,\text{SUCC}_{\text{SESPAKE}} \mid s_{\text{auth}} \in \text{Exec}\,\right] \cdot \Pr\left[\,s_{\text{auth}} \in \text{Exec}\,\right] + \frac{1}{2} \cdot \Pr\left[\,s_{\text{auth}} \in \text{Send}\,\right].\end{gathered} \tag{4}$$

Оценим преобладание противника $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$.

**Лемма 3.** Справедливо неравенство

$$\mathbf{Adv}\left(\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}\right) \leqslant 2q_H \cdot \mathbf{Insec}^{\text{CDH}}(t + \tau(2q_{\text{exec}} + q_{\text{send}}) + 3\tau) + O\left(\frac{q_H + q_{\text{HMAC}}}{2^n}\right).$$

***Доказательство.*** Пусть $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ — противник с параметрами $t$, $q_{\text{exec}}$, $q_{\text{send}}$, $q_H$, $q_{\text{HMAC}}$, решающий задачу ложной аутентификации с преобладанием $\mathbf{Adv}\left(\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}\right)$. Построим на его основе противника $\mathcal{A}_{\text{CDH}}$, решающего вычислительную задачу CDH, и оценим снизу его вероятность успеха $\mathbf{Adv}\left(\mathcal{A}_{\text{CDH}}\right)$.

Пусть противнику $\mathcal{A}_{\text{CDH}}$ необходимо решить задачу CDH для произвольных точек $Q_1$ и $Q_2$. Будем использовать противника $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ в качестве чёрного ящика.

Противник $\mathcal{A}_{\text{CDH}}$ запускает противника $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ и симулирует выполнение протокола, где в ответ на запрос к оракулу $\mathcal{O}_{\text{exec}}$ в качестве $u_1$ и $u_2$ использует точки $(Q_1 + \alpha P) - Q_{PW}$ и $(Q_2 + \beta P) + Q_{PW}$, $\alpha$ и $\beta$ — случайно выбранные значения. При этом в качестве сессионного ключа при запросах противника $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ к оракулам $\mathcal{O}_{\text{check}}$ и $\mathcal{O}_{\text{auth}}$ противник $\mathcal{A}_{\text{CDH}}$ использует значение произвольной неизвестной и недоступной для вычисления противнику $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$ случайной функции $H'(u_1, u_2)$. Затем он случайно выбирает один из $q_H$ запросов к оракулу $\mathcal{O}_H$ (обозначим этот запрос через $R$), которые осуществлял противник $\mathcal{A}_{\text{SESPAKE}_{\text{exec}}}$, и вычисляет значение $T$ следующим образом:

$$T = R - \beta Q_1 - \alpha Q_2 - \alpha\beta P,$$

где соответствующие $\alpha$ и $\beta$ выбираются по значениям точек $u_1$ и $u_2$, исходя из запросов к оракулу $\mathcal{O}_{\text{HMAC}}$ и ответов на запросы к оракулу $\mathcal{O}_H$. Вероятность того, что для разных $\alpha$ и $\beta$ мы получим одинаковые ключи, равна $O\left(2^{-n}\right)$.

Если противник $\mathcal{A}_{\mathrm{SESPAKE_{exec}}}$ правильно вычислил значение $R = \mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$, то $T$ равно искомому значению $\mathrm{CDH}(Q_1, Q_2)$. Тогда для построенного противника преобладание $\mathbf{Adv}\,(\mathcal{A}_{\mathrm{CDH}})$ равно $\Pr\,[T = \mathrm{CDH}(Q_1, Q_2)]$.

Обозначим через $A$ событие, при котором среди запросов к оракулу $\mathcal{O}_H$ есть значение, из которого можно получить искомое $\mathrm{CDH}(Q_1, Q_2)$ по указанной формуле. Тогда верно следующее соотношение:

$$\mathbf{Insec}^{\mathrm{CDH}}(t + \tau(2q_{\mathrm{exec}} + q_{\mathrm{send}}) + 3\tau) \geqslant \mathbf{Adv}\,(\mathcal{A}_{\mathrm{CDH}}) = \frac{1}{q_H} \cdot \Pr\,[\ A\ ]\,.$$

Рассмотрим преобладание $\mathbf{Adv}\,(\mathcal{A}_{\mathrm{SESPAKE_{exec}}}) = 2\Pr\,[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}}\,] - 1$ противника $\mathcal{A}_{\mathrm{SESPAKE_{exec}}}$:

$$\begin{aligned}\Pr\,[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}}\,] = \underbrace{\Pr\,[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}} \mid A\,]}_{\leqslant 1} \cdot \Pr\,[\,A\,] + \\ + \Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}} \mid \overline{A}\,\right] \underbrace{(1 - \Pr\,[\,A\,])}_{\leqslant 1} \leqslant \Pr\,[\,A\,] + \Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}} \mid \overline{A}\,\right].\end{aligned} \tag{5}$$

Рассмотрим случай, когда противнику необходимо угадать значение бита $b$ при условии, что среди запросов к оракулу $\mathcal{O}_H$ нет значения $\mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$, и оценим вероятность его успеха $\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}} \mid \overline{A}\,\right]$.

В силу случайности функции $H$, данные, полученные в ответ на запрос к оракулу $\mathcal{O}_H$ со значениями, не равными $\mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$, случайны и поэтому не несут никакой информации о настоящих сессионных ключах. Следовательно, в этом случае перед противником встает следующая задача: исходя только из ответов на запросы к оракулам $\mathcal{O}_{\mathrm{check}}$ и $\mathcal{O}_{\mathrm{auth}}$, определить, на каком ключе вычислено значение HMAC. Так как в такой ситуации для противника и при $b = 0$, и при $b = 1$ ключи выглядят одинаково случайными, то угадать значение бита с вероятностью, равной единице с точностью до $O\,(2^{-n})$, он может только с помощью следующей тактики: случайно выбрать ключ, вычислить на нём значения HMAC и сравнить полученные значения с ответами на запросы к оракулам $\mathcal{O}_{\mathrm{check}}$ и $\mathcal{O}_{\mathrm{auth}}$. Если вычисленные на выбранном ключе значения HMAC совпали с ответом на запрос и к оракулу $\mathcal{O}_{\mathrm{check}}$, и к оракулу $\mathcal{O}_{\mathrm{auth}}$, то противник выдаёт $b' = 1$, и наоборот, если вычисленные на выбранном ключе значения HMAC совпали с ответом на запрос либо к оракулу $\mathcal{O}_{\mathrm{check}}$, либо к оракулу $\mathcal{O}_{\mathrm{auth}}$, то выдаёт $b' = 0$.

Приближённое равенство вероятности успеха единице (с точностью до $O\,(2^{-n})$) объясняется вероятностью ошибиться за счёт случайности функции $H$, либо когда значения HMAC, вычисленные на случайно выбранном ключе, совпали и с ответом на запрос к оракулу $\mathcal{O}_{\mathrm{check}}$ $(M_A)$, и с ответом на запрос к оракулу $\mathcal{O}_{\mathrm{auth}}$ при $b = 0$ $(M')$ (противник ошибочно выдаст $b' = 1$), либо когда при $b = 1$ значения HMAC, вычисленные на случайно выбранном ключе, совпали только с одним из значений $M_A$ или $M_B$ (противник ошибочно выдаст $b' = 0$). Тогда

$$\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE_{exec}}} \mid \overline{A}\,\right] = \frac{1}{2} + O\Big(\frac{q_H + q_{\mathrm{HMAC}}}{2^n}\Big).$$

Отсюда с учётом (5) следует, что

$$2\Pr\,[\,A\,] \geqslant \mathbf{Adv}\,(\mathcal{A}_{\mathrm{SESPAKE_{exec}}}) - O\Big(\frac{q_H + q_{\mathrm{HMAC}}}{2^n}\Big).$$

Следовательно,

$$\mathbf{Insec}^{\mathrm{CDH}}(t+\tau(2q_{\mathrm{exec}}+q_{\mathrm{send}})+3\tau)\geqslant\frac{1}{2q_H}\left(\mathbf{Adv}\left(\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{exec}}}\right)-O\Big(\frac{q_H+q_{\mathrm{HMAC}}}{2^n}\Big)\right).$$

Лемма 3 доказана. ■

По формуле полной вероятности верно следующее равенство:

$$\begin{aligned}\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\,\right]=\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\mid s_{\mathrm{auth}}\in\mathrm{Send}\,\right]\cdot\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Send}\,\right]+\\+\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\mid s_{\mathrm{auth}}\in\mathrm{Exec}\,\right]\cdot\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Exec}\,\right].\end{aligned}\tag{6}$$

Преобразуем выражения (3) и (4):

$$\begin{aligned}\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\mid s_{\mathrm{auth}}\in\mathrm{Send}\,\right]\cdot\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Send}\,\right]=\\=\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}_{\mathrm{send}}}\,\right]-\frac{1}{2}\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Exec}\,\right];\end{aligned}\tag{7}$$

$$\begin{aligned}\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\mid s_{\mathrm{auth}}\in\mathrm{Exec}\,\right]\cdot\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Exec}\,\right]=\\=\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}_{\mathrm{exec}}}\,\right]-\frac{1}{2}\Pr\left[\,s_{\mathrm{auth}}\in\mathrm{Send}\,\right].\end{aligned}\tag{8}$$

Подставив (7) и (8) в правую часть равенства (6), окончательно получим

$$\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}_{\mathrm{send}}}\,\right]=\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}}\,\right]-\Pr\left[\,\mathrm{SUCC}_{\mathrm{SESPAKE}_{\mathrm{exec}}}\,\right]+\frac{1}{2},$$

или

$$\begin{aligned}\mathbf{Insec}_{\mathrm{SESPAKE}_{\mathrm{send}}}(t,q_{\mathrm{exec}},q_{\mathrm{send}},q_H,q_{\mathrm{HMAC}})=\\=\mathbf{Insec}_{\mathrm{SESPAKE}}(t,q_{\mathrm{exec}},q_{\mathrm{send}},q_H,q_{\mathrm{HMAC}})-\mathbf{Insec}_{\mathrm{SESPAKE}_{\mathrm{exec}}}(t,q_{\mathrm{exec}},q_{\mathrm{send}},q_H,q_{\mathrm{HMAC}}).\end{aligned}$$

Воспользовавшись леммой 3, приходим к следующей оценке:

$$\begin{aligned}\mathbf{Insec}_{\mathrm{SESPAKE}_{\mathrm{send}}}(t,q_{\mathrm{exec}},q_{\mathrm{send}},q_H,q_{\mathrm{HMAC}})\geqslant\mathbf{Insec}_{\mathrm{SESPAKE}}(t,q_{\mathrm{exec}},q_{\mathrm{send}},q_H,q_{\mathrm{HMAC}})-\\-2q_H\cdot\mathbf{Insec}^{\mathrm{CDH}}(t+\tau(2q_{\mathrm{exec}}+q_{\mathrm{send}}))-O\left(\frac{q_H+q_{\mathrm{HMAC}}}{2^n}\right).\end{aligned}$$

Построим противника «с предупреждением» $\mathcal{A}_{\mathrm{SESPAKE},w}$ на основе противника «без предупреждения» $\mathcal{A}_{\mathrm{SESPAKE}}$, используя его в качестве чёрного ящика. Для осмысленного противника «с предупреждением» логичнее делать запрос к оракулу $\mathcal{O}_{\mathrm{auth}}$ для участника Send-сессии, так как преобладание противника $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{exec}}}$ сравнимо с вероятностью успеха решения задачи CDH. Поэтому вместо противника $\mathcal{A}_{\mathrm{SESPAKE}}$ можно рассматривать противника $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$. В процессе атаки на протокол противник «с предупреждением», работая как транслятор между оракулами и противником $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$, случайно и равновероятно выбирает одну из Send-сессий, инициированных противником $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$, и делает запрос к оракулу $\mathcal{O}_{\mathrm{auth}}$ для участника выбранной сессии.

Если противник $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$ сделал запрос к оракулу $\mathcal{O}_{\mathrm{check}}$ для участника выбранной противником $\mathcal{A}_{\mathrm{SESPAKE},w}$ сессии, $\mathcal{A}_{\mathrm{SESPAKE},w}$ возвращает противнику $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$ первую часть ответа на сделанный им запрос к оракулу $\mathcal{O}_{\mathrm{auth}}$. Если противник $\mathcal{A}_{\mathrm{SESPAKE}_{\mathrm{send}}}$ сделал запрос к оракулу $\mathcal{O}_{\mathrm{check}}$ для участника другой сессии, $\mathcal{A}_{\mathrm{SESPAKE},w}$ просто транслирует этот запрос.

Если противник $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ сделал запрос к оракулу $\mathcal{O}_{\text{auth}}$ для участника выбранной противником $\mathcal{A}_{\text{SESPAKE},w}$ сессии, $\mathcal{A}_{\text{SESPAKE},w}$ возвращает противнику $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ вторую часть ответа на сделанный запрос к оракулу $\mathcal{O}_{\text{auth}}$, а в качестве результата выдаёт тот же бит, что и противник $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$. Если $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ сделал запрос к оракулу $\mathcal{O}_{\text{auth}}$ для участника другой сессии, $\mathcal{A}_{\text{SESPAKE},w}$ случайно выбирает ключ, вычисляет на нём значение $M'$ и передаёт его противнику $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$, а в качестве результата выдаёт случайное значение бита.

Для противника $\mathcal{A}_{\text{SESPAKE},w}$ вероятность того, что выбранная им сессия совпадёт с сессией, для участника которой $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ сделает запрос к оракулу $\mathcal{O}_{\text{auth}}$, равна $1/q_{\text{send}}$. Следовательно, преобладание $\mathcal{A}_{\text{SESPAKE},w}$ меньше преобладания $\mathcal{A}_{\text{SESPAKE}_{\text{send}}}$ не более чем в $q_{\text{send}}$ раз:

$$\mathbf{Insec}_{\text{SESPAKE},w}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}) \geqslant \frac{1}{q_{\text{send}}} \mathbf{Insec}_{\text{SESPAKE}_{\text{send}}}(t, q_{\text{exec}}, q_{\text{send}}, q_H, q_{\text{HMAC}}).$$

Отсюда получаем искомую оценку. Теорема 9 доказана. ■

## 7. Комментарии и пояснения

Обсудим некоторые конструктивные особенности протокола SESPAKE и их влияние на его свойства.

### 7.1. Ограничения количества сеансов для одного пароля

Говоря неформально, протокол SESPAKE считается стойким, если лучшим способом определения сессионного ключа (или какой-либо информации о нём) является опробование пароля путём взаимодействия с участником сети (будем называть такой способ online-угадыванием пароля). Недопустимой для стойкого протокола альтернативой online-угадыванию является практически осуществимый перебор, который не требует такого взаимодействия (offline-перебор).

При online-угадывании пароля противником каждая неудачная попытка приводит к разрыву соединения, вызванному тем, что проверки, предусмотренные протоколом $\text{SESPAKE}_{KC}$, не проходят. Отсутствие ограничений на количество возможных неудачных попыток соединения приводит к существенному увеличению вероятности online-угадывания. В связи с этим в протоколе SESPAKE должно быть предусмотрено ограничение как количества неудачных попыток соединения подряд, так и количества неудачных соединений для данного установленного пароля. Необходимо также вводить ограничение на общее количество соединений — как удачных, так и неудачных. Эти ограничения являются важной частью протокола.

Отметим влияние указанных ограничений на оценку теоремы 4. Так, например, в слагаемом $(2q_{\text{exec}} + q_{\text{send}})^2/q$ величина $q_{\text{exec}}$ органичена количеством допустимых сеансов работы протокола, а $q_{\text{send}}$ — ещё и количеством допустимых неудачных соединений для данного пароля. Обычно допустимое количество неудачных соединений полагается равным нескольким десяткам, а допустимое количество сеансов ограничивается величиной порядка $10^7$–$10^9$. В таком случае, если $q \approx 2^{256}$, то

$$\frac{(2q_{\text{exec}} + q_{\text{send}})^2}{q} \leqslant (3 \cdot 10^9)^2/2^{256} \leqslant 2^{64}/2^{256} = 2^{-192}.$$

### 7.2. Обработка случая точки малого порядка

Рассмотрим причины, по которым при выработке общего ключа случай, когда $\frac{m}{q} Q_B = 0_E$ $\left(\text{или } \frac{m}{q} Q_A = 0_E\right)$, обрабатывается специальным образом. Под «специ-

альной обработкой» понимается использование флага $z_b$ и точки $P$ в качестве $Q_B$, если $\frac{m}{q} Q_B = 0_E$.

Протокол, в котором указанный случай не обрабатывается специальным образом, может быть уязвим к online-перебору паролей при нестрогом использовании счётчиков количества неудачных попыток аутентификации. Поясним этот тезис.

Пусть счётчик неудачных попыток аутентификации уменьшается в начале протокола SESPAKE$_{KC}$, а случай $\frac{m}{q} Q_B = 0_E$ не обрабатывается специальным образом. Тогда противник, имитируя сторону $A$, посылает стороне $B$ точки $u_1 = X - Q_{PW'}$, где $\frac{m}{q} X = 0_E$, для разных $PW'$ и измеряет время, через которое $B$ отвечает на запрос; после ответа стороны $B$ противник разрывает соединение. Таким образом, сторона $B$ не уменьшает счетчик неудачных попыток соединения, а измеренное время даёт противнику критерий для определения правильного пароля — для него время будет существенно меньше, так как операция вычисления кратной точки (для получения $src$) окажется вырожденной.

Разница во времени проверки условия $\frac{m}{q} Q_B = 0_E$ для случаев $Q_B = 0_E$ и $Q_B \neq 0_E$ несущественна, так как кофактор на практике достаточно мал. Это объясняется тем, что характеристика $p$ базового поля в целях оптимизации выбирается минимально возможной, чтобы достичь установленных в [28] значений параметра $q$ ($2^{254} < q < 2^{256}$ или $2^{508} < q < 2^{512}$). Поэтому значения $q$ и $p$ оказываются близкими, а значит, малым оказывается и значение кофактора $m/q$.

Описанный метод анализа становится неприменим, если стороны уменьшают счётчик неправильных попыток аутентификации в самом начале работы.

## Заключение

Протоколы типа PAKE являются одним из наиболее мощных механизмов для достижения стойкости аутентификации с использованием параметров, которые пользователь способен запомнить. Разработанный и стандартизированный в России протокол SESPAKE является самодостаточным протоколом такого типа, который может применяться в системах, в которых после аутентификации и выработки сеансовых ключей работает протокол защиты данных наподобие подпротокола Record протокола TLS. В настоящей работе приведены оценки стойкости данного протокола, которые позволяют выбирать значения его параметров для достижения требуемых порогов безопасности конечными реализациями.

## ЛИТЕРАТУРА

1. *Bellovin S. and Merritt M.* Encrypted key exchange: password-based protocols secure against dictionary attacks // Proc. IEEE Symp. Research in Security and Privacy, Oakland, CA, USA, May 1992. P. 72–84.
2. *Bellare M. and Rogaway P.* The AuthA protocol for password-based authenticated key exchange // Contributions to IEEE P1363. March 2000. `https://web.cs.ucdavis.edu/~rogaway/papers/autha.pdf`
3. *Bresson E., Chevassut O., and Pointcheval D.* Security proofs for an efficient password-based key exchange // Proc. 10th ACM Conf. CCS'03. 2003. P. 241–250.
4. *Boyko V., MacKenzie P., and Patel S.* Provably secure password authenticated and key exchange using Diffie — Hellman // LNCS. 2000. V. 1807. P. 156–171.

5. *Canetti R., Halevi S., Katz J., et al.* Universally composable password-based key exchange // LNCS. 2005. V. 3494. P. 404–421.
6. *Bellare M., Canetti R., and Krawczyk H.* A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract) // Proc. 30th Ann. ACM Symp. Theory Comput. ACM Press, 1998. P. 419–428.
7. *Bellare M. and Rogaway P.* Entity authentication and key distribution // LNCS. 1994. V. 773. P. 232–249.
8. *Bellare M., Pointcheval D., and Rogaway P.* Authenticated key exchange secure against dictionary attacks // LNCS. 2000. V. 1807. P. 139–155.
9. *Abdalla M. and Pointcheval D.* Simple password-based encrypted key exchange protocols // LNCS. 2005. V. 3376. P. 191–208.
10. *Bender J., Fischlin M., and Kugler D.* Security analysis of the PACE key-agreement protocol // LNCS. 2009. V. 5735. P. 33–48.
11. *Jager T., Kohlar F., Schage S., and Schwenk J.* On the Security of TLS-DHE in the Standard Model. Cryptology ePrint Archive: Report 2011/219.
12. Рекомендации по стандартизации Р 50.1.115–2016 «Информационная технология. Криптографическая защита информации. Протокол выработки общего ключа с аутентификацией на основе пароля». М.: Стандартинформ, 2016.
13. *Алексеев Е. К., Ахметзянова Л. Р., Ошкин И. Б., Смышляев С. В.* Обзор уязвимостей некоторых протоколов выработки общего ключа с аутентификацией на основе пароля и принципы построения протокола SESPAKE // Математические вопросы криптографии. 2016. Т. 7. № 4. С. 7–28.
14. *Abdalla M.* Reducing The Need For Trusted Parties In Cryptography. `https://tel.archives-ouvertes.fr/tel-00917187`. 2011.
15. *Bellare M.* Practice-oriented provable-security // LNCS. 1998. V. 1396. P. 221–231.
16. *Ahmetzyanova L. R., Alekseev E. K., Karpunin G. A., and Smyshlyaev S. V.* On cryptographic properties of the CVV and PVV parameters generation procedures in payment systems // Математические вопросы криптографии. 2018. Т. 9. № 2. С. 23–46.
17. *Goldreich O.* Foundations of Cryptography. V. 1. N.Y.: Cambridge University Press, 2006.
18. *Алексеев Е. К., Ахметзянова Л. Р., Зубков А. М. и др.* Об одном подходе к формализации задач криптографического анализа // Математические вопросы криптографии. 2020 (в печати).
19. *Halderman J. A., Schoen S. D., Heninger N., et al.* Lest we remember: Cold-boot attacks on encryption keys // Commun. ACM. 2009. V. 52. No. 5. `http://www.cs.umd.edu/class/spring2016/cmsc414/papers/coldboot.pdf`
20. *Bellare M., Desai A., Jokipii E., and Rogaway P.* A concrete security treatment of symmetric encryption // Proc. 38th Symp. Foundations Comput. Sci. IEEE, 2000. P. 394–403.
21. *Abdalla M., Fouque P.A., and Pointcheval D.* Password-based authenticated key exchange in the three-party setting // LNCS. 2005. V. 3386. P. 65–84.
22. *Bellare M. and Rogaway P.* Random oracles are practical: a paradigm for designing efficient protocols // Proc. 1st ACM Conf. CCS'93. 1993. P. 62–73. `https://doi.org/10.1145/168588.168596`
23. *Alekseev E., Nikolaev V., and Smyshlyaev S.* On the security properties of Russian standardized elliptic curves // Математические вопросы криптографии. 2018. Т. 9. № 3. С. 5–32.
24. Рекомендации по стандартизации Р 50.1.111–2016 «Информационная технология. Криптографическая защита информации. Парольная защита ключевой информации». М.: Стандартинформ, 2016.

25. ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция хэширования». М.: Стандартинформ, 2012.
26. Рекомендации по стандартизации Р 50.1.113–2016 «Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов электронной цифровой подписи и функции хэширования». М.: Стандартинформ, 2016.
27. *Vercauteren F.* Final Report on Main Computational Assumptions in Cryptography. ICT-2007-216676. ECRYPT II, European Network of Excellence in Cryptology II. D.MAYA.6. 2013.
28. ГОСТ Р 34.10-2012 «Информационная технология. Процессы формирования и проверки электронной цифровой подписи». М.: Стандартинформ, 2012.

REFERENCES

1. *Bellovin S. and Merritt M.* Encrypted key exchange: password-based protocols secure against dictionary attacks. Proc. IEEE Symp. Research in Security and Privacy, Oakland, CA, USA, May 1992, pp. 72–84.
2. *Bellare M. and Rogaway P.* The AuthA protocol for password-based authenticated key exchange. Contributions to IEEE P1363, March 2000. `https://web.cs.ucdavis.edu/~rogaway/papers/autha.pdf`
3. *Bresson E., Chevassut O., and Pointcheval D.* Security proofs for an efficient password-based key exchange. Proc. 10th ACM Conf. CCS'03, 2003, pp. 241–250.
4. *Boyko V., MacKenzie P., and Patel S.* Provably secure password authenticated and key exchange using Diffie — Hellman. LNCS, 2000, vol. 1807, pp. 156–171.
5. *Canetti R., Halevi S., Katz J., et al.* Universally composable password-based key exchange. LNCS, 2005, vol. 3494, pp. 404–421.
6. *Bellare M., Canetti R., and Krawczyk H.* A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). Proc. 30th Ann. ACM Symp. Theory Comput., ACM Press, 1998, pp. 419–428.
7. *Bellare M. and Rogaway P.* Entity authentication and key distribution. LNCS, 1994, vol. 773, pp. 232–249.
8. *Bellare M., Pointcheval D., and Rogaway P.* Authenticated key exchange secure against dictionary attacks. LNCS, 2000, vol. 1807, pp. 139–155.
9. *Abdalla M. and Pointcheval D.* Simple password-based encrypted key exchange protocols. LNCS, 2005, vol. 3376, pp. 191–208.
10. *Bender J., Fischlin M., and Kugler D.* Security analysis of the PACE key-agreement protocol. LNCS, 2009, vol. 5735, pp. 33–48.
11. *Jager T., Kohlar F., Schage S., and Schwenk J.* On the Security of TLS-DHE in the Standard Model. Cryptology ePrint Archive: Report 2011/219.
12. Rekomendatsii po standartizatsii R 50.1.115–2016 «Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Protokol vyrabotki obshchego klyucha s autentifikatsiey na osnove parolya» ["Information Technology. Cryptographic Data Security. Password Authenticated Key Establishment Protocol". Recommendations for Standardization R 50.1.115–2016]. Moscow, Standartinform Publ., 2016. (in Russian)
13. *Alekseev E. K., Akhmetzyanova L. R., Oshkin I. B., and Smyshlyaev S. V.* Obzor uyazvimostey nekotorykh protokolov vyrabotki obshchego klyucha s autentifikatsiey na osnove parolya i printsipy postroeniya protokola SESPAKE [A review of the password authenticated key exchange protocols vulnerabilities and principles of the SESPAKE protocol construction]. Matem. Vopr. Kriptogr., 2016, vol. 7, no. 4, pp. 7–28. (in Russian)

14. *Abdalla M.* Reducing The Need For Trusted Parties In Cryptography. https://tel.archives-ouvertes.fr/tel-00917187. 2011.
15. *Bellare M.* Practice-oriented provable-security. LNCS, 1998, vol. 1396, pp. 221–231.
16. *Ahmetzyanova L. R., Alekseev E. K., Karpunin G. A., and Smyshlyaev S. V.* On cryptographic properties of the CVV and PVV parameters generation procedures in payment systems. Matem. Vopr. Kriptogr., 2018, vol. 9, no. 2, pp. 23–46.
17. *Goldreich O.* Foundations of Cryptography. Vol. 1. New York, Cambridge University Press, 2006.
18. *Alekseev E. K., Axmetzyanova L. R., Zubkov A. M., et al.* Ob odnom podkhode k formalizatsii zadach kriptograficheskogo analiza [On one approach to formalization of cryptographic analysis tasks]. Matem. Vopr. Kriptogr., 2020, to be published. (in Russian)
19. *Halderman J. A., Schoen S. D., Heninger N., et al.* Lest we remember: Cold-boot attacks on encryption keys. Commun. ACM, 2009, vol. 52, no. 5. http://www.cs.umd.edu/class/spring2016/cmsc414/papers/coldboot.pdf
20. *Bellare M., Desai A., Jokipii E., and Rogaway P.* A concrete security treatment of symmetric encryption. Proc. 38th Symp. Foundations Comput. Sci., IEEE, 2000, pp. 394–403.
21. *Abdalla M., Fouque P.A., and Pointcheval D.* Password-based authenticated key exchange in the three-party setting. LNCS, 2005, vol. 3386, pp. 65–84.
22. *Bellare M. and Rogaway P.* Random oracles are practical: a paradigm for designing efficient protocols. Proc. 1st ACM Conf. CCS'93, 1993, pp. 62–73. https://doi.org/10.1145/168588.168596
23. *Alekseev E., Nikolaev V., and Smyshlyaev S.* On the security properties of Russian standardized elliptic curves. Matem. Vopr. Kriptogr., 2018, vol. 9, no. 3, pp. 5–32.
24. Rekomendatsii po standartizatsii R 50.1.111–2016 «Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Parol'naya zashchita klyuchevoy informatsii» ["Information Technology. Cryptographic Data Security. Password-Based Key Protection". Recommendations for Standardization R 50.1.111–2016]. Moscow, Standartinform Publ., 2016. (in Russian)
25. GOST R 34.11-2012 «Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Funktsiya kheshirovaniya» [GOST R 34.11-2012 "Information Technology. Cryptographic Data Security. Hash Function"]. Moscow, Standartinform Publ., 2012. (in Russian)
26. Rekomendatsii po standartizatsii R 50.1.113–2016 «Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Kriptograficheskie algoritmy, soputstvuyushchie primeneniyu algoritmov elektronnoy tsifrovoy podpisi i funktsii kheshirovaniya» ["Information Technology. Cryptographic Data Security. Additional Cryptographic Algorithms for Digital Signature Algorithms and Hash Function". Recommendations for Standardization R 50.1.113–2016]. Moscow, Standartinform Publ., 2016. (in Russian)
27. *Vercauteren F.* Final Report on Main Computational Assumptions in Cryptography. ICT-2007-216676, ECRYPT II, European Network of Excellence in Cryptology II. D.MAYA.6. 2013.
28. GOST R 34.10-2012 «Informatsionnaya tekhnologiya. Protsessy formirovaniya i proverki elektronnoy tsifrovoy podpisi» [GOST R 34.10-2012 "Information Technology. Cryptographic Data Security. Processes of Digital Signature Creation and Verification"]. Moscow, Standartinform Publ., 2012. (in Russian)

УДК 519.7

# КРИПТОАНАЛИЗ АСИММЕТРИЧНОГО ШИФРА НА БУЛЕВЫХ ФУНКЦИЯХ

И. В. Боровкова, В. А. Кондратьев, И. А. Панкратова

*Национальный исследовательский Томский государственный университет, г. Томск, Россия*

Рассматривается асимметричная шифрсистема ACBF, ключом в которой служит обратимая векторная булева функция. Ключевая функция строится из порождающей (которая считается известной) с помощью операций инверсии и перестановки переменных и координат. Из этих четырёх операций некоторые являются тождественными (о чём заранее известно криптоаналитику); остальные образуют множество ключевых параметров; нахождение их значений является целью атаки. Для семи из 15 возможных наборов ключевых параметров описаны атаки с известным (для некоторых — и с выбираемым) открытым текстом, приведены оценки их сложности.

**Ключевые слова:** *криптосистема ACBF, векторные булевы функции, криптоанализ.*

DOI 10.17223/20710410/50/2

# CRYPTANALYSIS OF AN ASYMMETRIC CIPHER ON BOOLEAN FUNCTIONS

I. V. Borovkova, V. A. Kondrat'ev, I. A. Pankratova

*National Research Tomsk State University, Tomsk, Russia*

**E-mail:** iborovkova95@gmail.com, wadim.condratjev@yandex.ru, pank@mail.tsu.ru

The asymmetric encryption system ACBF is considered. Its key is an invertible vectorial Boolean function constructing from a generating function (which is considered known) using the negation and permutation operations of variables and coordinates. Of these four operations, some are identical, the rest form a set of key parameters; finding them is the goal of the attack. For seven of 15 possible sets of key parameters, attacks with known plaintext are described, their complexity is given. For five sets of key parameters, attacks with choosen plaintext are presented too. The main stage of the attacks is the solution of the auxiliary problem of finding a columns permutation, with the means of which one Boolean matrix is obtained from another. It has been proved that, for uniquely determing the key, it is necessary to have $2\log n$ plaintexts (in average) in the attack with a known plaintext, and it is enough to choose $\log n$ plaintexts in the attack with a choosen plaintext, where $n$ is the length of text.

**Keywords:** *ACBF cryptosystem, vectorial Boolean functions, cryptanalysis.*

## Введение

Рассматривается шифрсистема ACBF (Asymmetric Cryptosystem on Boolean Functions) [1, 2]. Открытые тексты и шифртексты в ней — это булевы векторы длины $n$;

открытый ключ — функция $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$; закрытый ключ — функция $f^{-1}$; шифрование и расшифрование выполняются по правилам $y = f(x)$ и $x = f^{-1}(y)$ соответственно.

Функция $f$ строится так. Выбирается обратимая функция $g : \mathbb{F}_2^n \to \mathbb{F}_2^n$ — *порождающая функция криптосистемы*; функция $f$ получается из неё с помощью перестановки и инверсии переменных $x_i$ и координат $g_i$: $f(x) = \pi_2(g^{\sigma_2}(\pi_1(x^{\sigma_1})))$, где $\sigma_1, \sigma_2 \in \mathbb{F}_2^n$; $\pi_1, \pi_2 \in \mathbb{S}_n$; $x^{\sigma_1} = x \oplus \overline{\sigma}_1$; $g^{\sigma_2}(x) = g(x) \oplus \overline{\sigma}_2$; $\pi_1(x) = x_{\pi_1(1)} \dots x_{\pi_1(n)}$; $\pi_2(g(x)) = g_{\pi_2(1)}(x) \dots g_{\pi_2(n)}(x)$.

В качестве ключевых параметров шифрсистемы ACBF выступают элементы любого непустого подмножества $J \subseteq \{\pi_1, \pi_2, \sigma_1, \sigma_2\}$ (всего 15 вариантов); операции из множества $\{\pi_1, \pi_2, \sigma_1, \sigma_2\} \backslash J$ считаются тождественными.

Рассмотрим атаки с известным и с выбираемым открытым текстом в следующих предположениях:

1) для любого $x \in \mathbb{F}_2^n$ криптоаналитик может вычислить $g(x)$ и $g^{-1}(x)$;
2) криптоаналитик знает, какие именно из операций $\pi_1, \pi_2, \sigma_1, \sigma_2$ входят в множество $J$, но их конкретные значения ему не известны; цель — найти эти значения;
3) при атаке с выбираемым открытым текстом криптоаналитик может вычислить $f(x)$ для любого $x \in \mathbb{F}_2^n$.

В работе [1] в тех же предположениях описаны атаки с известным открытым текстом для всех 15 вариантов подмножеств ключевых параметров, приведены оценки их сложности. В данной работе более подробно представлены алгоритмы и уточнены оценки, а также рассмотрены атаки с выбираемым открытым текстом для некоторых вариантов.

Заметим, что случаи $J = \{\sigma_1\}$ и $\{\sigma_2\}$ не представляют интереса, так как для них ключевой параметр находится тривиально по одной паре «открытый текст — шифртекст»:

$$J = \{\sigma_1\}: \quad y = f(x) = g(x \oplus \overline{\sigma}_1) \;\Rightarrow\; \overline{\sigma}_1 = x \oplus g^{-1}(y),$$
$$J = \{\sigma_2\}: \quad y = f(x) = g(x) \oplus \overline{\sigma}_2 \;\Rightarrow\; \overline{\sigma}_2 = g(x) \oplus y.$$

## 1. Поиск перестановки столбцов

Рассмотрим вспомогательную задачу. Пусть даны две булевых матрицы, одна из которых получена перестановкой столбцов второй матрицы. Требуется найти эту перестановку.

Пусть $\pi \in \mathbb{S}_n$ — подстановка степени $n$; $A = ||a_{ij}||$, $B = ||b_{ij}||$ — булевы матрицы размера $m \times n$; $A_i, B_i$, $i = 1, \dots, m$, — их строки; $A^{(j)}, B^{(j)}$, $j = 1, \dots, n$, — вектор-столбцы, причём $B^{(j)} = A^{(\pi(j))}$ для $j = 1, \dots, n$ (будем обозначать $B = \pi(A)$); для $x, \sigma \in \{0, 1\}$ положим $x^\sigma = \begin{cases} x, & \sigma = 1, \\ \overline{x}, & \sigma = 0. \end{cases}$ Построим матрицу $D = ||d_{jk}||$ размера $n \times n$ так:

$$d_{jk} = \bigwedge_{i=1}^{m} a_{ik}^{b_{ij}}, \quad j, k = 1, \dots, n,$$

то же самое можно записать через покомпонентную конъюнкцию строк:

$$D_j = \bigwedge_{i=1}^{m} A_i^{b_{ij}}, \quad j = 1, \dots, n. \tag{1}$$

Будем обозначать $D = T(A, B)$.

**Утверждение 1.** Пусть $A = ||a_{ij}||$, $B = ||b_{ij}||$ — булевы матрицы размера $m \times n$, $B = \pi_1(A)$ для некоторой $\pi_1 \in \mathbb{S}_n$. Тогда в матрице $D = T(A, B)$ элемент $d_{jk} = 1$, если и только если существует подстановка $\pi \in \mathbb{S}_n$, такая, что $B = \pi(A)$ и $\pi(j) = k$.

***Доказательство.*** Заметим, что $d_{jk} = 1 \Leftrightarrow a_{ik} = b_{ij}$ для всех $i = 1, \ldots, m \Leftrightarrow$ $\Leftrightarrow B^{(j)} = A^{(k)}$.

Необходимость. Пусть $d_{jk} = 1$. Если $\pi_1(j) = k$, то $\pi = \pi_1$ — искомая подстановка. В противном случае пусть $\pi_1(j) = s \neq k$ и $\pi_1(i) = k$ для некоторого $i \in \{1, \ldots, n\}$. Тогда $B^{(j)} = A^{(s)}$, $B^{(i)} = A^{(k)}$, откуда ввиду $B^{(j)} = A^{(k)}$ получаем $A^{(k)} = A^{(s)}$. Умножим $\pi_1$ на транспозицию $(s, k)$, получим подстановку $\pi$, для которой $\pi(j) = k$, $\pi(i) = s$, $\pi(A) = B$.

Достаточность. Из условий $B = \pi(A)$ и $\pi(j) = k$ получаем $B^{(j)} = A^{(k)}$, откуда $d_{jk} = 1$. ■

**Замечание 1.** Если в матрице $A$ все столбцы различны, то существует единственная подстановка $\pi$ со свойством $B = \pi(A)$; в этом случае $D = T(A, B)$ — матрица этой подстановки (т. е. $B = AD^{\mathrm{T}}$).

**Замечание 2.** Пусть множество столбцов матрицы $A$ можно разбить на классы $Q_1, \ldots, Q_k$ одинаковых столбцов, $1 \leqslant k \leqslant n$, так, что $|Q_j| = r_j$, $j = 1, \ldots, k$, $r_1 + \ldots + r_k = n$. Тогда множество строк матрицы $D = T(A, B)$ тоже разбивается на $k$ классов одинаковых строк мощностей $r_1, \ldots, r_k$; вес строк в каждом классе равен мощности класса; все возможные подстановки $\pi$, для которых верно $\pi(A) = B$, удовлетворяют условию $\pi(i) = j \Leftrightarrow d_{ij} = 1$; количество таких подстановок равно $\prod\limits_{i=1}^{k} r_i!$.

Будем говорить, что *матрица $D = T(A, B)$ содержит все подстановки $\pi$, удовлетворяющие уравнению $\pi(A) = B$.*

**Пример 1.** Пусть

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

Тогда

$$D = T(A, B) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}; \quad \pi(1) = 2; \quad \pi(2), \pi(4) \in \{1, 3\}; \quad \pi(3) = 4,$$

т. е. все подстановки $\pi$, такие, что $\pi(A) = B$, — это $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$ и $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$.

**Замечание 3.** Рассмотрим случай, когда семейства столбцов в матрицах $A$ и $B$ различны, т. е. матрицу $B$ невозможно получить из матрицы $A$ никакой перестановкой столбцов. Пусть, например, $B^{(j_1)} = \ldots = B^{(j_r)} = A^{(k_1)} = \ldots = A^{(k_s)}$ и $r \neq s$ (в частности, может быть $s = 0$). Тогда в матрице $D = T(A, B)$ строки $D_{j_1}, \ldots, D_{j_r}$ одинаковы и имеют по $s$ единиц — мощность класса не равна весу строк в нём. В таком случае будем говорить, что $D$ *не является матрицей подстановок.*

**Пример 2.** Пусть

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Тогда $D = T(A,B) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ и $D$ не является матрицей подстановок, так как вес строк в классе $\{D_1, D_2\}$ равен 1, а в классе $\{D_3\}$ — 2, что не совпадает с мощностью классов.

## 2. Криптоанализ вариантов шифрсистемы ACBF

### 2.1. Случай $J = \{\pi_1\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = g(\pi_1(x)); \quad \pi_1(x) = g^{-1}(y).$$

Тогда если $P$ — матрица размера $m \times n$ со строками открытых текстов $P_1, \ldots, P_m$,

$$C_i = g(\pi_1(P_i)),\ i = 1, \ldots, m, \tag{2}$$

$P'$ — матрица размера $m \times n$ со строками $g^{-1}(C_i)$, $i = 1, \ldots, m$, то матрица $D = T(P, P')$ содержит все возможные подстановки $\pi_1$, удовлетворяющие системе уравнений (2).

Таким образом, алгоритм криптоанализа состоит в применении формулы (1) к матрицам $P$ и $P'$ в качестве $A$ и $B$ соответственно; его сложность равна $O(nm)$. В [3, алгоритм 1] приведён другой способ нахождения подстановки $\pi_1$; главный его недостаток — необходимость сравнивать *столбцы* булевых матриц, в то время как формула (1) представляет собой покомпонентную конъюнкцию *строк* или их инверсий и может быть выполнена для всех (или, по крайней мере, группы столбцов — в зависимости от длины строки и разрядности используемого типа данных) одновременно.

Кроме того, при вычислении по формуле (1) можно выполнять конъюнкцию не всех $m$ строк (или их инверсий) матрицы $P$, а остановиться тогда, когда вес строки $D_j$ станет равным 1. Так, в примере 1 для вычисления строки $D_1$ достаточно двух шагов ($\neg(1010) \wedge (1110) = (0100)$), как и для вычисления $D_3$ ($\neg(1010) \wedge \neg(1110) = (0001)$).

Компьютерные эксперименты с программами на языке ЛЯПАС [4] (в котором, в частности, есть операция взвешивания булева вектора) показывают преимущество вычислений по формуле (1) перед алгоритмом 1 из [3]; так, при $n = m = 31$ первый способ быстрее второго примерно в 10 раз, если учитывать время транспонирования матрицы в алгоритме 1, и в 1,3 раза без учёта этого времени.

Согласно замечаниям 1 и 2, ключ (подстановка $\pi_1$) определяется однозначно, если и только если все столбцы в матрице $P$ различны.

**Утверждение 2.** Если открытые тексты распределены случайно равновероятно, то при атаке с известным открытым текстом для однозначного определения ключа в среднем понадобится $2 \log n$ открытых текстов.

***Доказательство.*** В булевой матрице с $m$ строками столбцы могут принимать $2^m$ различных значений; при случайных равновероятных строках они также случайны и равновероятны. Согласно парадоксу дней рождения [5, Ch. 2.1.5], в среднем число попарно различных столбцов равно $n = \sqrt{\pi 2^m / 2}$. Отсюда получаем $m \approx 2 \log n$. ■

Справедливость утверждения 2 подтверждена в компьютерном эксперименте.

Пусть $x$ — булев вектор длины $k$; через $(x)^t$ обозначим булев вектор длины $kt$ — конкатенацию $t$ одинаковых векторов $x$.

**Утверждение 3.** Если производится атака с выбираемым открытым текстом, то для нахождения ключа достаточно рассмотреть $m = \lceil \log n \rceil$ пар $(P_i, C_i)$.

***Доказательство.*** Согласно замечанию 1, достаточно выбрать открытые тексты $P_i$, $i = 1, \ldots, m$, так, чтобы в матрице $P$ со строками $P_i$ все столбцы были различны.

Пусть $m = \lceil \log n \rceil$, $t = 2^m \geqslant n$. Построим матрицу $P''$ размера $m \times t$ со строками

$$\begin{aligned} P''_1 &= (0)^{t/2}(1)^{t/2}, \\ P''_2 &= (0)^{t/4}(1)^{t/4}(0)^{t/4}(1)^{t/4}, \\ &\ldots \\ P''_m &= (01)^{t/2} \end{aligned} \tag{3}$$

(очевидно, все её столбцы различны); удалим из неё любые $(t - n)$ столбцов; строки получившейся матрицы примем за открытые тексты $P_i$, $i = 1, \ldots, m$. ■

### 2.2. Случай $J = \{\pi_2\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = \pi_2(g(x)).$$

Тогда если $P_1, \ldots, P_m$ — открытые тексты, $C$ — матрица размера $m \times n$ со строками шифртекстов

$$C_i = \pi_2(g(P_i)),\ i = 1, \ldots, m, \tag{4}$$

$C'$ — матрица размера $m \times n$ со строками $g(P_i)$, $i = 1, \ldots, m$, то матрица $D = T(C', C)$ содержит все возможные подстановки $\pi_2$, удовлетворяющие системе уравнений (4).

Если открытые тексты $P_i$, $i = 1, \ldots, m$, выбираются в $\mathbb{F}_2^n$ случайно равновероятно, то, в силу обратимости (взаимной однозначности) функции $g$, значения $g(P_i)$ также имеют равномерное распределение. Поэтому по аналогии с утверждением 2 получаем, что при атаке с известным открытым текстом для однозначного определения ключа в среднем понадобится $2 \log n$ открытых текстов.

При атаке с выбираемым открытым текстом надо выбрать тексты $P_i$ так, чтобы в матрице со строками $g(P_i)$, $i = 1, \ldots, m$, все столбцы были различны. Это можно сделать, например, так: построить матрицу $P''$, как в (3), удалить из неё любые $t - n$ столбцов (обозначим строки получившейся матрицы $B_i$) и положить $P_i = g^{-1}(B_i)$, $i = 1, \ldots, m$.

### 2.3. Случай $J = \{\pi_1, \sigma_1\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = g(\pi_1(x^{\sigma_1})) = g(\pi_1(x \oplus b)),$$

где $b = \overline{\sigma}_1$. Поскольку $\pi_1(x \oplus b) = \pi_1(x) \oplus \pi_1(b)$, имеем

$$g^{-1}(y) = \pi_1(x) \oplus \pi_1(b).$$

Составим систему

$$\begin{cases} g^{-1}(C_1) = \pi_1(P_1) \oplus \pi_1(b), \\ \ldots \\ g^{-1}(C_m) = \pi_1(P_m) \oplus \pi_1(b), \\ g^{-1}(C_{m+1}) = \pi_1(P_{m+1}) \oplus \pi_1(b), \end{cases}$$

прибавим по модулю 2 последнее уравнение к каждому из предыдущих и снова применим свойство линейности подстановки $\pi_1$; в результате получим систему

$$\begin{cases} g^{-1}(C_1) \oplus g^{-1}(C_{m+1}) = \pi_1(P_1 \oplus P_{m+1}), \\ \dots \\ g^{-1}(C_m) \oplus g^{-1}(C_{m+1}) = \pi_1(P_m \oplus P_{m+1}), \\ g^{-1}(C_{m+1}) = \pi_1(P_{m+1}) \oplus \pi_1(b). \end{cases} \tag{5}$$

Построим матрицы $P_\oplus$ со строками $P_i \oplus P_{m+1}$ и $P'_\oplus$ со строками $g^{-1}(C_i) \oplus g^{-1}(C_{m+1})$, $i = 1, \dots, m$. Тогда матрица $D = T(P_\oplus, P'_\oplus)$ содержит все возможные подстановки $\pi_1$, удовлетворяющие первым $m$ уравнениям системы (5).

Зная $\pi_1$, значение $\sigma_1$ найдём из последнего уравнения: $\sigma_1 = \overline{b}$, $b = \pi_1^{-1}(g^{-1}(C_{m+1})) \oplus$ $\oplus P_{m+1}$. Сложность атаки та же, что в предыдущих случаях: $O(nm)$; для однозначного определения ключа в атаке с известным открытым текстом в среднем понадобится $2 \log n$ открытых текстов; при атаке с выбором открытого текста ключ определится однозначно, если открытые тексты $P_i$, $i = 1, \dots, m+1$, $m = \lceil \log n \rceil$, выбрать следующим образом: построить матрицу $P''$, как в (3); удалить из неё любые $t - n$ столбцов; $P_{m+1}$ выбрать случайно в $\mathbb{F}_2^n$; открытый текст $P_i$ для $i = 1, \dots, m$ получить как сумму $i$-й строки матрицы и $P_{m+1}$.

### 2.4. Случай $J = \{\pi_2, \sigma_2\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = \pi_2(g^{\sigma_2}(x)) = \pi_2(g(x) \oplus d),$$

где $d = \overline{\sigma}_2$. Пусть $P_1, \dots, P_m, P_{m+1}$ — открытые тексты; $C_1, \dots, C_m, C_{m+1}$ — соответствующие шифртексты. Составим систему

$$\begin{cases} \pi_2^{-1}(C_1) = g(P_1) \oplus d, \\ \dots \\ \pi_2^{-1}(C_m) = g(P_m) \oplus d, \\ \pi_2^{-1}(C_{m+1}) = g(P_{m+1}) \oplus d, \end{cases}$$

прибавим по модулю 2 последнее уравнение к каждому из предыдущих, применим к обеим частям уравнений подстановку $\pi_2$ и учтём её линейность; в результате получим систему

$$\begin{cases} C_i \oplus C_{m+1} = \pi_2(g(P_i) \oplus g(P_{m+1})), \quad i = 1, \dots, m, \\ \pi_2^{-1}(C_{m+1}) = g(P_{m+1}) \oplus d. \end{cases} \tag{6}$$

Построим матрицы $C_\oplus$ со строками $C_i \oplus C_{m+1}$ и $C'_\oplus$ со строками $g(P_i) \oplus g(P_{m+1})$, $i =$ $= 1, \dots, m$. Тогда матрица $D = T(C'_\oplus, C_\oplus)$ содержит все возможные подстановки $\pi_2$, удовлетворяющие первым $m$ уравнениям системы (6).

Зная $\pi_2$, значение $\sigma_2$ найдём из условия $\sigma_2 = \overline{d}$, $d = \pi_2^{-1}(C_{m+1}) \oplus g(P_{m+1})$. Сложность атаки $O(nm)$; для однозначного определения ключа в атаке с известным открытым текстом в среднем понадобится $2 \log n$ открытых текстов; при атаке с выбором открытого текста ключ определится однозначно, если открытые тексты $P_i$, $i = 1, \dots, m+1$, $m = \lceil \log n \rceil$, выбрать следующим образом: построить матрицу $P''$, как в (3); удалить из неё любые $t - n$ столбцов (обозначим строки получившейся матрицы $B_i$); $P_{m+1}$ выбрать случайно в $\mathbb{F}_2^n$ и положить $P_i = g^{-1}(B_i \oplus g(P_{m+1}))$, $i = 1, \dots, m$.

### 2.5. Случай $J = \{\pi_1, \sigma_2\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = g^{\sigma_2}(\pi_1(x)) = g(\pi_1(x)) \oplus d, \quad \pi_1(x) = g^{-1}(y \oplus d), \quad \text{где } d = \overline{\sigma}_2.$$

**Утверждение 4.** Если производится атака с выбираемым открытым текстом, то для нахождения ключевых параметров достаточно $\lceil \log n \rceil + 1$ пар $(P_i, C_i)$.

***Доказательство.*** Положим $m = \lceil \log n \rceil$ и выберем открытые тексты $P_1, \ldots, P_m$, как в утверждении 3, а $P_{m+1} \in \{(0)^n, (1)^n\}$. Пусть

$$C_i = g(\pi_1(P_i)) \oplus d, \quad i = 1, \ldots, m+1. \tag{7}$$

Тогда $\pi_1(P_{m+1}) = P_{m+1}$, следовательно, $d$ находится однозначно: $d = g(P_{m+1}) \oplus C_{m+1}$. Построим матрицы $P$ и $P'$ размера $m \times n$ со строками $P_i$ и $g^{-1}(C_i \oplus d)$ соответственно. Матрица $D = T(P, P')$ содержит все подстановки $\pi_1$, удовлетворяющие системе (7). ■

Атака с известным открытым текстом описана в [1] и использует тот факт, что перестановка компонент вектора $x$ не меняет его вес $\mathrm{w}(x)$, поэтому

$$\mathrm{w}(P_i) = \mathrm{w}(g^{-1}(C_i \oplus d)), \quad i = 1, \ldots, m. \tag{8}$$

Систему (8) будем решать последовательно. Для нахождения всех $d$, удовлетворяющих первому уравнению, построим множества

$$X = \{x \in \mathbb{F}_2^n : \mathrm{w}(x) = \mathrm{w}(P_1)\}, \quad D_1 = \{g(x) \oplus C_1 : x \in X\}.$$

Тогда $\mathrm{w}(P_1) = \mathrm{w}(g^{-1}(C_1 \oplus d))$ для любого $d \in D_1$. Затем построим множества

$$D_i = \{d \in D_{i-1} : \mathrm{w}(P_i) = \mathrm{w}(g^{-1}(C_i \oplus d))\}, \quad i = 2, \ldots, m;$$

имеем: $D_m$ — множество значений $d$, удовлетворяющих системе (8).

Поскольку $|X| = |D_1| = \mathrm{C}_n^{\mathrm{w}(P_1)}$, для сокращения перебора имеет смысл в качестве $P_1$ выбирать из имеющегося множества открытых текстов такой, вес которого максимально далёк от $n/2$.

Для каждого $d \in D_m$ находим все возможные подстановки $\pi_1$, как в доказательстве утверждения 4. Если для некоторого $d$ получим, что матрица $T(P, P')$ не является матрицей подстановок (см. замечание 3), то такое $d$ не может быть частью ключа. Количество значений $d \in D_m$ в худшем случае равно $\mathrm{C}_n^{n/2}$.

### 2.6. Случай $J = \{\pi_2, \sigma_1\}$

Пусть $x, y \in \mathbb{F}_2^n$ — пара «открытый текст — соответствующий шифртекст». По определению криптосистемы ACBF получаем

$$y = f(x) = \pi_2(g(x^{\sigma_1})) = \pi_2(g(x \oplus b)), \quad \text{где } b = \overline{\sigma}_1.$$

Атака с известным открытым текстом аналогична предыдущему случаю: находим все векторы $b$, удовлетворяющие условию

$$\mathrm{w}(C_i) = \mathrm{w}(g(P_i \oplus b)), \quad i = 1, \ldots, m,$$

выбирая в качестве $C_1$ шифртекст с максимально далёким от $n/2$ весом; для каждого из них все возможные подстановки $\pi_2$ содержатся в матрице $T(C', C)$, где $C', C$ — матрицы со строками $g(P_i \oplus b)$ и $C_i$ соответственно.

### 2.7. Случай $J = \{\pi_1, \pi_2\}$

По определению получаем

$$y = f(x) = \pi_2(g(\pi_1(x))).$$

В [3] сформулировано (без доказательства) следующее утверждение.

**Утверждение 5.** Пусть имеется $m$ пар открытых текстов и шифртекстов вида $(P_i, C_i)$, $i = 1, \ldots, m$. Составим матрицы $P$ и $C$ размера $m \times n$ со строками $P_i$ и $C_i$ соответственно. Необходимым условием единственности решения системы уравнений

$$C_i = \pi_2(g(\pi_1(P_i))), \quad i = 1, \ldots, m, \tag{9}$$

относительно подстановок $\pi_1, \pi_2$ является отсутствие одинаковых столбцов в каждой из матриц $P$ и $C$.

***Доказательство.*** Предположим, что подстановки $\pi_1, \pi_2$ удовлетворяют системе (9) и в матрице $P$ есть одинаковые столбцы, например $P^{(j)} = P^{(k)}$, $j \neq k$. Построим подстановку $\pi_1'$, умножив $\pi_1$ на транспозицию $(\pi_1(k), \pi_1(j))$. Тогда $\pi_1'(P) = \pi_1(P)$, т. е. $\pi_1'(P_i) = \pi_1(P_i)$ для всех $i = 1, \ldots, m$, и пара $\pi_1', \pi_2$ также удовлетворяет системе (9).

Аналогично доказывается необходимость отсутствия одинаковых столбцов в матрице $C$. ■

Атака с известным открытым текстом может быть проведена следующим образом:

1) строим матрицу $C$ со строками $C_i$, $i = 1, \ldots, m$;
2) перебираем $n!$ подстановок $\pi_1'$ («кандидатов» в $\pi_1$);
3) для каждой $\pi_1'$ строим матрицу $C'$ со строками $g(\pi_1'(P_i))$, $i = 1, \ldots, m$;
4) находим $D = T(C', C)$;
5) если $D$ не является матрицей подстановок, то $\pi_1'$ не может быть частью ключа; иначе все пары $(\pi_1, \pi_2)$, где $\pi_1 = \pi_1'$ и $\pi_2$ содержится в $D$, удовлетворяют системе (9).

Сложность атаки та же, что и у предложенных в [1, 3], — $O(n!)$. Можно попытаться сократить перебор подстановок $\pi_1'$, воспользовавшись, как и в прошлых двух случаях, инвариантностью веса булева вектора относительно перестановки его координат, а именно: в качестве «кандидатов» в $\pi_1$ рассматривать только такие подстановки $\pi_1'$, которые удовлетворяют условиям

$$\mathrm{w}(g(\pi_1'(P_i))) = \mathrm{w}(C_i), \quad i = 1, \ldots, m. \tag{10}$$

Для этого на шаге 3 атаки надо дополнительно проверять условие $\mathrm{w}(g(\pi_1'(P_i))) = \mathrm{w}(C_i)$ и прекращать построение матрицы $C'$, переходя к следующей подстановке $\pi_1'$, сразу же, как только оно нарушится для некоторого $i \in \{1, \ldots, m\}$. Для сокращения перебора имеет смысл переупорядочить пары $(P_i, C_i)$ так, чтобы сначала шли шифртексты с максимально далёким от $n/2$ весом.

Полезность такой модификации на практике составляет предмет дальнейших исследований; в частности, предстоит выяснить:

1) что более трудозатратно — вычисление $m$ раз (в худшем случае) веса вектора $g(\pi_1'(P_i))$ или проверка того, является ли матрица $D$ матрицей подстановок;
2) сколько в среднем строк матрицы $C'$ приходится строить, прежде чем обнаруживается нарушение условия (10).

## Заключение

Для шифрсистемы ACBF рассмотрены следующие подмножества ключевых параметров: $J = \{\pi_1\}$, $\{\pi_2\}$, $\{\pi_1, \sigma_1\}$, $\{\pi_2, \sigma_2\}$, $\{\pi_1, \sigma_2\}$, $\{\pi_2, \sigma_1\}$, $\{\pi_1, \pi_2\}$. Описаны атаки с известным открытым текстом; для первых пяти вариантов — и с выбираемым открытым текстом.

В таблице приведён порядок $h(n)$ вычислительной сложности $O(h(n))$ для предложенных атак; для сравнения в третьей строке указан порядок сложности атаки, построенной по общей схеме [2], $p(n)$ — некоторый полином от $n$. Видно, что для $J = \{\pi_1, \sigma_1\}$ получена атака меньшей сложности; для остальных случаев уточнены оценки и/или алгоритмы атак.

| $J$ | $\{\pi_1\}$ | $\{\pi_2\}$ | $\{\pi_1, \sigma_1\}$ | $\{\pi_2, \sigma_2\}$ | $\{\pi_1, \sigma_2\}$ | $\{\pi_2, \sigma_1\}$ | $\{\pi_1, \pi_2\}$ |
|---|---|---|---|---|---|---|---|
| $h(n)$ (настоящая работа) | $n \log n$ | $n \log n$ | $n \log n$ | $n \log n$ | $\mathrm{C}_n^{n/2}$ | $\mathrm{C}_n^{n/2}$ | $n!$ |
| $h(n)$ [2] | $p(n)$ | $p(n)$ | $\mathrm{C}_n^{n/2}$ | $p(n)$ | $\mathrm{C}_n^{n/2}$ | $\mathrm{C}_n^{n/2}$ | $n!$ |

## ЛИТЕРАТУРА


1. *Agibalov G. P. and Pankratova I. A.* Asymmetric cryptosystems on Boolean functions // Прикладная дискретная математика. 2018. № 40. С. 23–33.
2. *Агибалов Г. П., Панкратова И. А.* Криптосистемы с открытым ключом на булевых функциях // Прикладная дискретная математика. Приложение. 2018. № 11. С. 54–57.
3. *Боровкова И. В., Панкратова И. А.* Криптоанализ шифрсистемы ACBF // Прикладная дискретная математика. Приложение. 2019. № 12. С. 90–93.
4. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для русского языка программирования // Прикладная дискретная математика. 2013. № 3(21). С. 93–104.
5. *Menezes A. J., Van Oorshot P. C., and Vanstone S. A.* Handbook of Applied Cryptography. N.Y.: CRC Press, 1997.


## REFERENCES


1. *Agibalov G. P. and Pankratova I. A.* Asymmetric cryptosystems on Boolean functions. Prikladnaya Diskretnaya Matematika, 2018, no. 40, pp. 23–33.
2. *Agibalov G. P. and Pankratova I. A.* Kriptosistemy s otkrytym klyuchom na bulevykh funktsiyakh [Public key cryptosystems on Boolean functions]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2018, no. 11, pp. 54–57. (in Russian)
3. *Borovkova I. V. and Pankratova I. A.* Kriptoanaliz shifrsistemy ACBF [Cryptanalysis of the ACBF encryption system]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2019, no. 12, pp. 90–93. (in Russian)
4. *Agibalov G. P., Lipskiy V. B., and Pankratova I. A.* O kriptograficheskom rasshirenii i ego realizatsii dlya russkogo yazyka programmirovaniya [Cryptographic extension and its implementation for Russian programming language]. Prikladnaya Diskretnaya Matematika, 2013, no. 3(21), pp. 93–104. (in Russian)
5. *Menezes A. J., Van Oorshot P. C., and Vanstone S. A.* Handbook of Applied Cryptography. N.Y., CRC Press, 1997.

УДК 519.1+519.719.2+519.712

# КРИПТОСИСТЕМА ШИФРОВАНИЯ С АУТЕНТИФИКАЦИЕЙ С ПРОИЗВОДНЫМИ РАЗОВЫМИ КЛЮЧАМИ

А. Ю. Зубов

*ООО «Центр сертификационных исследований», г. Москва, Россия*

Продолжается исследование предложенной автором математической модели криптосистемы шифрования с аутентификацией на основе кода аутентификации с секретностью. Алгоритм шифрования использует вычисления в полях характеристики два, счётчиковую последовательность, зависящую от ключа, одноразовые производные ключи, определяемые основным ключом и вектором инициализации с помощью ортогональных латинских квадратов, а также имитовставку полиномиального типа. Предлагается байтовый метод реализации алгоритма, проводится его сравнение со стандартизованным криптоалгоритмом GCM. Предлагается выбор параметров модели, гарантирующих доказуемую стойкость к атакам на основе шифртекста. Проводится анализ стойкости к атакам с выбранным открытым текстом.

**Ключевые слова:** *криптосистема шифрования с аутентификацией, GCM, квазигруппа, ортогональные латинские квадраты, доказуемая стойкость, атаки на основе шифртекста, атаки с выбранным открытым текстом.*

DOI 10.17223/20710410/50/3

# AUTHENTICATION ENCRYPTION CRYPTOSYSTEM WITH DERIVED ONE-TIME KEYS

A. Yu. Zubov

*Certification Research Center, Moscow, Russia*

**E-mail:** zubovanatoty@yandex.ru

A research of the previously proposed by the author mathematical model of authenticated encryption cryptosystem based on authentication code with secrecy is continued. An encryption algorithm uses calculations in the fields of characteristics two, a counters sequence depending on the key, one-time derived keys defined by the main key and initialization vectors using orthogonal Latin squares, and a polynomial-type MAC. A byte method for implementing the algorithm is proposed and compared with the standardized GCM cryptographic algorithm. The choice of model parameters that guarantee provable security to ciphertext-based attacks is proposed. The analysis of the cryptosystem's security to chosen-plaintext attacks is performed.

**Keywords:** *authenticated encryption cryptosystem, GCM, quasigroup, orthogonal Latin squares, provable security, ciphertext-based attacks, chosen-plaintext attacks.*

## 1. Базовая конструкция криптосистемы $\mathbb{CS}$

В [1] предложена модель криптосистемы $\mathbb{CS}$ шифрования с аутентификацией на основе кода аутентификации с секретностью. Настоящая работа продолжает начатые в [1] исследования этой модели. Приведём её описание.

Будем использовать обозначения $a$, $\overline{a}$, $\mathbf{a}$ для элемента поля $a \in \mathbb{F}_{2^n} = \mathrm{GF}(2^n)$, строки $a \in \{0,1\}^n$ коэффициентов многочлена, представляющего $a$ как элемент факторкольца $\mathbb{F}_2[x]/(h(x)) \cong \mathbb{F}_{2^n}$, и числа $\mathbf{a} \in \mathbb{Z}_{2^n}$, двоичная запись которого совпадает с $\overline{a}$.

Криптосистема $\mathbb{CS}$ использует для зашифрования открытых текстов $s^{(\tau)}$, $\tau = 1, 2, \ldots, N$, секретный ключ

$$k = (a, b) \in (\mathbb{F}_{2^n})^2,$$

векторы инициализации

$$iv^{(\tau)} = \left(\gamma^{(\tau)}, \delta^{(\tau)}\right) \in (\mathbb{F}_{2^n})^2,$$

отображения

$$f_x : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n},\ x \in (\mathbb{F}_{2^n})^2, \quad \varphi : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}, \quad g : (\mathbb{F}_{2^n})^4 \to (\mathbb{F}_{2^n})^2$$

и наборы констант

$$\{w_i \in \mathbb{F}_{2^n} : \mathbf{w}_i = (i + i_0) \pmod{2^n}\}, \quad \{c_i(k), d_i(k)\},$$
$$c_i(k) = f_k(w_i), \quad d_i(k) = \varphi(c_i(k)), \quad i = 1, 2, \ldots,$$

где $i_0 \geqslant 0$ — число, определяющее начальное значение счётчиковой последовательности $\{w_i\}$.

Вектор инициализации должен содержать счётчик, который делает вектор неповторяющимся, метку времени и атрибуты передаваемого сообщения. Текст $s^{(\tau)}$, представленный строкой

$$s^{(\tau)} = \left(s_1^{(\tau)}, \ldots, s_{r_\tau}^{(\tau)}\right)$$

элементов из $\mathbb{F}_{2^n}$, шифруется на производном ключе

$$k^{(\tau)} = \left(a^{(\tau)}, b^{(\tau)}\right) = g\left(k, iv^{(\tau)}\right) \in (\mathbb{F}_{2^n})^2.$$

Результат зашифрования $m^{(\tau)} = \left(m_1^{(\tau)}, \ldots, m_{r_\tau}^{(\tau)}, m_{r_\tau+1}^{(\tau)}\right)$ — это строка элементов из $\mathbb{F}_{2^n}$, где

$$m_i^{(\tau)} = s_i^{(\tau)} \oplus c_i(k) a^{(\tau)} \oplus d_i(k) b^{(\tau)}, \quad i = 1, 2, \ldots, r_\tau,\ r_\tau > 1; \tag{1}$$

$$m_{r_\tau+1}^{(\tau)} = a^{(\tau)} \oplus m_1^{(\tau)} \left(b^{(\tau)}\right)^{r_\tau} \oplus \ldots \oplus m_{r_\tau}^{(\tau)} \left(b^{(\tau)}\right)^1. \tag{2}$$

Отправитель посылает сообщение $(iv, m)$. Получатель сообщения $(iv, m)$, где $m = (m_1, \ldots, m_r, m_{r+1})$, вычисляет производный ключ

$$k' = (a', b') = g(k, iv).$$

Критерий аутентичности сообщения — равенство

$$m_{r+1} = a' \oplus m_1 \cdot b'^r \oplus \ldots \oplus m_r \cdot b'^1.$$

Если оно выполнено, то получатель вычисляет $c_i(k), d_i(k)$ и текст $s = (s_1, \ldots, s_r)$, где

$$s_i = m_i \oplus c_i(k) a' \oplus d_i(k) b', \quad i = 1, 2, \ldots, r.$$

При случайном выборе ключа $k$ не исключается случай, когда $k^{(\tau)} = (0, 0)$. Он нежелателен, поскольку все блоки гаммы становятся равными $0^n$. Избавиться от этого

недостатка можно следующим образом. При вычислении ключа $k^{(\tau)}$ вводится дополнительный шаг — проверка равенства $k^{(\tau)} = (0, 0)$. Если оно выполняется, то нужно сменить $iv^{(\tau)}$. Для этого, например, в $\overline{iv^{(\tau)}}$ выделить один бит и в случае, когда равенство выполняется, инвертировать этот бит.

На рис. 1 представлена упрощённая схема алгоритма $\mathbb{CS}$ (без использования ассоциированных данных и дополнения последнего неполного блока).

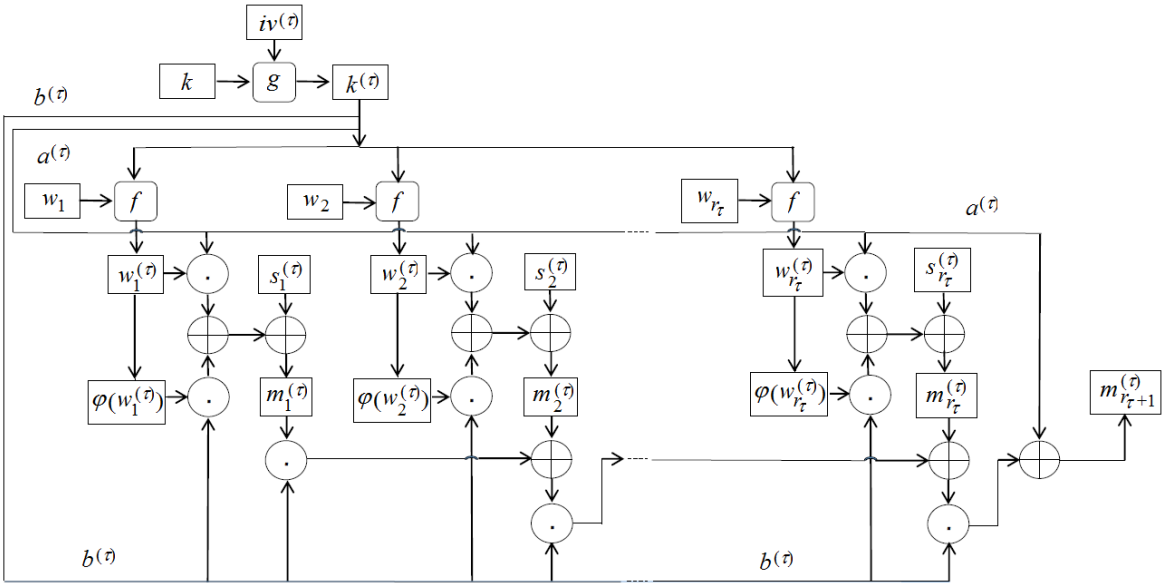


Рис. 1. Схема алгоритма $\mathbb{CS}$

Выбор отображений $\varphi, f_k, g$ диктуется следующими условиями:

a) $f_k$ инъективно;
b) $c_i(k)d_j(k) \neq c_j(k)d_i(k)$ при $i \neq j$;
c) разным $\tau$ отвечают разные $k^{(\tau)}$;
d) при случайном равновероятном выборе $k$ случайно равновероятно выбирается и $k^{(\tau)}$.

Требования (a)–(d) необходимы для получения оценок стойкости криптосистемы к атакам на основе шифртекста (см. далее п. 5).

## 2. Выбор отображений

В [1] предложен следующий выбор отображений $\varphi, f_k, g$ для $k = (a, b)$ и $iv = (\gamma, \delta)$:

$$\varphi(\vartheta) = \vartheta^2 \text{ либо } \varphi(\vartheta) = \vartheta \oplus \alpha,$$

где $\alpha$ — любой ненулевой элемент из $\mathbb{F}_{2^n}$;

$$f_k(w) = \Lambda\left(w * a\right) *' b; \quad g\left(k, iv\right) = \left(\Lambda(a * \gamma) *' a, \Lambda(b * \delta) *' b\right), \tag{3}$$

в которых используются квазигрупповые операции $*$, $*'$, определяемые значениями подходящих многочленов $P(x, y) \in \mathbb{Z}_{2^n}[x, y]$, и биективное преобразование $\Lambda : \mathbb{F}_{2^n} \to \to \mathbb{F}_{2^n}$.

Анализ выбора параметров алгоритма в варианте (3) обнаружил его недостатки. Так, оказалось, что отображение $g$, определённое формулой (3), может не гарантировать свойство (d), поскольку не всякие квазигрупповые операции указанного типа

обеспечивают равномерное распределение на множестве производных ключей. Кроме того, производные ключи достаточно сложно вычисляются. В связи с этим предлагается новый вариант, устраняющий указанные недостатки.

Определим $\varphi$ и $\Lambda$ так же, как в (3), и положим

$$f_k(w) = \Lambda\left(w * a\right) * b, \;\; g\left(k, iv\right) = \left(a *_{iv} b, \;\; a *'_{iv} b\right), \tag{4}$$

где $k = (a, b)$; $iv = (\gamma, \delta)$; $*_{iv}$, $*'_{iv}$ — операции на $\mathbb{F}_{2^n}$, определяемые ортогональными латинскими квадратами $L_1$, $L_2$ на $\mathbb{F}_{2^n}$ следующим образом.

Пусть $L_1$, $L_2$ — таблицы Кэли квазигрупп $(\mathbb{F}_{2^n}, *)$, $(\mathbb{F}_{2^n}, *')$ соответственно; $L_\alpha(x, y)$ — элемент латинского квадрата $L_\alpha$, расположенный в клетке $(x, y)$. Тогда положим

$$a^{(\tau)} = a *_{iv} b = L_1\left(\Lambda\left(a * \gamma\right), \Lambda\left(b * \delta\right)\right), \;\; b^{(\tau)} = a *'_{iv} b = L_2\left(\Lambda\left(a * \gamma\right), \Lambda\left(b * \delta\right)\right). \tag{5}$$

В (4) предлагается использовать ту же операцию $*$, что и в (5).

Несложно видеть, что для любого вектора инициализации $iv$ таблицы Кэли операций $*_{iv}$ и $*'_{iv}$, определённых формулами (5), являются ортогональными латинскими квадратами на $\mathbb{F}_{2^n}$. Это следует из того, что $L_1$, $L_2$ — ортогональные латинские квадраты и $\Lambda$ — биективное преобразование поля $\mathbb{F}_{2^n}$. Далее мы уточним выбор $L_1$, $L_2$.

Заметим, что вариант (4) выбора отображений гарантирует выполнение свойств (a)–(d). Свойства (a), (b) обеспечиваются выбором отображений $\varphi$ и $f_k$. Пусть $*_{iv}$,$*'_{iv}$ — операции, определённые формулами (5). Тогда при любых $u, v \in \mathbb{F}_{2^n}$ система уравнений

$$\begin{cases} a *_{iv} b = u, \\ a *'_{iv} b = v \end{cases} \tag{6}$$

имеет единственное решение $(a, b)$ для любого $iv$. Отсюда следует свойство (d). Из формул (5) также очевидно, что разным векторам инициализации $iv^{(\tau)}$ и $iv^{(\tau')}$ отвечают разные производные ключи $k^{(\tau)}$ и $k^{(\tau')}$, поэтому выполняется условие (c).

## 3. Байтовый метод

Рассмотрим следующий вариант квазигрупповых операций в формулах (4) и (5).

Пусть $x, y$ — элементы поля $\mathbb{F}_{2^n}$ и $\overline{x}$, $\overline{y}$ — соответствующие им векторы из $\{0,1\}^n$. Представим $\overline{x}, \overline{y}$ в виде последовательности байтов $\overline{x} = \overline{x}_1 || \ldots || \overline{x}_{n/8}$, $\overline{y} = \overline{y}_1 || \ldots || \overline{y}_{n/8}$. Будем интерпретировать байты $\overline{x}_i$ и $\overline{y}_i$ как элементы $x_i, y_i$ поля $\mathbb{F}_{2^8} = \mathrm{GF}(2)/(x^8 + x^7 + x^6 + x + 1)$ и использовать запись $x = \left(x_1, \ldots, x_{n/8}\right)$, $y = \left(y_1, \ldots, y_{n/8}\right)$.

Пусть $\overline{*}$ — квазигрупповая операция на $\mathbb{F}_{2^8}$. Определим на $\mathbb{F}_{2^n}$ операцию $*$ как

$$x * y = \left(x_1 \,\overline{*}\, y_1, \ldots, x_{n/8} \,\overline{*}\, y_{n/8}\right). \tag{7}$$

Ясно, что $*$ является квазигрупповой операцией на $\mathbb{F}_{2^n}$.

В (4) произведение $w_i * a$ будем вычислять в соответствии с (7), а $f_k(w)$ — следующим образом. Пусть $\overline{\Lambda}$ — матрица размера $n/8 \times n/8$ над $\mathbb{F}_{2^8}$, обладающая хорошими рассеивающими свойствами. Например, при $n = 128$ в качестве $\overline{\Lambda}$ можно выбрать максимально рассеивающую матрицу, используемую в шифрсистеме «Кузнечик». Пусть $y = w_i * a = \left(y_1, \ldots, y_{n/8}\right)$ и

$$z = (z_1, \ldots, z_8) = \left(y_1, \ldots, y_{n/8}\right) \overline{\Lambda}. \tag{8}$$

Тогда $f_k(w) = z * b$ будем вычислять в соответствии с формулой (7).

В описании байтового метода остаётся объяснить выбор квазигрупповой операции $\overline{*}$.

Построим пару ортогональных латинских квадратов $M_1$, $M_2$ на $\mathbb{F}_{2^8}$, используя, например, метод Боуза [2], который состоит в следующем: пусть

$$\mathbb{F}_{2^t} = \{\alpha_0 = 0, \alpha_1 = 1, \alpha_2, \ldots, \alpha_{2^t-1}\}.$$

Тогда матрицы

$$M_r = \left(\alpha_{i,j}^{(r)}\right), \quad r = 1, \ldots, 2^t - 1,$$

где

$$\alpha_{i,j}^{(r)} = \alpha_i \cdot \alpha_r \oplus \alpha_j, \quad i, j = 0, 1, \ldots, 2^t - 1,$$

образуют полную систему ортогональных латинских квадратов на $\mathbb{F}_{2^t}$. Произвольную пару таких квадратов при $t = 8$ выберем в качестве $M_1, M_2$. Вместе с ними можем получить порядка $(256!)^3$ других пар $M_1^{(\pi)}, M_2^{(\pi)}$ ортогональных латинских квадратов путём одинаковых перестановок в $M_1, M_2$ строк, столбцов и перенумераций элементов. Каждая из пар $M_1^{(\pi)}, M_2^{(\pi)}$ определяет квазигрупповые операции $\overline{*}$ и $\overline{*}'$ на $\mathbb{F}_{2^8}$.

Недостатком латинских квадратов $M_1, M_2$, построенных методом Боуза, является достаточно простое алгебраическое задание соответствующих операций $\overline{*}$ и $\overline{*}'$, позволяющее аналитическое исследование связи между основным и производным ключами. Поэтому предлагается выбрать пару латинских квадратов $M_1^{(\pi)}, M_2^{(\pi)}$, для которых операции $\overline{*}$ и $\overline{*}'$ не будут обладать «хорошими» алгебраическими свойствами, например свойствами коммутативности, ассоциативности, дистрибутивности относительно $\oplus$ и т. д. Это сделать нетрудно. Для наших целей целесообразно также выбирать такую операцию $\overline{*}$, чтобы функция $F : (\mathbb{F}_{2^8})^2 \to \mathbb{F}_{2^8}$, определённая формулой $F(u, v) = u\overline{*}v$, имела как можно большую нелинейность в том смысле, как это принято в теории дискретных функций. Её выбор представляет предмет отдельного исследования. Отметим, что число возможных пар $M_1^{(\pi)}, M_2^{(\pi)}$ делает практически невозможной попытку восстановления исходных латинских квадратов $M_1, M_2$ по выбранным латинским квадратам $M_1^{(\pi)}, M_2^{(\pi)}$.

После того как сделан выбор $M_1^{(\pi)}, M_2^{(\pi)}$, используем соответствующие им операции $\overline{*}$ и $\overline{*}'$ для определения операций $*, *'$ по формуле (7). В качестве $L_1, L_2$ в формуле (5) будем использовать таблицы Кэли операций $*, *'$. Для хранения и использования выбранных квадратов потребуется память объёма $2 \cdot 256^2$ байтов.

Недостатком байтового метода является необходимость использования достаточно большого объёма памяти в 131072 байта. Значительно меньший объём памяти потребуется для «полубайтового» метода, который строится аналогично байтовому путём замены байтов на полубайты. В этом случае объём памяти уменьшается до 512 байтов.

## 4. Анализ эффективности реализации алгоритма $\mathbb{CS}$

Преимущество байтового (полубайтового) метода состоит в том, что вычисления в $\mathbb{F}_{2^8}$ (соответственно в $\mathbb{F}_{2^4}$) реализуются значительно проще, чем в $\mathbb{F}_{2^n}$ при $n \geqslant 64$. Кроме того, необходимые операции задаются таблично. Например, при $n = 128$ для этого потребуется память объёма 131328 байтов для байтового и 768 байтов для полубайтового методов для записи двух латинских квадратов $M_1^{(\pi)}, M_2^{(\pi)}$ и матрицы $\overline{\Lambda}$. Обращения в такую память при выполнении операций алгоритма сравнимы по сложности с булевыми операциями. Единственными «сложными» операциями алгоритма $\mathbb{CS}$ в таком случае остаются операции умножения в поле $\mathbb{F}_{2^n}$ в формулах (1) и (2). Число

операций умножения в поле $\mathbb{F}_{2^n}$, выполняемых алгоритмом $\mathbb{CS}$, сопоставимо с числом аналогичных операций стандартизованного алгоритма GCM [3, 4].

Чтобы убедиться в этом, приведём схему алгоритма GCM (рис. 2) и оценим сложность двух алгоритмов в сопоставимых операциях, пользуясь при реализации $\mathbb{CS}$ формулами (5) в качестве метода вычисления производного ключа и байтовым методом.

Рассмотрим лишь упрощённые схемы алгоритмов (не использующие ассоциированные данные), полагая, что открытые тексты состоят из полных $n$-битовых блоков.

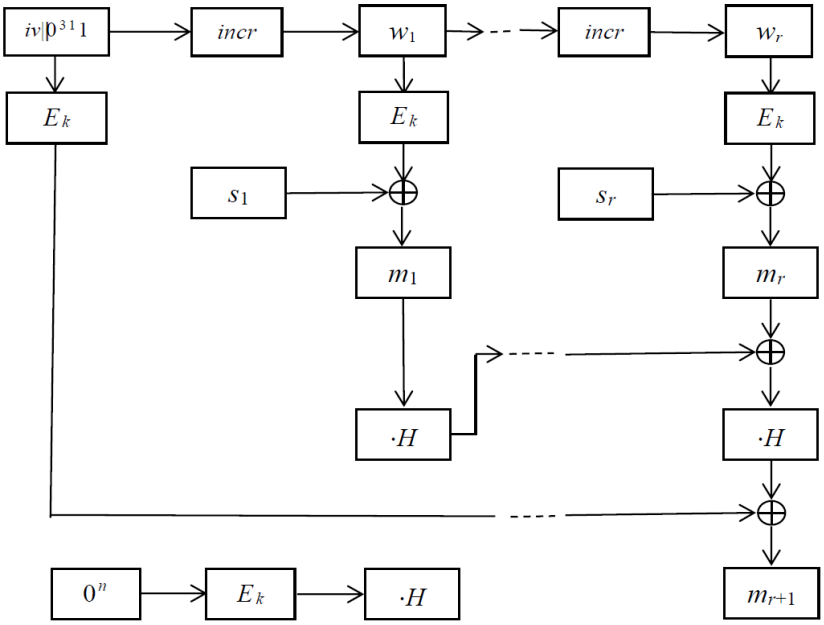


Рис. 2. Схема алгоритма GCM

Пусть $T_g, T_f, T_\oplus, T_\odot, T_\ll, T_P$ — соответственно трудоёмкости выполнения операций $g_y, f_x, \oplus, \odot, \ll$ и обращения в память, где $\oplus$ — операция XOR; $\odot$ — операция умножения в поле $\mathbb{F}_{2^n}$; $\ll$ — операция сдвига двоичного вектора.

В случае, когда $\varphi(\vartheta) = \vartheta^2$, трудоёмкость $T_{\mathbb{CS}}$ алгоритма зашифрования сообщения $s$, состоящего из $r$ блоков, выражается формулой

$$T_{\mathbb{CS}} = [T_g + rT_f + 2rT_\oplus + 3rT_\odot] + r\,(T_\oplus + T_\odot)\,, \tag{9}$$

где слагаемое в квадратных скобках — трудоёмкость получения $r$ блоков шифртекста, а второе слагаемое — трудоёмкость получения имитовставки.

Из схемы алгоритма GCM находим аналогичную величину для $T_{\mathrm{GCM}}$:

$$T_{\mathrm{GCM}} = [(r+2)\,T_E + rT_\oplus] + r\,(T_\oplus + T_\odot)\,. \tag{10}$$

Здесь $T_E$ — трудоёмкость зашифрования одного блока данных для GCM.

Для сравнения $T_{\mathbb{CS}}$ и $T_{\mathrm{GCM}}$ сделаем грубую прикидку. Сошлёмся на работы [3, 4], в которых указывается, что в случае, когда GCM использует для шифрования AES, имеет место соотношение $T_E \approx 3T_\odot$. В свою очередь будем полагать, что операция $\odot$ равносильна $3n$ операциям типа $\ll$ и $\oplus$. Это можно пояснить следующим образом.

Пусть, например, $\mathbb{F}_{2^{128}}$ порождается многочленом $x^{128}+x^7+x^2+x+1$. Тогда для $\overline{\alpha}=(\alpha_{127},\ldots,\alpha_1,\alpha_0)$ произведение $\alpha\cdot x$ можно вычислить, пользуясь формулой

$$\overline{\alpha\cdot x}=\begin{cases}\overline{\alpha}\ll 1, & \text{если } \alpha_{127}=0,\\ (\overline{\alpha}\ll 1)\oplus 0^{120}||10000111, & \text{если } \alpha_{127}=1.\end{cases}$$

Представив $\overline{\beta}=(\beta_{127},\ldots,\beta_1,\beta_0)$ в виде многочлена $\beta_{127}x^{127}+\ldots+\beta_1x+\beta_0$, вычислим произведение $\alpha\cdot\beta$ не более чем за $3\cdot 128$ операций указанного типа.

Заметим, что произведение (8) вычисляется за $n/8$ операций обращения в память и $n/8-1$ операций XOR. С учётом этого из (4), (5), (7), (8) получаем

$$T_g=(n-4)T_P+(n/4)T_{\oplus}; \tag{11}$$

$$T_f=(3n/8)\,T_P+(n/8-1)\,T_{\oplus}. \tag{12}$$

Будем также полагать, что $T_{\oplus},T_{\ll},T_P$ равносильны (по сложности или времени выполнения). Поскольку основные операции алгоритмов выражаются через эти операции, будем каждую из них считать элементарной. Выразим $T_{\mathbb{CS}}$ и $T_{GCM}$ в элементарных операциях. Непосредственно из (9)–(12) получаем

$$T_{\mathbb{CS}}=12{,}5rn+1{,}25n+2r-5,\quad T_{\mathrm{GCM}}=12rn+18n+2r,$$

или приближённо (с учётом того, что $r$ достаточно велико)

$$T_{\mathbb{CS}}\approx 12{,}5rn,\quad T_{\mathrm{GCM}}\approx 12rn.$$

Отметим возможность повышения эффективности алгоритма $\mathbb{CS}$ за счёт использования дополнительной памяти для хранения констант $c_i(k),d_i(k)$.

Пусть необходимо зашифровать ряд сообщений длины $r$ (где $r$ не слишком велико) с помощью основного ключа $k$. Будем записывать вычисленные при зашифровании первого сообщения константы $c_i(k),d_i(k)$, $i=1,2,\ldots,r$, в память объёма $2rn$ битов, а при шифровании второго и следующего сообщений обращаться в эту память, извлекая из неё необходимую информацию. Такой способ позволит сэкономить за счёт того, что обращение в память значительно проще, чем вычисление констант $c_i(k),d_i(k)$.

Аналогичный расчёт для случая, когда в алгоритме $\mathbb{CS}$ используется отображение $\varphi(\vartheta)=\vartheta+\alpha$, $\alpha\in\mathbb{F}_{2^n}$, даёт следующие результаты:

$$T_{\mathbb{CS}}=9{,}5rn,\quad T_{\mathrm{GCM}}=12rn.$$

Конечно, при сравнении величин $T_{\mathbb{CS}}$ и $T_{\mathrm{GCM}}$ необходимо учитывать поправку на то, что при реализации $\mathbb{CS}$ операция умножения $\odot$ «сто́ит» несколько «дороже», чем при реализации GCM. Дело в том, что при шифровании каждого сообщения с помощью $\mathbb{CS}$ приходится вычислять произведения элементов поля $\mathbb{F}_{2^n}$ на компоненты $a^{(\tau)},b^{(\tau)}$ очередного производного ключа $k^{(\tau)}$, а для GCM — лишь произведения элементов поля на один и тот же ключ аутентификации $H=E_k(\overline{0})$.

Более точное сравнение алгоритмов возможно лишь для конкретных реализаций.

## 5. Анализ стойкости криптосистемы $\mathbb{CS}$

Рассмотрим возможность определения ключа $k=(a,b)$ криптосистемы $\mathbb{CS}$ в атаке с выбранным открытым текстом. Ключ $k$ находится из системы уравнений

$$\begin{cases}c_i(k)\cdot a^{(\tau)}\oplus d_i(k)b^{(\tau)}=u_i^{(\tau)},\\ a^{(\tau)}\oplus m_1^{(\tau)}\left(b^{(\tau)}\right)^{r_\tau}\oplus\ldots\oplus m_{r_\tau}^{(\tau)}\left(b^{(\tau)}\right)^{1}=m_{r_\tau+1}^{(\tau)},\\ i=1,2,\ldots,r_\tau,\ \tau=1,2,\ldots,L,\end{cases} \tag{13}$$

в которой $u_i^{(\tau)} = s_i^{(\tau)} \oplus m_i^{(\tau)}$, $a^{(\tau)}$, $b^{(\tau)}$ определяются формулами (5), $c_i(k) = \Lambda\,(w * a) * b$, $d_i(k) = (c_i(k))^2$ либо $d_i(k) = c_i(k) \oplus \alpha$. Несмотря на то, что эта система содержит много уравнений и всего две переменные ($a$ и $b$), её аналитическое решение не представляется возможным в силу разнородности используемых операций.

Вид уравнений системы (13) приводит к естественной задаче нахождения из этой системы хотя бы одного производного ключа. Если эта задача будет успешно решена, то это облегчит нахождение основного ключа. Рассмотрим этот подход.

Находить производный ключ $k^{(\tau)} = \left(a^{(\tau)}, b^{(\tau)}\right)$ можно перебором возможных значений $b^{(\tau)}$, вычислением из второго уравнения соответствующего $a^{(\tau)}$ и проверкой получившегося допустимого варианта $k^{(\tau)}$ с помощью первого уравнения. Сложность этого вычисления оценивается величиной $O(2^n)$. Проверка допустимого варианта невозможна, если неизвестны $c_i(k)$. Чтобы найти их, требуется ключ $k$. Но если при этом, имея допустимый вариант $k^{(\tau)}$, удастся найти $k$ с помощью метода, основанного на взаимной зависимости констант $c_i(k), c_j(k)$, с трудоёмкостью меньшей $O(2^n)$, то можно получить метод определения ключа со сложностью меньше полного перебора ключей. Покажем, что использование в формуле $c_i(k) = \Lambda\,(w * a) * b$ рассеивающего преобразования $\Lambda$ устраняет указанную возможность при использовании байтового метода.

Пусть константа $c_i(k)$ определяется формулой $c_i(k) = (w * a) * b$. Тогда заметим, что если векторы $\overline{w}_i$ и $\overline{w}_{i+1}$ отличаются лишь одним байтом, то и векторы $\overline{f_k\,(w_i)}, \overline{f_k\,(w_{i+1})}$ также отличаются лишь одним байтом, т. е. сумма $f_k\,(w_i) \oplus f_k\,(w_{i+1}) = w$ принадлежит известному множеству, мощность которого не превосходит $2^8$. В случае, когда $\varphi(\vartheta) = \vartheta^2$, из первого уравнения системы (13) получаем равенства

$$f_k\,(w_i)\,a^{(\tau)} \oplus f_k^2\,(w_i)\,b^{(\tau)} = u_i^{(\tau)},$$
$$f_k\,(w_{i+1})\,a^{(\tau)} \oplus f_k^2\,(w_{i+1})\,b^{(\tau)} = u_{i+1}^{(\tau)},$$

складывая которые, получаем $w \cdot a^{(\tau)} \oplus w^2 \cdot b^{(\tau)} = u$, где $u = u_i^{(\tau)} \oplus u_{i+1}^{(\tau)}$. Аналогично, при $j \neq i$ получаем $w' \cdot a^{(\tau)} \oplus w'^2 \cdot b^{(\tau)} = u'$, где $u' = u_j^{(\tau)} \oplus u_{j+1}^{(\tau)}$, а $w'$ принадлежит известному множеству, мощность которого не превосходит $2^8$. В результате имеем систему уравнений

$$\begin{cases} w \cdot a^{(\tau)} \oplus w^2 \cdot b^{(\tau)} = u, \\ w' \cdot a^{(\tau)} \oplus w'^2 \cdot b^{(\tau)} = u' \end{cases}$$

относительно $a^{(\tau)}, b^{(\tau)}$ с частично известными $w, w'$. Перебирая не более чем $2^{16}$ возможных значений пары $\left(w, w^{'}\right)$, мы можем определить ключ $k^{(\tau)}$, решая эту систему.

Аналогично в случае, когда $\varphi(\vartheta) = \vartheta \oplus \alpha$, $\alpha \in \mathbb{F}_{2^n}$, из первого уравнения системы (13) следуют равенства

$$f_k\,(w_i)\,a^{(\tau)} \oplus (f_k\,(w_i) \oplus \alpha)\,b^{(\tau)} = u_i^{(\tau)},$$
$$f_k\,(w_{i+1})\,a^{(\tau)} \oplus (f_k\,(w_{i+1}) \oplus \alpha)\,b^{(\tau)} = u_{i+1}^{(\tau)}.$$

Полагая

$$f_k\,(w_i) \oplus f_k\,(w_{i+1}) = w, \quad u = u_i^{(\tau)} \oplus u_{i+1}^{(\tau)},$$

получаем равенство

$$w\left(a^{(\tau)} \oplus b^{(\tau)}\right) = u$$

(с частично известным $w$). Выражая $a^{(\tau)}$ через $b^{(\tau)}$, из (13) получаем полиномиальное уравнение относительно $b^{(\tau)}$. Хотя такое уравнение может не иметь эффективного решения, мы получаем полезную информацию о производном ключе.

Использование рассеивающего преобразования $\Lambda$ в формуле $c_i(k) = \Lambda(w * a) * b$ делает описанный метод неприменимым, поскольку в таком случае каждый байт вектора $\overline{\Lambda(w * a)}$ будет зависеть от каждого байта вектора $\overline{w * a}$ (как при максимально рассеивающем преобразовании в шифрсистеме «Кузнечик»). Отметим, что по той же причине для «разрушения» байтовой структуры рассеивающее преобразование применяется и в формуле (5). Использование неявной зависимости констант $c_i(k)$ и $c_j(k)$ иным образом не представляется возможным.

Предположим, что тем или иным образом стал известен ключ $k^{(\tau)}$. Насколько сложно тогда получить ключ $k$? Поскольку компоненты ключа $k^{(\tau)}$ вычисляются по формуле (5) как элементы $a^{(\tau)} = L_1(a, b)$, $b^{(\tau)} = L_2(a, b)$ заданных таблично ортогональных латинских квадратов $L_1, L_2$, найти ключ $k$ можно лишь следующим образом. Будем перебирать возможные значения одной из переменных, например $a$, и находить соответствующие значения $b$ из (6). Средняя сложность такого метода оценивается величиной $O(2^n)$. Учитывая сложность определения ключа $k^{(\tau)}$, можно сделать вывод, что предложенный метод выбора производного ключа не опасен.

$\mathbb{CS}$ является блочной криптосистемой. Применимы ли к ней известные методы анализа блочных шифров?

Линейный и разностный методы используются для анализа итерационных блочных шифров. Для их реализации требуется значительное число известных пар блоков открытых и шифрованных текстов, полученных в результате шифрования на общем ключе. Та же ситуация имеет место при атаках различения. Для анализа многораундовых шифров применяются также атаки со связанными ключами. Известные версии всех этих методов не работают в нашем случае, поскольку $\mathbb{CS}$ построена не по итерационному принципу и каждое сообщение шифруется на своём ключе, а связь между ключами выражается сложной неявной зависимостью.

В алгебраических атаках исследуется зависимость между битами блоков открытых и шифрованных текстов, полученная путём линеаризации нелинейных перемешивающих преобразований $S$-боксов. Задача нахождения ключа сводится к составлению и решению системы уравнений, число неизвестных в которой растёт экспоненциально от размеров $S$-боксов. В нашем случае потребуется, в частности, линеаризовать нелинейное перемешивающее преобразование $(\mathbb{F}_{2^8})^2 \to \mathbb{F}_{2^8}$, реализуемое квазигрупповой операцией $\overline{*}$ (аналог $S$-бокса). Число неизвестных в соответствующей системе уравнений будет непомерно велико.

Приведённый анализ позволяет сделать вывод о том, что предлагаемые варианты выбора параметров криптосистемы $\mathbb{CS}$ не имеют «видимых» слабостей в различных вариантах атак с выбранным открытым текстом. Более детальный анализ шифрсистемы возможен лишь при конкретном выборе параметров ортогональных латинских квадратов $L_1, L_2$, рассеивающего преобразования $\Lambda$ и соотношения, связывающего константы $c_i(k)$ и $d_i(k)$.

Каковы возможности проведения противником активных атак? Как, наблюдая ряд аутентифицированных сообщений, построить подделку, т. е. новое сообщение, снабжённое корректной имитовставкой?

Для этого нужно хотя бы что-то знать о следующем производном ключе, например то, что он — тот же, что и предыдущий, иначе корректную имитовставку можно лишь угадывать. Но в силу того, что для $\mathbb{CS}$ выполняется условие (d), для противника следующий производный ключ выбирается случайно равновероятно из всего множества возможных ключей. Даже имея перехват многих сообщений, зашифрованных на общем основном ключе, противник не сможет ограничить для себя выбор следующего

производного ключа. Зная открытые и шифрованные тексты ряда сообщений, можно попытаться вычислить текущий производный ключ из системы уравнений (13). Но, как уже отмечено, эта задача сложна, даже если противник имеет возможность подбирать открытые тексты. Таким образом, успех в активной атаке возможен лишь в тех случаях, когда противник умеет определять производные ключи (ещё лучше, если основной ключ). Отсюда следует, что для проведения успешных активных атак противник должен уметь решать те же задачи, что и при проведении пассивных атак.

Рассмотрим теперь вопрос об оценке стойкости криптосистемы $\mathbb{CS}$ к атакам на основе шифртекста. Речь идёт о возможности решения противником следующих задач:

**Задача 1**: по шифртексту произвольного сообщения найти его открытый текст.

**Задача 2**: подменить перехваченное шифрованное сообщение таким образом, чтобы модифицированное сообщение было принято получателем как аутентичное.

**Задача 3**: составить такое поддельное шифрованное сообщение и имитировать его передачу от легального пользователя, чтобы оно было принято получателем как аутентичное.

В этих задачах криптосистема $\mathbb{CS}$ может рассматриваться как код аутентификации с секретностью при одноразовом использовании секретного ключа, поскольку противник не имеет практической возможности учесть взаимную зависимость производных ключей, обладая лишь перехватом шифртекстов. В этой постановке указанные задачи решались в [5, 6]. Результат решения задач 1–3 сформулирован в [6]. В [1] этот результат продублирован для варианта криптосистемы (теорема 2). Приведём здесь эти результаты, которые имеют место для каждого из рассмотренных в данной статье вариантов выбора параметров криптосистемы $\mathbb{CS}$, дополнив их ещё одной оценкой.

Обозначим через $p_{\mathrm{im}}$ и $p_{\mathrm{sub}}$ вероятности успеха имитации и подмены (т. е. вероятности успешного решения задач 3 и 2 соответственно), а $p_{\mathrm{im}}^{(N)}$ — вероятность успеха имитации при условии, что противник наблюдает сообщения $m^{(1)}, \ldots, m^{(N)}$, зашифрованные с помощью общего основного ключа $k$.

**Теорема 1.** Пусть ключ $k$ криптосистемы $\mathbb{CS}$ выбирается случайно равновероятно, а распределение вероятностей $\mathcal{P}_X$ на множестве открытых текстов $\mathcal{X}$ таково, что для действительных чисел $\alpha, \beta, \sigma$, удовлетворяющих неравенствам $0 < \alpha$, $\beta < 1 \leqslant \sigma$, выполняются условия $\alpha \leqslant p_X(x) \leqslant \beta$, $\beta/\alpha \leqslant \sigma$ для каждого $x \in \mathcal{X}$. Тогда $\mathbb{CS}$ реализует $\varepsilon$-совершенное шифрование очередного текста $s^{(\tau)}$, $\tau \geqslant 1$, состоящего из $r^{(\tau)} > 1$ блоков, для $\varepsilon < \sigma 2^{-n}$. При этом $p_{\mathrm{im}} = 2^{-n}$, $p_{\mathrm{sub}} \leqslant r^{(\tau)}\sigma 2^{-n}$. Кроме того, при $N < 2^n$ справедлива оценка $p_{\mathrm{im}}^{(N)} \leqslant 2^n / (2^{2n} - N)$.

Приведённая в теореме 1 оценка величины $p_{\mathrm{im}}^{(N)}$ дополняет утверждение теоремы 2 в [1]. Она следует из того, что рассматриваемые варианты выработки производного ключа гарантируют свойство (c).

Теорема 1 свидетельствует о доказуемой стойкости криптосистемы $\mathbb{CS}$ к атакам на основе шифртекста, так как при подходящих значениях $n, r, \sigma$ гарантируется как $\varepsilon$-совершенное шифрование, так и достаточно малые значения вероятностей успеха активных атак. Так, при $n = 128$, $\sigma = 2^{32}$, $r \leqslant 2^{32}$ криптосистема реализует $\varepsilon$-совершенное шифрование текстов длины не больше $2^{32}$ блоков для $\varepsilon < 2^{-96}$ и их имитозащиту со значениями $p_{\mathrm{im}} = 2^{-128}$, $p_{\mathrm{sub}} < 2^{-64}$. В частности, $p_{\mathrm{im}}^{(N)} < 1/(2^{128} - 1)$ при $N = 2^{128} - 1$.

Получение этих оценок для криптосистем, реализующих шифрование информации и её аутентификацию, оказалось возможным вследствие того, что для шифрования используется не традиционная блочная шифрсистема, а параметризованная ключом

система отображений, порождённая комбинаторной структурой, в данном случае — кодом Рида — Соломона.

В заключение отметим, что предложенные варианты параметров модели криптосистемы выбирались, исходя из соображений разумной минимизации их сложности, не приводящей к «видимым» криптографическим недостаткам. Можно предложить варианты усиления схемы, усложняющие задачу нахождения ключа в атаке с выбранным открытым текстом, для которых остаются в силе оценки сформулированной теоремы.

## ЛИТЕРАТУРА

1. *Зубов А. Ю.* Криптосистема шифрования с аутентификацией на основе кода аутентификации с секретностью // Прикладная дискретная математика. 2019. № 43. С. 60–69.
2. *Сачков В. Н.* Введение в комбинаторные методы дискретной математики. М.: МЦНМО, 2004. 424 с.
3. *McGrew D. and Viega J.* The security and performance of Galois/Counter mode of operation // LNCS. 2004. V. 3348. P. 343–355.
4. *Viega J. and McGrew D.* Galois/Counter mode (GCM) overview fibre channel security protocols. Cisco Systems Inc., 2005. 32 p.
5. *Зубов А. Ю.* Почти совершенные шифры и коды аутентификации // Прикладная дискретная математика. 2011. № 4 (14). С. 28–33.
6. *Зубов А. Ю.* О понятии $\varepsilon$-совершенного шифра // Прикладная дискретная математика. 2016. № 3 (33). С. 45–52.

## REFERENCES

1. *Zubov A. Yu.* Kriptosistema shifrovaniya s autentifikatsiey na osnove koda autentifikatsii s sekretnost'yu [Authentication encryption based on authentication code with secrecy]. Prikladnaya Diskretnaya Matematika, 2019, no. 43, pp. 60–69. (in Russian)
2. *Sachkov V. N.* Vvedenie v kombinatornye metody diskretnoy matematiki [Introduction to Combinatorial Methods of Discrete Mathematics]. Moscow, MCCME Publ., 2004. 424 p. (in Russian)
3. *McGrew D. and Viega J.* The security and performance of Galois/Counter mode of operation. LNCS, 2004, vol. 3348, pp. 343–355.
4. *Viega J. and McGrew D.* Galois/Counter mode (GCM) overview fibre channel security protocols. Cisco Systems Inc., 2005. 32 p.
5. *Zubov A. Yu.* Pochti sovershennye shifry i kody autentifikatsii [Almost perfect ciphers and authentication codes]. Prikladnaya Diskretnaya Matematika, 2011, no. 4 (14), pp. 28–33. (in Russian)
6. *Zubov A. Yu.* O ponyatii $\varepsilon$-sovershennogo shifra [On the concept of a $\varepsilon$-perfect cipher]. Prikladnaya Diskretnaya Matematika, 2016, no. 3 (33), pp. 45–52. (in Russian)

UDC 519.7 DOI 10.17223/20710410/50/4

# PROBLEMS IN THEORY OF CRYPTANALYTICAL INVERTIBILITY OF FINITE AUTOMATA

G. P. Agibalov

*National Research Tomsk State University, Tomsk, Russia*

**E-mail:** agibalov@mail.tsu.ru

The paper continues an investigation of the cryptanalytical invertibility concept of finite automata with a finite delay introduced by the author in his previous papers where he also gave a constructive set theory test for an automaton $A$ to be cryptanalytically invertible, that is, to have a recovering function $f$ which allows to calculate a prefix of a length $m$ in an input sequence of the automaton $A$ by using its output sequence of a length $m+\tau$ and some additional information about $A$ known to cryptanalysts, defining a type of its invertibility and of its recovering functon. Here, we expound a test for that of another kind, namely some logical necessary and sufficient conditions for an automaton $A$ to have or not a recovering function $f$ of a certain type. Results related to specific types of automata invertibility (invertibility tests, inversion algorithms, synthesis of inverse automata and others) are subjects of further researching and publications.

**Keywords:** *finite automata, information-lossless automata, automata invertibility, recovering function, cryptanalytical invertibility, cryptanalytical invertibility conditions.*

## 1. Introduction

The finite automata invertibility with a finite delay is studied from a cryptanalyst's point of view, namely in dependence on a priori information accessible to an inversion algorithm. In cryptanalysis of symmetric finite automata ciphers by attack with known ciphertext, the case, when there is a need for solution to an automaton inversion problem by a partially informed cryptanalyst, is very typical. The cryptanalysts can or can not know some information about the transfer or output functions of the automaton, about its initial, intermediate or final state, a part of input word, playing a service role, its length or the place in the input sequence and others. The variety of information, which can be known to a cryptanalyst, provides many different types of the automaton invertibility and many different classes of invertible automata. In this paper, it is assumed that the transfer and output functions of the automaton under consideration are known completely or up to accuracy of their classes and the cryptanalysis problem is to recover the prefix of input word.

The automaton cryptanalytic invertibility with a finite delay plays a very important role in the analysis and synthesis of finite automata cryptographic systems. In this paper it is studied with a delay denoted by an integer $\tau$. From the cryptanalyst's point of view, this notion means the theoretical possibility for recovering, under some conditions, any prefix $\alpha$ of a length $m$ in an unknown input sequence $\alpha\delta$ of an automaton from its output sequence $\gamma$ of the length $m+\tau$ and perhaps an additional information such as parameters $\tau$ and $m$, initial, intermediate or final states of the automaton or the suffix $\delta$ of the length $\tau$ in the input sequence. The conditions imposed on the recovering algorithm require for prefix $\alpha$ to

be arbitrary and may require for the initial state $q$ and suffix $\delta$ to be arbitrary or existent, that is, the variable $\alpha$ is always bound by the universal quantifier and each of variables $q$ and $\delta$ may be bound by any of quantifiers — universal ($\forall$) or existential ($\exists$) one. The variety of information, which can be known to a cryptanalyst, provides many different types of the automaton invertibility and, respectively, many different classes of invertible automata.

Thus, in the paper, an invertibility with a finite delay $\tau$ of a finite automaton $A$ is a property of this automaton that allows to precisely determine any input word $\alpha$ of a length $m$ for the output word $\gamma$ being the result of transforming by the automaton $A$ at its initial state $q$ the input word $\alpha\delta$ with the delay word $\delta$ of length $\tau$ and with the known $m, \tau, A, \gamma$ and a subset $\upsilon$ of the set, consisting of the delay word $\delta$ and initial, intermediate and final states $q, \psi(\alpha, q), \psi(\alpha\delta, q)$, to which $A$ transits from $q$ under acting of input words $\alpha$ and $\alpha\delta$ respectively and where $q$ and $\delta$ may be arbitrary or some elements in their sets. According to this, the automaton $A$ is called invertible with a delay $\tau$ if there exists a function $f(\gamma, \upsilon)$ and a triplet of quantifiers $\kappa \in \{K_1x_1K_2x_2K_3x_3 : K_ix_i \in \{\forall q, \exists q, \forall\alpha, \forall\delta, \exists\delta\}, i \neq j \Rightarrow$ $\Rightarrow x_i \neq x_j\}$ such that $\kappa(f(\gamma, \upsilon) = \alpha)$; in this case $f$ is called a recovering function, $\kappa$ — an invertibility type, $\upsilon$ — an invertibility order of the automaton $A$ and $\exists f\kappa(f(\gamma, \upsilon) = \alpha)$ — an invertibility condition for the automaton $A$. The most known invertibility conditions for finite automata described earlier as strong and weak by scientists D. A. Huffman, A. Gill, Sh. Even, A. A. Kurmit, Z. D. Dai, D. F. Ye, K. Y. Lam, R. Tao from [1 – 8] are $(\forall q\forall\alpha\forall\delta, \varnothing)$ and $(\forall q\forall\alpha\forall\delta, \{q\})$ respectively.

There are many scientific problems which are related to the cryptanalytical notion of the automaton invertibility with finite delay and are thoroughly enumerated in the previous author's paper [9]. Some of them were particularly solved in [10]. Here we study the decision problem of finding out whether a given automaton is invertible of a given type with a given delay.

The main result of this study is presented by the logical expressions from quantifier logic that are constructively describe necessary and sufficient conditions for an automaton $A$ to have a recovery function $f$ of a certain type and, consequently, to be invertible with a finite delay and of a given type.

Further, in sections 3 and 4, as a short review, we tell some own results related to automaton and function cryptanalytical invertibility conditions.

## 2. Finite automata and normal logical formulas

A finite automaton is described as $A = (X, Q, Y, \psi, \varphi)$, where $X$, $Q$ and $Y$ are its input alphabet, state set and output alphabet respectively, $\psi$ and $\varphi$ — its transfer and output functions, $\psi : X \times Q \to Q$ and $\varphi : X \times Q \to Y$. The last ones being defined for pairs $xq \in X \times Q$, are extended to pairs $\alpha q \in X^* \times Q$ by induction on the length $|\alpha|$ of the word $\alpha \in X^*$, namely the functions $\psi : X^* \times Q \to Q$ and $\bar{\varphi} : X^* \times Q \to Y^*$ are defined as $\psi(\Lambda, q) = q$, $\psi(\alpha\beta, q) = \psi(\beta, \psi(\alpha, q))$, $\bar{\varphi}(\Lambda, q) = \Lambda$, $\bar{\varphi}(x, q) = \varphi(x, q)$ and $\bar{\varphi}(\alpha\beta, q) = \bar{\varphi}(\alpha, q)\bar{\varphi}(\beta, \psi(\alpha, q))$. Here the symbol $\Lambda$ denotes the empty word in any alphabet. Thus, $\psi(\alpha, q)$ is a state, into which the automaton $A$ comes from a state $q$ under acting input word $\alpha$, and $\bar{\varphi}(\alpha, q)$ is an output word which the automaton produces this time.

Everywhere further $\tau$ is a non-negative integer called a delay and it is supposed that in logical formulas $a \in X$, $b \in X$, $\alpha \in X^*$, $\beta \in X^*$, $\delta \in X^\tau$, $\varepsilon \in X^\tau$, $q \in Q$, $s \in Q$.

## 3. Decision Problem for finite Automaton Cryptanalytical Invertibility

Consider a finite automaton $A = (X, Q, Y, \psi, \varphi)$. Let $q, \alpha, \delta$ be variables with values in $Q, X^*, X^\tau$, denoting respectively initial state, prefix and suffix of input word $\alpha\delta$ of the

automaton $A$, and $K = \{\forall q, \forall\alpha, \forall\delta, \exists q, \exists\delta\}$ be the set of universal and existential quantifiers bounding these variables. In reality, the quantifiers in $K$ are $\forall q \in Q$, $\forall\alpha \in X^*$, $\forall\delta \in X^\tau$, $\exists q \in Q$, $\exists\delta \in X^\tau$. For brevity, we write them without ranges of its variables. Note, that $K$ doesn't contain the quantifier $\exists\alpha$. Also, let $V$ be the set of all subsets $\upsilon(q, \alpha, \delta)$ of the set $V_0 = \{q, \delta, \psi(\alpha, q), \psi(\alpha\delta, q)\}$ with the initial, intermediate, and final states $q, \psi(\alpha, q)$, and $\psi(\alpha\delta, q)$ respectively and a delay word $\delta$.

We say, that the automaton $A$ is *cryptanalytically invertible* (with a *delay* IDel $\tau = |\delta|$, of a *type* IT $= (K_1x_1, K_2x_2, K_3x_3)$ with $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, of a *degree* IDeg $= (K_1, K_2, K_3)$, $K_i \in \{\forall, \exists\}$, $K_ix_i \in K$, $i = 1, 2, 3$, and of *an order* IO $= \upsilon(q, \alpha, \delta) \in V$) if there exists an *inverse* (*recovering*) *function* IF $f : Y^* \times V \to X^*$ *with the invertibility property* expressed in the form

$$K_1x_1K_2x_2K_3x_3(f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) = \alpha), \tag{1}$$

in this case the used delay IDel, type IT, degree IDeg, order IO and function IF we call the parameters of this invertibility.

In reality, 208 different kinds of cryptanalytical invertibility of finite automata are now defined. All of them are presented in the Table 1 [9].

Table 1

**Formulas for cryptanalytical invertibility conditions of a finite automaton**

| No | Quantifier prefix | No | Underlying expression |
|---|---|---|---|
| | | 1 | $f(\bar{\varphi}(\alpha\delta, q)) = \alpha$ |
| | | 2 | $f(\bar{\varphi}(\alpha\delta, q), q) = \alpha$ |
| | | 3 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha, q)) = \alpha$ |
| 1 | $\exists f\,\forall q\,\forall\alpha\,\forall\delta$ | 4 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha\delta, q)) = \alpha$ |
| 2 | $\exists f\,\forall q\,\forall\alpha\,\exists\delta$ | 5 | $f(\bar{\varphi}(\alpha\delta, q), \delta) = \alpha$ |
| 3 | $\exists f\,\forall q\,\exists\delta\,\forall\alpha$ | 6 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha, q)) = \alpha$ |
| 4 | $\exists f\,\exists q\,\forall\alpha\,\forall\delta$ | 7 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha\delta, q)) = \alpha$ |
| 5 | $\exists f\,\exists q\,\forall\alpha\,\exists\delta$ | 8 | $f(\bar{\varphi}(\alpha\delta, q), q, \delta) = \alpha$ |
| 6 | $\exists f\,\exists q\,\exists\delta\,\forall\alpha$ | 9 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha, q), \psi(\alpha\delta, q)) = \alpha$ |
| 7 | $\exists f\,\forall\alpha\,\exists q\,\forall\delta$ | 10 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha, q), \delta) = \alpha$ |
| 8 | $\exists f\,\forall\alpha\,\exists q\,\exists\delta$ | 11 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha\delta, q), \delta) = \alpha$ |
| 9 | $\exists f\,\forall\alpha\,\forall\delta\,\exists q$ | 12 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha, q), \psi(\alpha\delta, q)) = \alpha$ |
| 10 | $\exists f\,\forall\alpha\,\exists\delta\,\forall q$ | 13 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha, q), \delta) = \alpha$ |
| 11 | $\exists f\,\forall\delta\,\exists q\,\forall\alpha$ | 14 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha\delta, q), \delta) = \alpha$ |
| 12 | $\exists f\,\exists\delta\,\forall q\,\forall\alpha$ | 15 | $f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha, q), \psi(\alpha\delta, q), \delta) = \alpha$ |
| 13 | $\exists f\,\exists\delta\,\forall\alpha\,\exists q$ | 16 | $f(\bar{\varphi}(\alpha\delta, q), q, \psi(\alpha, q), \psi(\alpha\delta, q), \delta) = \alpha$ |

The definition of the cryptanalytical invertibility of a finite automaton given above unfortunately is not constructive one. It doesn't contain any necessary and sufficient conditions for an automaton to be cryptanalytically invertible with the given parameters, but implies the existence of a recovering function, and should be provided with an algorithmic test in order to effectively find out whether there exist the function $f$ such that the formula (1) is true and in case of a positive answer to produce an effective algorithm for constructing such a function, to become convinced in the automaton cryptanalytical invertibility with the nedeed properties and possibly with the existence of the inverse automaton and to design the last.

So, the following is the first Decision Problem which we need to put and try to solve for further developing the theory of Cryptanalytically Invertible finite automata — ACIDP: given a finite automaton $A$, the values of invertibility delay IDel $\tau$, an invertibility type IT,

an invertibility degree IDeg and (or) an invertibility order IO for cryptanalytical invertibility of the automaton $A$, find out whether the automaton $A$ is invertible of type IT with the delay $\tau$ and of order IO and if so, to construct a proper recovering function $f$ satisfying the condition (1).

The important place in this row takes the problem of creation or generating invertible automata of all possible types. In various decisions of this problem different requirements to generated automata can be presented: with equal probability in given invertibility class, with bounded complexity, with the great or, otherwise, small delay of invertibility and so on. Its solution seems to be impossible without proper decision of the first problem.

Further, in this Section, we tell some results related to ACIDP, published in [9, 10] and given here in the form of Propositions.

**Proposition 1** [9]. The automaton $A$ is cryptanalytically invertible with the delay $\tau = |\delta|$, of type $(\forall q, \forall\alpha, \forall\delta)$, and of an order $\upsilon(q, \alpha, \delta)$, that is, there exists a function $f : Y^* \times V \to X^*$ with the invertibility property

$$\forall q \forall\alpha \forall\delta (f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) = \alpha),$$

if and only if

$$\forall q \forall\alpha \forall\delta \forall s \forall\beta \forall\varepsilon (\alpha \neq \beta \Rightarrow (\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) \neq (\bar{\varphi}(\beta\varepsilon, s), \upsilon(s, \beta, \varepsilon)).$$

**Proposition 2** [9]. If the automaton $A$ is invertible of a delay $|\delta|$, of a type $K_1x_1K_2x_2K_3x_3$, and of an order $\upsilon(q, \alpha, \delta)$, that is, if there exists a function $f : Y^* \times V \to X^*$ with the invertibility property (1), then

$$K_1x_1K_2x_2K_3x_3K_1y_1K_2y_2K_3y_3\big(\alpha \neq \beta \Rightarrow (\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) \neq (\bar{\varphi}(\beta\varepsilon, s), \upsilon(s, \beta, \varepsilon))\big),$$

where $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}, \{y_1, y_2, y_3\} = \{s, \beta, \varepsilon\}$.

**Definition 1** [10]. Having a quantifier sequence $K_1x_1, \ldots, K_nx_n$ (a quantifier prefix of a predicate logical formula in a normal form), we believe $n = m + s$, $i_1 \leqslant \ldots \leqslant i_m$, $j_1 \leqslant \ldots \leqslant j_s$, $\{i_1, \ldots i_m, j_1, \ldots, j_s\} = \{1, \ldots, n\}$, $K_{i_1} = \ldots = K_{i_m} = \forall$, $K_{j_1} = \ldots = K_{j_s} = \exists$. Also, for $r \in \{j_1, \ldots, j_s\}$ let $\varepsilon_r : D_1 \times \ldots \times D_{r-1} \to D_r$ be a function (called existential) the value $\varepsilon_r(a_1, \ldots, a_{r-1})$ of which is computed by the quantifier $K_rx_r$ with $K_r = \exists$ as a next value $a_r$ of the variable $x_r$ in dependence on the last values $a_1, \ldots, a_{r-1}$ of variables $x_1, \ldots, x_{r-1}$, predecessors to variable $x_r$. Finally, define the vector existential function $\varepsilon = \varepsilon_{j_1} \ldots \varepsilon_{j_s}$ and the set $M_\varepsilon$ called the existential domain of the quantifier sequence $K_1x_1 \ldots K_nx_n$ corresponding to existential functions in $\varepsilon$. By the definition

$$a_1 \ldots a_n \in M_\varepsilon \Leftrightarrow (a_1 \ldots a_n \in D_1 \times \ldots \times D_n) \& (a_{j_1} \ldots a_{j_s} = \varepsilon_{j_1}(a_1 \ldots a_{j_1-1}) \ldots \varepsilon_{j_s}(a_1 \ldots a_{j_s-1}).$$

**Proposition 3** [10]. The automaton $A$ is cryptanalytically invertible of a delay $|\delta|$, of a type $K_1x_1K_2x_2K_3x_3$, where $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, and of an order $\upsilon(q, \alpha, \delta)$, that is, there exists a function $f : Y^* \times V \to X^*$ with the invertibility property, if and only if for $K_1x_1K_2x_2K_3x_3$ there is an existential vector function $\varepsilon$ such that

$$\forall a_1a_2a_3 \in M_\varepsilon \forall b_1b_2b_3 \in M_\varepsilon (\alpha \neq \beta \Rightarrow ((\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) \neq (\bar{\varphi}(\beta\varepsilon, s), \upsilon(s, \beta, \gamma)))),$$

where $q$ and $s \in Q, \alpha$ and $\beta \in X^*, \delta$ and $\gamma \in X^\tau, a_1a_2a_3$ and $b_1b_2b_3 \in M_\varepsilon$, if $a_k = q, \alpha$ or $\delta$, then $b_k = s, \beta$ or $\gamma$ respectively, $k \in \{1, 2, 3\}$.

## 4. Decision Problem for Function Cryptanalytical Invertibility

This problem we use as an auxiliary one to the main problem — ACIDP.

Let $g(x_1, \ldots, x_n)$ be a function in variables $x_1, \ldots, x_n$ with a range $D_g$, $K_i x_i$ be a quantifier bounding the variable $x_i$, $i = 1, \ldots n$, $k_0 \in \{1, \ldots, n\}$, and $K_{k_0} = \forall$. Let $f : D_g \to D_{k_0}$ denote an arbitrary function with the domain $D_g$ and the range $D_{k_0}$ and, for definition correctness of $f$, let $|D_g| \geqslant |D_{k_0}|$. Moreover, always further we believe that each function under consideration has a domain with the number of elements not less than the number of elements in its range.

We say, that the function $g(x_1, x_2, \ldots, x_n)$ is *cryptanalytically invertible* (with respect to an invertibility variable IV $= x_{k_0}$ and of the type IT $= K_1 x_1 \ldots K_n x_n$) if there exists an *inverse* (*recovering*) *function* $f : D_g \to D_{k_0}$ with the *invertibility property* expressed in the form

$$K_1 x_1 \ldots K_n x_n (f(g(x_1, \ldots, x_n)) = x_{k_0}), \tag{2}$$

in this case the used number $k_0$, type IT, and the function $f$ we call the parameters of this invertibility.

The following is the Decision Problem for Function Cryptanalytical Invertibility — FCIDP: given a function $g$, a variable $x_{k_0}$ and invertibility type IT $= K_1 x_1 \ldots K_n x_n$ for cryptanalytical invertibility of the function $g$, find out whether the function $g$ is invertible with respect to IV $x_{k_0}$ and of type IT and if so, to construct a function $f$ satisfying the condition (2).

We should see that our main problem ACIDP is obtained from the particular case of our auxiliary problem FCIDP by taking $n = 3$, $k_0 \in \{1, 2, 3\}$, $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, $x_{k_0} = \alpha$, $g(x_1, x_2, x_3) = (\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta))$. Thus, a method deciding the auxiliary problem also decides the main one and so, our problem ACIDP reduces to our problem FCIDP.

Now we tell some theoretical results related to FCIDP published in [9, 10].

**Proposition 4** [9]. For any function $g : D_1 \times \ldots \times D_n \to D_g$ a function $f : D_g \to D_{k_0}$ with the invertibility property

$$\forall x_1 \ldots \forall x_n (f(g(x_1, \ldots, x_n)) = x_{k_0})$$

exists if and only if

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n (x_{k_0} \neq y_{k_0} \Rightarrow g(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_n)).$$

**Proposition 5** [9]. For any function $g$, if there exists a function $f : D_g \to D_{k_0}$ with the invertibility property (2), then

$$K_1 x_1 \ldots K_n x_n K_1 y_1 \ldots K_n y_n (x_{k_0} \neq y_{k_0} \Rightarrow g(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_n)).$$

**Proposition 6** [10]. For any function $g(x_1, \ldots, x_n)$ there exists a function $f : D_g \to$ $\to D_{k_0}$ with the invertibility property (2), if and only if for each $k \in \{j_1, \ldots, j_s\}$ there exists an existential function $\varepsilon_k : D_1 \times \ldots \times D_{k-1} \to D_k$ such that the set $M_\varepsilon$ corresponding to $\varepsilon = \varepsilon_{j_1} \ldots \varepsilon_{j_s}$ satisfies the following condition:

$$\forall a = a_1 \ldots a_n \in M_\varepsilon \forall b = b_1 \ldots b_n \in M_\varepsilon (a_{k_0} \neq b_{k_0} \Rightarrow g(a) \neq g(b)).$$

## 5. Formulas for cryptanalytical invertibility conditions for finite automata

The Table 1 from [9] is presented here for constructing invertibility condition of an automaton from its parts. The line number $i \in \{1, 2, \ldots, 13\}$ specifies the quantifier prefix $\exists f K_1 x_1 K_2 x_2 K_3 x_3$ and the line number $j \in \{1, 2, \ldots, 16\}$ describes the underlying expression $f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) = \alpha$, where $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$. These parts form the complete condition denoted $U_{i,j}$ or $U_{i,j}[\tau]$, if a reference to the invertibility delay is required. For instance, $U_{1,1}[\tau] = \exists f \forall q \forall \alpha \forall \delta (f(\bar{\varphi}(\alpha\delta, q)) = \alpha)$, $U_{1,2}[\tau] = \exists f \forall q \forall \alpha \forall \delta (f(\bar{\varphi}(\alpha\delta, q), q) = \alpha)$, $U_{5,10}[\tau] = \exists f \exists q \forall \alpha \exists \delta (f(\bar{\varphi}(\alpha\delta, q), \psi(\alpha, q), \delta) = \alpha)$.

Let $\kappa^{(i)}(q, \alpha, \delta)$ and $\upsilon_j(q, \alpha, \delta)$ be respectively $K_1 x_1 K_2 x_2 K_3 x_3$ where $\exists f K_1 x_1 K_2 x_2 K_3 x_3$ is the quantifier prefix from the line number $i$ in the Table 1 and the invertibility order $\upsilon_j(q, \alpha, \delta)$ from the line number $j$ in underlying expression $f(\bar{\varphi}(\alpha\delta, q), \upsilon_j(q, \alpha, \delta)) = \alpha$ in the Table 1. For example, $\kappa^{(5)}(q, \alpha, \delta) = \exists q \forall \alpha \exists \delta$ and $\upsilon_{13}(q, \alpha, \delta) = \{q, \psi(\alpha, q), \delta\}$.

So for any $i \in \{1, 2, \ldots, 13\}$ and $j \in \{1, 2, \ldots, 16\}$, $\kappa^{(i)}(q, \alpha, \delta)$ and $\upsilon_j(q, \alpha, \delta)$ present some cryptanalytical parameters of finite automata such as their invertibility type, degree, order and so on.

Let also $\kappa_i(q, \alpha, \delta, s, \beta, \varepsilon) = \kappa^{(i)}(q, \alpha, \delta)\kappa^{(i)}(s, \beta, \varepsilon)$, $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, $\{y_1, y_2, y_3\} = \{s, \beta, \varepsilon\}$, $x_{k_0} = \alpha$, $y_{k_0} = \beta$, $g(x_1, \ldots, x_n) = (\bar{\varphi}(\alpha\delta, q), \upsilon_j(q, \alpha, \delta))$,

$$K_1 x_1 K_2 x_2 K_3 x_3 K_1 y_1 K_2 y_2 K_3 y_3 = \kappa_i(q, \alpha, \delta, s, \beta, \varepsilon),$$

and we can write down the invertibility condition (1) in the following new form

$$U_{i,j} = \exists f \kappa^{(i)}(q, \alpha, \delta)[f(\bar{\varphi}(\alpha\delta, q), \upsilon_j(q, \alpha, \delta)) = \alpha],$$

its equivalent from Proposition 2 in the following new form

$$\kappa_i(q, \alpha, \delta, s, \beta, \varepsilon)[(\alpha \neq \beta \Rightarrow (\bar{\varphi}(\alpha\delta, q), \upsilon_j(q, \alpha, \delta)) \neq (\bar{\varphi}(\beta\varepsilon, s), \upsilon_j(s, \beta, \varepsilon)))].$$

where $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}, \{y_1, y_2, y_3\} = \{s, \beta, \varepsilon\}$.

Thus, the following assertion is proved.

**Proposition 7.** For all $i \in \{1, 2, \ldots, 13\}$ and $j \in \{1, 2, \ldots, 16\}$,

$$U_{i,j} = \kappa_i(q, \alpha, \delta, s, \beta, \varepsilon)[\alpha \neq \beta \ \& \ \upsilon_j(q, \alpha, \delta) = \upsilon_j(s, \beta, \varepsilon) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]. \quad (3)$$

For example,

$$U_{2,7} = \forall q \forall \alpha \exists \delta \forall s \forall \beta \exists \varepsilon [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)].$$

For compact presentation of obtained so, including simplifications, logic formulas for all 208 invertibility conditions $U_{i,j}$ of automata $A$, below at first separately the notation in them is enumerated for all possible quantifier prefixes and underlying expressions, and then in the Table 2 the formulas themselfes in the form $\kappa\Gamma$, where $\kappa$ and $\Gamma$ — notation respectively of quantifier prefix and of underlying expression from the given list.

In some cases formula (3) for $U_{i,j}$ can be simplified, namely: if the condition $\upsilon_j(q, \alpha, \delta) = \upsilon_j(s, \beta, \varepsilon)$ includes the equalities $q = s$ and (or) $\delta = \varepsilon$, and the quantifier prefix $\kappa^{(i)}(q, \alpha, \delta)\kappa^{(i)}(s, \beta, \varepsilon)$ contains quantifiers $\forall q$, $\forall s$ and (or) $\forall\delta$, $\forall\varepsilon$, then it is possible to exclude simultaneously from this condition these equalities, to exclude from prefix quantifiers $\forall s$ and (or) $\forall\varepsilon$, and to put in conclusion $(\bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s))$ $s = q$ and (or) $\varepsilon = \delta$ respectively. This simplification is an equivalent transformation and doesn't change the truth value of the formula.

Notations of quantifier prefixes $\kappa$ in the formulas of automaton $A$ invertibility conditions:

$$
\begin{array}{lll}
a = \forall q \forall \alpha \forall \delta \forall s \forall \beta \forall \varepsilon; & i = \forall \alpha \forall \delta \exists q \forall \beta \forall \varepsilon \exists s; & b_2 = \forall q \forall \alpha \exists \delta \forall \beta \exists \varepsilon; \\
b = \forall q \forall \alpha \exists \delta \forall s \forall \beta \exists \varepsilon; & j = \forall \alpha \exists \delta \forall q \forall \beta \exists \varepsilon \forall s; & c_2 = \forall q \exists \delta \forall \alpha \exists \varepsilon \forall \beta; \\
c = \forall q \exists \delta \forall \alpha \forall s \exists \varepsilon \forall \beta; & k = \forall \delta \exists q \forall \alpha \forall \varepsilon \exists s \forall \beta; & d_1 = \exists q \forall \alpha \forall \delta \exists s \forall \beta; \\
d = \exists q \forall \alpha \forall \delta \exists s \forall \beta \forall \varepsilon; & l = \exists \delta \forall q \forall \alpha \exists \varepsilon \forall s \forall \beta; & g_1 = \forall \alpha \exists q \forall \delta \forall \beta \exists s; \\
e = \exists q \forall \alpha \exists \delta \exists s \forall \beta \exists \varepsilon; & m = \exists \delta \forall \alpha \exists q \exists \varepsilon \forall \beta \exists s; & i_1 = \forall \alpha \forall \delta \exists q \forall \beta \exists s; \\
f = \exists q \exists \delta \forall \alpha \exists s \exists \varepsilon \forall \beta; & a_2 = \forall q \forall \alpha \forall \delta \forall \beta \forall \varepsilon; & j_2 = \forall \alpha \exists \delta \forall q \forall \beta \exists \varepsilon; \\
g = \forall \alpha \exists q \forall \delta \forall \beta \exists s \forall \varepsilon; & a_1 = \forall q \forall \alpha \forall \delta \forall s \forall \beta & k_1 = \forall \delta \exists q \forall \alpha \exists s \forall \beta; \\
h = \forall \alpha \exists q \exists \delta \forall \beta \exists s \exists \varepsilon; & a_0 = \forall q \forall \alpha \forall \delta \forall \beta; & l_2 = \exists \delta \forall q \forall \alpha \exists \varepsilon \forall \beta.
\end{array}
$$

Notation of underlying expressions $\Gamma$ in formulas of automaton $A$ invertibility conditions:

$$
\begin{aligned}
A &= [\alpha \neq \beta \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
B &= [\alpha \neq \beta \ \& \ q = s \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
B_2 &= [\alpha \neq \beta \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
C &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
D &= [\alpha \neq \beta \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
E &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
E_1 &= [\alpha \neq \beta \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
F &= [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
F_2 &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
G &= [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
G_2 &= [\alpha \neq \beta \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
H &= [\alpha \neq \beta \ \& \ q = s \ \& \ \delta = \varepsilon \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
H_2 &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
H_1 &= [\alpha \neq \beta \ \& \ q = s \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
H_0 &= [\alpha \neq \beta \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, q)]; \\
I &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
J &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
J_1 &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
K &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
K_1 &= [\alpha \neq \beta \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
L &= [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
L_2 &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, q) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
M &= [\alpha \neq \beta \ \& \ q = s \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
M_2 &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
M_1 &= [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha, q) = \psi(\beta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
M_0 &= [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, q)]; \\
N &= [\alpha \neq \beta \ \& \ q = s \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]; \\
N_2 &= [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]; \\
N_1 &= [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]; \\
N_0 &= [\alpha \neq \beta \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, q)];
\end{aligned}
$$

$$O = [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)];$$
$$O_1 = [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)];$$
$$P = [\alpha \neq \beta \ \& \ q = s \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, s) \Rightarrow$$
$$\Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)];$$
$$P_2 = [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, q) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\varepsilon, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)];$$
$$P_1 = [\alpha \neq \beta \ \& \ q = s \ \& \ \psi(\alpha, q) = \psi(\beta, s) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, s) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)];$$
$$P_0 = [\alpha \neq \beta \ \& \ \psi(\alpha, q) = \psi(\beta, q) \ \& \ \psi(\alpha\delta, q) = \psi(\beta\delta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, q)].$$

Formulas $\kappa\Gamma$ for conditions of the automaton $A$ invertibility of all possible types are presented in the Table 2. For any words $\xi, \zeta$ in $X^*$ and states $q, s$ in $Q$ inequality $\bar{\varphi}(\xi, q) \neq \neq \bar{\varphi}(\zeta, s)$ is always provided when $|\xi| \neq |\zeta|$, therefore in any of these formulas it is possible to count $|\alpha| = |\beta|$. Further we use this possibility without additional reservations.

Table 2

**Formulas $\kappa\Gamma$ for conditions of the automaton invertibility of all possible types**

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $U_{i,1}$ | $aA$ | $bA$ | $cA$ | $dA$ | $eA$ | $fA$ | $gA$ | $hA$ | $iA$ | $jA$ | $kA$ | $lA$ | $mA$ |
| $U_{i,2}$ | $a_2B_2$ | $b_2B_2$ | $c_2B_2$ | $dB$ | $eB$ | $fB$ | $gB$ | $hB$ | $iB$ | $j_2B_2$ | $kB$ | $l_2B_2$ | $mB$ |
| $U_{i,3}$ | $aC$ | $bC$ | $cC$ | $dC$ | $eC$ | $fC$ | $gC$ | $hC$ | $iC$ | $jC$ | $kC$ | $lC$ | $mC$ |
| $U_{i,4}$ | $aD$ | $bD$ | $cD$ | $dD$ | $eD$ | $fD$ | $gD$ | $hD$ | $iD$ | $jD$ | $kD$ | $lD$ | $mD$ |
| $U_{i,5}$ | $a_1E_1$ | $bE$ | $cE$ | $d_1E_1$ | $eE$ | $fE$ | $g_1E_1$ | $hE$ | $i_1E_1$ | $jE$ | $k_1E_1$ | $lE$ | $mE$ |
| $U_{i,6}$ | $a_2F_2$ | $b_2F_2$ | $c_2F_2$ | $dF$ | $eF$ | $fF$ | $gF$ | $hF$ | $iF$ | $j_2F_2$ | $kF$ | $l_2F_2$ | $mF$ |
| $U_{i,7}$ | $a_2G_2$ | $b_2G_2$ | $c_2G_2$ | $dG$ | $eG$ | $fG$ | $gG$ | $hG$ | $iG$ | $j_2G_2$ | $kG$ | $l_2G_2$ | $mG$ |
| $U_{i,8}$ | $a_0H_0$ | $b_2H_2$ | $c_2H_2$ | $d_1H_1$ | $eH$ | $fH$ | $g_1H_1$ | $hH$ | $i_1H_1$ | $j_2H_2$ | $k_1H_1$ | $l_2H_2$ | $mH$ |
| $U_{i,9}$ | $aI$ | $bI$ | $cI$ | $dI$ | $eI$ | $fI$ | $gI$ | $hI$ | $iI$ | $jI$ | $kI$ | $lI$ | $mI$ |
| $U_{i,10}$ | $a_1J_1$ | $bJ$ | $cJ$ | $d_1J_1$ | $eJ$ | $fJ$ | $g_1J_1$ | $hJ$ | $i_1J_1$ | $jJ$ | $k_1J_1$ | $lJ$ | $mJ$ |
| $U_{i,11}$ | $a_1K_1$ | $bK$ | $cK$ | $d_1K_1$ | $eK$ | $fK$ | $g_1K_1$ | $hK$ | $i_1K_1$ | $jK$ | $k_1K_1$ | $lK$ | $mK$ |
| $U_{i,12}$ | $a_2L_2$ | $b_2L_2$ | $c_2L_2$ | $dL$ | $eL$ | $fL$ | $gL$ | $hL$ | $iL$ | $j_2L_2$ | $kL$ | $l_2L_2$ | $mL$ |
| $U_{i,13}$ | $a_0M_0$ | $b_2M_2$ | $c_2M_2$ | $d_1M_1$ | $eM$ | $fM$ | $g_1M_1$ | $hM$ | $i_1M_1$ | $j_2M_2$ | $k_1M_1$ | $l_2M_2$ | $mM$ |
| $U_{i,14}$ | $a_0N_0$ | $b_2N_2$ | $c_2N_2$ | $d_1N_1$ | $eN$ | $fN$ | $g_1N_1$ | $hN$ | $i_1N_1$ | $j_2N_2$ | $k_1N_1$ | $l_2N_2$ | $mN$ |
| $U_{i,15}$ | $a_1O_1$ | $bO$ | $cO$ | $d_1O_1$ | $eO$ | $fO$ | $g_1O_1$ | $hO$ | $i_1O_1$ | $jO$ | $k_1O_1$ | $lO$ | $mO$ |
| $U_{i,16}$ | $a_0P_0$ | $b_2P_2$ | $c_2P_2$ | $d_1P_1$ | $eP$ | $fP$ | $g_1P_1$ | $hP$ | $i_1P_1$ | $j_2P_2$ | $k_1P_1$ | $l_2P_2$ | $mP$ |

Let in the Table 2 for any $i = 1, 2, \ldots, 13$ and $j = 1, 2, \ldots, 16$ element on intersection of line with $U_{i,j}$ at the head and column with the number $i$ is denoted $T_{ij}$. Then $U_{i,j} = T_{ij}$.

So, for instance, $U_{1,1} = aA = \forall q \forall \alpha \forall \delta \forall s \forall \beta \forall \varepsilon [\alpha \neq \beta \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)]$; $U_{4,8} = d_1H_1 = \exists q \forall \alpha \forall \delta \exists s \forall \beta [\alpha \neq \beta \ \& \ q = s \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\delta, s)]$; $U_{10,13} = j_2M_2 = = \forall \alpha \exists \delta \forall q \forall \beta \exists \varepsilon [\alpha \neq \beta \ \& \ \delta = \varepsilon \ \& \ \psi(\alpha, q) = \psi(\beta, q) \Rightarrow \bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, q)]$, and so on.

If in expression for $U_{i,j}$ in (3) the condition $\upsilon_j(q, \alpha, \delta) = \upsilon_j(s, \beta, \varepsilon)$ includes into itself equalities $q = s$ and (or) $\delta = \varepsilon$, and quantifier prefix $\kappa^{(i)}(q, \alpha, \delta)\kappa^{(i)}(s, \beta, \varepsilon)$ contains quantifiers $\exists q, \exists s$ and (or) $\exists \delta, \exists \varepsilon$, then, putting down from it these equalities and quantifiers $\exists s$ and (or) $\exists \varepsilon$ and placing in its conclusion $\bar{\varphi}(\alpha\delta, q) \neq \bar{\varphi}(\beta\varepsilon, s)$ respectively $s = q$ and (or) $\varepsilon = \delta$, we obtain expression, denoted further $U'_{i,j}$, for which the implication $U'_{i,j} \Rightarrow U_{i,j}$ is true. This means that the condition $U'_{i,j}$ is sufficient, but not obligatory necessary for invertibility of type $\kappa^{(i)}$ and order $\upsilon_j$ for automaton $A$. All obtained so sufficient conditions of invertibility for automaton $A$ are represented in Table 3. In it quantifier prefixes have the following notation:

$b_1 = \forall q \forall \alpha \exists \delta \forall s \forall \beta$; $e_1 = \exists q \forall \alpha \exists \delta \exists s \forall \beta$; $h_2 = \forall \alpha \exists q \exists \delta \forall \beta \exists \varepsilon$; $k_2 = \forall \delta \exists q \forall \alpha \forall \varepsilon \forall \beta$;
$b_0 = \forall q \forall \alpha \exists \delta \forall \beta$; $e_0 = \exists q \forall \alpha \exists \delta \forall \beta$; $h_1 = \forall \alpha \exists q \exists \delta \forall \beta \exists s$; $k_0 = \forall \delta \exists q \forall \alpha \forall \beta$;

$c_1 = \forall q \exists \delta \forall \alpha \forall s \forall \beta$; $f_2 = \exists q \exists \delta \forall \alpha \exists \varepsilon \forall \beta$; $h_0 = \forall \alpha \exists q \exists \delta \forall \beta$; $l_1 = \exists \delta \forall q \forall \alpha \forall s \forall \beta$;
$c_0 = \forall q \exists \delta \forall \alpha \forall \beta$; $f_1 = \exists q \exists \delta \forall \alpha \exists s \forall \beta$; $i_2 = \forall \alpha \forall \delta \exists q \forall \beta \forall \varepsilon$; $l_0 = \exists \delta \forall q \forall \alpha \forall \beta$;
$d_2 = \exists q \forall \alpha \forall \delta \forall \beta \forall \varepsilon$; $f_0 = \exists q \exists \delta \forall \alpha \forall \beta$; $i_0 = \forall \alpha \forall \delta \exists q \forall \beta$; $m_2 = \exists \delta \forall \alpha \exists q \exists \varepsilon \forall \beta$;
$d_0 = \exists q \forall \alpha \forall \delta \forall \beta$; $g_2 = \forall \alpha \exists q \forall \delta \forall \beta \forall \varepsilon$; $j_1 = \forall \alpha \exists \delta \forall q \forall \beta \forall s$; $m_1 = \exists \delta \forall \alpha \exists q \forall \beta \exists s$;
$e_2 = \exists q \forall \alpha \exists \delta \forall \beta \exists \varepsilon$; $g_0 = \forall \alpha \exists q \forall \delta \forall \beta$; $j_0 = \forall \alpha \exists \delta \forall q \forall \beta$; $m_0 = \exists \delta \forall \alpha \exists q \forall \beta$.

Table 3

**Formulas for sufficient conditions of the automaton invertibility of some types**

| $i$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $U'_{i,2}$ | | | $d_2B_2$ | $e_2B_2$ | $f_2B_2$ | $g_2B_2$ | $h_2B_2$ | $i_2B_2$ | | $k_2B_2$ | | $m_2B_2$ |
| $U'_{i,5}$ | $b_1E_1$ | $c_1E_1$ | | $e_1E_1$ | $f_1E_1$ | | $h_1E_1$ | | $j_1E_1$ | | $l_1E_1$ | $m_1E_1$ |
| $U'_{i,6}$ | | | $d_2F_2$ | $e_2F_2$ | $f_2F_2$ | $g_2F_2$ | $h_2F_2$ | $i_2F_2$ | | $k_2F_2$ | | $m_2F_2$ |
| $U'_{i,7}$ | | | $d_2G_2$ | $e_2G_2$ | $f_2G_2$ | $g_2G_2$ | $h_2G_2$ | $i_2G_2$ | | $k_2G_2$ | | $m_2G_2$ |
| $U'_{i,8}$ | $b_0H_0$ | $c_0H_0$ | $d_0H_0$ | $e_0H_0$ | $f_0H_0$ | $g_0H_0$ | $h_0H_0$ | $i_0H_0$ | $j_0H_0$ | $k_0H_0$ | $l_0H_0$ | $m_0H_0$ |
| $U'_{i,10}$ | $b_1J_1$ | $c_1J_1$ | | $e_1J_1$ | $f_1J_1$ | | $h_1J_1$ | | $j_1J_1$ | | $l_1J_1$ | $m_1J_1$ |
| $U'_{i,11}$ | $b_1K_1$ | $c_1K_1$ | | $e_1K_1$ | $f_1K_1$ | | $h_1K_1$ | | $j_1K_1$ | | $l_1K_1$ | $m_1K_1$ |
| $U'_{i,12}$ | | | $d_2L_2$ | $e_2L_2$ | $f_2L_2$ | $g_2L_2$ | $h_2L_2$ | $i_2L_2$ | | $k_2L_2$ | | $m_2L_2$ |
| $U'_{i,13}$ | $b_0M_0$ | $c_0M_0$ | $d_0M_0$ | $e_0M_0$ | $f_0M_0$ | $g_0M_0$ | $h_0M_0$ | $i_0M_0$ | $j_0M_0$ | $k_0M_0$ | $l_0M_0$ | $m_0M_0$ |
| $U'_{i,14}$ | $b_0N_0$ | $c_0N_0$ | $d_0N_0$ | $e_0N_0$ | $f_NH_0$ | $g_0N_0$ | $h_0N_0$ | $i_0N_0$ | $j_0N_0$ | $k_0N_0$ | $l_0N_0$ | $m_0N_0$ |
| $U'_{i,15}$ | $b_1O_1$ | $c_1O_1$ | | $e_1O_1$ | $f_1O_1$ | | $h_1O_1$ | | $j_1O_1$ | | $l_1O_1$ | $m_1O_1$ |
| $U'_{i,16}$ | $b_0P_0$ | $c_0P_0$ | $d_0P_0$ | $e_0P_0$ | $f_0P_0$ | $g_0P_0$ | $h_0P_0$ | $i_0P_0$ | $j_0P_0$ | $k_0P_0$ | $l_0P_0$ | $m_0P_0$ |

## 6. Problems in the theory of automata cryptanalytical invertibility

**ACIDP**. Given a cryptanalyical invertibility type (quantifier prefix $K_1x_1K_2x_3K_3x_3$, $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, and invertibility order $\upsilon(q, \alpha, \delta)$), as well as an invertibility delay $\tau$, a number $k_0 \in \{1, 2, 3\}$ such that $K_{k_0} = \forall$, $x_{k_0} = \alpha$, a subject direct transformation $g(x_1, x_2, x_3) = (\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta))$, and a subject inverse transformation $f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta))$, find out whether $f$ is a proper recovering function, that is, (1) is true, and hence the automaton under consideration is really invertible of the given type and with the given delay.

**FIDP**. Given a cryptanalytical invertibility type (quantifier prefix $K_1x_1 \dots K_nx_n$), a number $k_0 \in \{1, \dots, n\}$ such that $K_{k_0} = \forall$, and an abstract function $g(x_1, \dots, x_n)$, find out whether there exists a recovering function $f$ such that $f(g(x_1, \dots, x_n)) = x_{k_0}$, and if it exists, then construct it.

For any specific $j$, classes $C_{i,j}(\tau)$ [9] with all possible $i$ and $\tau$ and automata in them we call respectively classes of invertibility and invertible automata *of one propositionality* $(j)$, or *one-propositional* (*of index* $j$). Due to the fact that, for any predicates $P(x)$ and $R(x, y)$, implications $\forall xP(x) \Rightarrow \exists xP(x)$, $\forall x \forall yR(x, y) \Rightarrow \exists x \forall yR(x, y)$ and $\exists x \forall yR(x, y) \Rightarrow$ $\Rightarrow \forall y \exists xR(x, y)$ are true, the inclusion relation is possible between some one-propositional classes of invertibility with equal delay [9].

The invertibility notion of finite automaton considered in this paper doesn't foresee for invertible automaton of obligatory existence of an inverse automaton. Moreover, it is admitted that the function of recovering input prefix for some types of automata invertibility can not be finite automaton. In this situation, naturally, the problem arises to find out for given invertible (of a certain) type automaton whether it has inverse automaton, and if it has, then to construct it. Decision of this problem, in turn, intends the introduction of the definition of an inverse automaton for an arbitrary automaton of every invertibility class.

The important place in this row takes the problem of creation invertible automata of all possible types. In various formulations of this problem different requirements to generated automata can be considered — with equal probability in given invertibility class, with bounded complexity, with a great or, otherwise, small delay of invertibility and so on. Its solution seems to be impossible without proper decision of the ACIDP.

REFERENCES

1. *Huffman D. A.* Canonical forms for information-lossles finite-state logical machines. IRE Trans. Circuit Theory, 1959, vol. 6, spec. suppl., pp. 41–59.
2. *Huffman D. A.* Notes on information-lossles finite-state automata. Nuovo Cimento, 1959, vol. 13, suppl. 2, pp. 397–405.
3. *Gill A.* Introduction to the Theory of Finite-State Machines. N.Y., McGraw-Hill Book Company, 1962. 300 p.
4. *Even Sh.* On information-lossles automata of finite order. IEEE Trans. Electron. Comput., 1965, vol. 14, no. 4, pp. 561–569.
5. *Kurmit A. A.* Information Lossless Automata of Finite Order. N.Y., John Wiley, 1974.
6. *Zakrevskiy A. D.* Metod avtomaticheskoy shifratsii soobshcheniy [The method for messages automatic encryption]. Prikladnaya Diskretnaya Matematika, 2009, no. 2(4), pp. 127–137. (in Russian)
7. *Dai Z. D., Ye D. F., and Lam K. Y.* Weak invertibility of finite automata and cryptanalysis on FAPKC. LNCS, 1998, vol. 1514, pp. 227–241.
8. *Tao R.* Finite Automata and Application to Cryptography. N.Y., Springer, 2009. 406 p.
9. *Agibalov G. P.* Cryptanalytic concept of finite automaton invertibility with finite delay. Prikladhaya Diskrethaya Matematika, 2019, no. 44, pp. 34–42.
10. *Agibalov G. P.* Cryptanalytical finite automaton invertibility with finite delay. Prikladnaya Diskretnaya Matematika, 2019, no. 46, pp. 27–37.

# ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ

УДК 621.391.7

## О НЕКОТОРЫХ СВОЙСТВАХ ПРОИЗВЕДЕНИЯ ШУРА — АДАМАРА ДЛЯ ЛИНЕЙНЫХ КОДОВ И ИХ ПРИЛОЖЕНИЯХ

В. М. Деундяк, Ю. В. Косолапов

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

Произведение Шура — Адамара активно используется при криптоанализе асимметричных кодовых криптосистем типа Мак-Элиса, основанных на линейных кодах. Именно, это произведение успешно применяется при криптоанализе кодовых систем на подкодах обобщённых кодов Рида — Соломона, на двоичных кодах Рида — Маллера и их подкодах коразмерности 1, на соединении некоторых известных кодов. В качестве способа усиления стойкости криптосистемы авторами ранее предложена система на тензорном произведении линейных кодов. С целью анализа стойкости этой системы в настоящей работе исследуются свойства произведения Шура — Адамара для тензорного произведения произвольных линейных кодов. В результате получены необходимые и достаточные условия, когда $s$-я степень тензорного произведения кодов перестановочно эквивалентна прямой сумме кодов. Этот результат позволяет, в частности, выбирать параметры линейных кодов так, чтобы произведение Шура — Адамара для тензорного произведения совпадало со всем пространством, в котором это произведение определено. Таким образом, могут быть определены параметры линейных кодов, при которых атака на основе произведения Шура — Адамара, применённого к публичному ключу, не проходит. Получены некоторые новые свойства произведения Шура — Адамара для линейных кодов, которые позволили, в частности, доказать неразложимость двоичных кодов Рида — Маллера. Как следствие, доказана теорема о структуре группы перестановочных автоморфизмов прямой суммы неразложимых кодов.

**Ключевые слова:** *тензорное произведение, разложимость кодов, криптосистемы типа Мак-Элиса.*

DOI 10.17223/20710410/50/5

## ON SOME PROPERTIES OF THE SCHUR — HADAMARD PRODUCT FOR LINEAR CODES AND THEIR APPLICATIONS

V. M. Deundyak, Yu. V. Kosolapov

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

**E-mail:** vl.deundyak@gmail.com, itaim@mail.ru

The Shur — Hadamard product is actively used in the cryptanalysis of asymmetric code cryptosystems like McEliece based on linear codes. Namely, this product is successfully used in cryptanalysis of code systems on subcodes of generalized Reed — Solomon codes, on binary Reed — Muller codes and their subcodes of codimension 1, on the combination of some well known codes. As a way to enhance the security of

a cryptosystem, the authors have previously proposed a system based on the tensor product of linear codes. In order to analyze the security of this system, in this paper we study the properties of the Schur — Hadamard product for the tensor product of arbitrary linear codes. As a result, necessary and sufficient conditions are obtained when the $s$th power of the tensor product of codes is permutationally equivalent to the direct sum of codes. This result allows, in particular, to choose the parameters of linear codes so that the Schur — Hadamard product for the tensor product coincides with the entire space in which this product is defined. Thus, the parameters of linear codes can be determined, at which the attack based on the Shur — Hadamard product applied to the public key fails. Also, some new results on the Schur — Hadamard product for linear codes were obtained, which made it possible, in particular, to prove the indecomposability of binary Reed — Muller codes. A theorem on the structure of the group of permutation automorphisms of a direct sum of indecomposable codes is proved.

**Keywords:** *tensor product codes, decomposability of codes, McEliece type systems.*

## Введение

Асимметричные кодовые криптосистемы типа Мак-Элиса [1] характеризуются более высокой скоростью операций шифрования и расшифрования по сравнению с соответствующими операциями используемых в настоящее время асимметричных криптосистем. Это связано с тем, что в операциях шифрования и расшифрования кодовых систем используются, как правило, только матричная арифметика и алгоритмы решения систем линейных уравнений над полями небольшой мощности. В системах RSA или Эль-Гамаля выполняются операции с большими числами, что в вычислительном отношении более затратно. Кодовые криптосистемы устойчивы также к атакам с использованием квантовых компьютеров [2], поэтому они рассматриваются как альтернатива современным асимметричным криптосистемам [3].

Многие успешные атаки на кодовые криптосистемы основаны на применении произведения Шура — Адамара к публичному ключу [4 – 11]. В [12] предложена кодовая криптосистема типа Мак-Элиса, основанная на тензорном произведении кодов $C_1 \otimes C_2$. Настоящая работа посвящена исследованию свойств произведения Шура — Адамара для тензорного произведения линейных кодов. Работа имеет следующую структуру. В п. 1 исследуются свойства произведения Шура — Адамара для тензорного произведения кодов, в частности получены необходимые и достаточные условия того, что $s$-я степень тензорного произведения кодов перестановочно эквивалентна прямой сумме кодов. В п. 2 на основе некоторых результатов из п. 1 доказана неразложимость бинарных кодов Рида — Маллера, а также вычислена группа перестановочных автоморфизмов прямой суммы неразложимых кодов.

## 1. Свойства произведения Шура — Адамара для тензорного произведения кодов

Пусть $\mathbb{F}_q$ — поле Галуа мощности $q$. Под линейным $[n, k, d]_q$-кодом будем понимать подпространство размерности $k$ пространства $\mathbb{F}_q^n$ над конечным полем $\mathbb{F}_q$, имеющее кодовое расстояние $d$. Такой код иногда будем называть $[n, k]_q$-кодом. Тензорное произведение $A \otimes B$ для $(k_1 \times n_1)$-матрицы $A$ и $(k_2 \times n_2)$-матрицы $B$ определяется равенством

$$A \otimes B = \begin{pmatrix} a_{1,1}B & \dots & a_{1,n_1}B \\ \vdots & \ddots & \vdots \\ a_{k_1,1}B & \dots & a_{k_1,n_1}B \end{pmatrix}.$$

Для векторов $\mathbf{a} = (a_1, \dots, a_n)$ и $\mathbf{b} = (b_1, \dots, b_n)$ из $\mathbb{F}_q^n$ рассмотрим покоординатное умножение $\mathbf{a} \star \mathbf{b} = (a_1 b_1, \dots, a_n b_n)$, также называемое произведением Шура или произведением Адамара [13]. Произведением Шура — Адамара $(k_A \times n)$-матрицы $A = (\mathbf{a}_i)_{i=1}^{k_A}$ и $(k_B \times n)$-матрицы $B = (\mathbf{b}_i)_{i=1}^{k_B}$ называется $(k_A k_B \times n)$-матрица вида

$$A \star B = \begin{pmatrix} \mathbf{a}_1 \star \mathbf{b}_1 \\ \vdots \\ \mathbf{a}_1 \star \mathbf{b}_{k_B} \\ \mathbf{a}_2 \star \mathbf{b}_1 \\ \vdots \\ \mathbf{a}_{k_A} \star \mathbf{b}_{k_B} \end{pmatrix}. \tag{1}$$

Произведение $A \star A$ будем обозначать $A^2$. Для натурального $n$ симметрическую группу перестановок множества $\underline{n} = \{1, \dots, n\}$ обозначим $\mathcal{S}_n$. Под действием перестановки $\sigma \in \mathcal{S}_n$ на вектор $\mathbf{a} = (a_1, \dots, a_n)$ будем понимать перестановку координат этого вектора в соответствии с $\sigma$:

$$\sigma(\mathbf{a}) = (a_{\sigma(1)}, \dots, a_{\sigma(n)});$$

$\sigma(U) = \{\sigma(\mathbf{u}) : \mathbf{u} \in U\}$, где $U$ — множество векторов длины $n$. Композицию перестановок $\sigma, \delta \in \mathcal{S}_n$ обозначим $\sigma \circ \delta$, причём $(\sigma \circ \delta)(i) = \sigma(\delta(i))$. Пусть $r(A)$ — ранг матрицы $A$. Единичную $(n \times n)$-матрицу обозначим $I_n$.

**Лемма 1.** Произведение Шура — Адамара обладает следующими свойствами:

1) $\forall \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n \, (\mathbf{a} \star \mathbf{b} = \mathbf{b} \star \mathbf{a})$;
2) $\forall \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n \, \forall \sigma \in \mathcal{S}_n \, \forall a, b \in \mathbb{F} \, \big(\sigma(a\mathbf{a} \star b\mathbf{b}) = ab(\sigma(\mathbf{a}) \star \sigma(\mathbf{b}))\big)$;
3) $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_q^n \, \big((\mathbf{a} + \mathbf{b}) \star \mathbf{c} = (\mathbf{a} \star \mathbf{c}) + (\mathbf{b} \star \mathbf{c})\big)$;
4) для любых матриц $A$ и $B$ размера $(k_A \times n)$ и $(k_B \times n)$ соответственно

$$r(A \star B) \leqslant k_A k_B;$$

5) $\forall \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n \, \forall \mathbf{c}, \mathbf{d} \in \mathbb{F}_q^m \, \big((\mathbf{a} \otimes \mathbf{c}) \star (\mathbf{b} \otimes \mathbf{d}) = (\mathbf{a} \star \mathbf{b}) \otimes (\mathbf{c} \star \mathbf{d})\big)$;
6) для любых матриц $A$ и $B$ размера $(k_A \times n)$ и $(k_B \times n)$ соответственно и любой $(n \times n)$-матрицы перестановки $Q$ выполняется равенство

$$(AQ) \star (BQ) = (A \star B)Q.$$

***Доказательство.*** Свойства 1–4 вытекают из определения произведения Шура — Адамара для векторов и матриц.

Докажем свойство 5. Пусть $\mathbf{a} = (a_1, \dots, a_n)$, $\mathbf{b} = (b_1, \dots, b_n)$. По определению тензорного произведения и произведения Шура — Адамара получаем

$$\begin{aligned}(\mathbf{a} \otimes \mathbf{c}) \star (\mathbf{b} \otimes \mathbf{d}) &= (a_1\mathbf{c}, \dots, a_n\mathbf{c}) \star (b_1\mathbf{d}, \dots, b_n\mathbf{d}) = \\ &= (a_1 b_1 \mathbf{c} \star \mathbf{d}, \dots, a_n b_n \mathbf{c} \star \mathbf{d}) = (\mathbf{a} \star \mathbf{b}) \otimes (\mathbf{c} \star \mathbf{d}),\end{aligned}$$

где запись $(\mathbf{x}, \mathbf{y})$ означает приписывание вектора $\mathbf{y}$ справа к вектору $\mathbf{x}$.

Докажем свойство 6. Из определения (1) произведения Шура — Адамара для матриц получаем, что

$$(AQ) \star (BQ) = ((\mathbf{a}_i Q) \star (\mathbf{b}_j Q))_{i \in \underline{k_A}, j \in \underline{k_B}},$$

где $\mathbf{a}_i$ и $\mathbf{b}_j$ — строки матриц $A$ и $B$ соответственно. Из свойства 2 вытекает, что $(\mathbf{a}_i Q) \star \star (\mathbf{b}_j Q) = (\mathbf{a}_i \star \mathbf{b}_j)Q$. Отсюда получаем требуемое равенство. ■

Множество всех порождающих матриц кода $C$ обозначим $\mathcal{G}(C)$. Отметим, что $C = \mathcal{L}(G_C)$ для всех $G_C$ из $\mathcal{G}(C)$, где $\mathcal{L}(A)$ — линейная оболочка, натянутая на строки матрицы $A$. Для $[n, k_1]_q$-кода $C_1$ и $[n, k_2]_q$-кода $C_2$ их произведение Шура — Адамара определяется как линейная оболочка, натянутая на множество векторов $\{\mathbf{a} \star \mathbf{b} : \mathbf{a} \in C_1, \mathbf{b} \in C_2\}$:

$$C_1 \star C_2 = \mathcal{L}\left(\{\mathbf{a} \star \mathbf{b} : \mathbf{a} \in C_1, \mathbf{b} \in C_2\}\right). \tag{2}$$

Квадратом кода $C$ называется пространство (код) $C \star C = C^2$. Аналогично может быть определена любая степень $s$ кода $C$: $C^s = C^{s-1} \star C$. Для $\tau \subseteq \underline{n}$ через $\mathbb{F}_q^n(\tau)$ обозначим $|\tau|$-мерное координатное подпространство пространства $\mathbb{F}_q^n$, полагая, что все векторы из $\mathbb{F}_q^n(\tau)$ на координатах из $\underline{n} \setminus \tau$ имеют нулевые значения.

Носителем вектора $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ назовём множество $\mathrm{supp}(\mathbf{x}) = \{i \in \underline{n} : x_i \neq 0\}$, а носителем $A \subseteq \mathbb{F}_q^n$ — множество $\mathrm{supp}(A) = \bigcup\limits_{\mathbf{a} \in A} \mathrm{supp}(\mathbf{a})$. Говорят, что $[n, k, d]_q$-код $C$ не имеет фиктивных или нулевых координат, если $\mathrm{supp}(C) = \underline{n}$. Заметим, что $[n, k, d]_q$-код $C$ для $n > 1$ не имеет фиктивных координат тогда и только тогда, когда кодовое расстояние кода $C^\perp$, дуального к коду $C$, не менее двух.

**Лемма 2.** Пусть $C$ — $[n, k]_q$-код, $\tau \subseteq \underline{n}$ — множество ненулевых координат кода. Тогда $C \star \mathbb{F}_q^n = \mathbb{F}_q^n \star C = \mathbb{F}_q^n(\tau)$.

***Доказательство.*** Пусть $M_C$ и $M_{\mathbb{F}_q^n}$ — матрицы, строками которых являются все кодовые слова кода $C$ и пространства $\mathbb{F}_q^n$ соответственно. По условию в матрице $M_C$ столбцы с номерами из $\tau$ ненулевые, а остальные нулевые, следовательно, в матрице $M_C \star M_{\mathbb{F}_q^n}$ все столбцы с номерами из $\underline{n} \setminus \tau$ нулевые, а с номерами из $\tau$ — ненулевые, причём в $M_C \star M_{\mathbb{F}_q^n}$ найдутся $|\tau|$ строк, образующих базис пространства $\mathbb{F}_q^n(\tau)$. Последнее следует из того, что для каждого $i \in \tau$ в матрице $M_C$ найдётся строка с ненулевой $i$-й координатой и найдётся строка в матрице $M_{\mathbb{F}_q^n}$ веса один также с ненулевой $i$-й координатой. Поэтому из (2) имеем $C \star \mathbb{F}_q^n = \mathbb{F}_q^n \star C = \mathcal{L}(M_C \star M_{\mathbb{F}_q^n}) = \mathbb{F}_q^n(\tau)$. ■

Из леммы 2 получаем, что если $[n, k]_q$-код $C$ не имеет фиктивных координат, то

$$C \star \mathbb{F}_q^n = \mathbb{F}_q^n \star C = \mathbb{F}_q^n. \tag{3}$$

В соответствии с [14] введём определение внешней (прямой) и внутренней суммы двух кодов. Для этого понадобятся понятия проекции вектора и проекции пространства. Под проекцией вектора $\mathbf{x} = (x_1, \ldots, x_n)$ на множество $\tau \subseteq \underline{n}$ будем понимать $n$-мерный вектор $\mathbf{y} = \Pi_\tau(\mathbf{x})$, такой, что $y_i = x_i$ для $i \in \tau$, и $y_i = 0$, если $i \notin \tau$. Проекцию кода $C$ на множество номеров $\tau$ определим по правилу

$$\Pi_\tau(C) = \{\Pi_\tau(\mathbf{c}) : \mathbf{c} \in C\}. \tag{4}$$

Отметим, что $\mathrm{supp}(\Pi_\tau(\mathbf{x})) \subseteq \tau$. Прямой суммой кодов $C \subseteq \mathbb{F}_q^n$ и $D \subseteq \mathbb{F}_q^m$ назовём код

$$C \oplus D = \{(\mathbf{c}, \mathbf{d}) : \mathbf{c} \in C, \mathbf{d} \in D\} \subseteq \mathbb{F}_q^{n+m}. \tag{5}$$

Порождающая матрица $G_{C\oplus D}$ кода $C \oplus D$ может быть представлена в блочно-диагональном виде:

$$G_{C\oplus D} = \operatorname{diag}(G_C, G_D) = \begin{pmatrix} G_C & 0 \\ 0 & G_D \end{pmatrix}.$$

Пусть $\tau = \{1, \ldots, n\}$, $\nu = \{n+1, \ldots, n+m\}$. Коду $C$ сопоставим подкод $\tilde{C} = \Pi_\tau(C \oplus D)$ кода $C \oplus D$, а коду $D$ — подкод $\tilde{D} = \Pi_\nu(C \oplus D)$. Тогда

$$C \oplus D = \tilde{C} + \tilde{D},\ \operatorname{supp}(\tilde{C}) \cap \operatorname{supp}(\tilde{D}) = \varnothing,\ \dim(\tilde{C}) = \dim(C),\ \dim(\tilde{D}) = \dim(D), \quad (6)$$

где

$$\tilde{C} + \tilde{D} = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \tilde{C}, \mathbf{b} \in \tilde{D}\} \subseteq \mathbb{F}_q^{n+m} \quad (7)$$

— внутренняя сумма кодов. Таким образом, прямая сумма $C \oplus D$ кодов $C$ и $D$ представима в виде внутренней суммы подкодов $\tilde{C} \subseteq C \oplus D$ и $\tilde{D} \subseteq C \oplus D$ с непересекающимися носителями. При этом заметим, что могут найтись два подкода $\tilde{C}' \subseteq C \oplus D$ и $\tilde{D}' \subseteq C \oplus D$ с пересекающимися носителями, такие, что $\tilde{C}' + \tilde{D}' = C \oplus D$.

По аналогии с (5) и (7) определяется внешняя и внутренняя сумма $u$ кодов, $u \in \mathbb{N}$.

**Лемма 3** [13]. Для кодов $C_1$, $C_2$ и $C_3$ из $\mathbb{F}_q^n$ выполняются следующие равенства:

$$(C_1 + C_2) \star C_3 = C_1 \star C_3 + C_2 \star C_3. \quad (8)$$

Из леммы 3 и представления (6) вытекает, что

$$(C \oplus D) \star E = (\tilde{C} + \tilde{D}) \star E = \tilde{C} \star E + \tilde{D} \star E = \tilde{C} \star \Pi_{\operatorname{supp}(\tilde{C})}(E) + \tilde{D} \star \Pi_{\operatorname{supp}(\tilde{D})}(E)$$

для любых кодов $C \subseteq \mathbb{F}_q^n$, $D \subseteq \mathbb{F}_q^m$ и $E \subseteq \mathbb{F}_q^{n+m}$. Тогда

$$(C \oplus D) \star E = (C \star E_1) \oplus (D \star E_2), \quad (9)$$

где код $E_1 \subseteq \mathbb{F}_q^n$ получается из кода $E$ отбрасыванием последних $m$ координат в кодовых словах из $E$, а код $E_2 \subseteq \mathbb{F}_q^m$ — отбрасыванием первых $n$ координат. Из (9), в частности, вытекает, что

$$(C \oplus D)^2 = C^2 \oplus D^2.$$

В общем случае

$$(C_1 \oplus \ldots \oplus C_n)^s = C_1^s \oplus \ldots \oplus C_n^s.$$

**Лемма 4.** Для любых $G_{C_1} = (\mathbf{g}_i^1)_{i=1}^{k_1} \in \mathcal{G}(C_1)$ и $G_{C_2} = (\mathbf{g}_i^2)_{i=2}^{k_2} \in \mathcal{G}(C_2)$ выполняется равенство

$$C_1 \star C_2 = \mathcal{L}\left(G_{C_1} \star G_{C_2}\right).$$

***Доказательство.*** По определению кода $C_1 \star C_2$ произвольный вектор $\mathbf{c} \in C_1 \star C_2$ представляет собой некоторую линейную комбинацию вида

$$\mathbf{c} = \sum_{i=1}^{l_\mathbf{c}} \gamma_i(\mathbf{a}_i \star \mathbf{b}_i),\ \gamma_i \neq 0,\ \mathbf{a}_i \in C_1,\ \mathbf{b}_i \in C_2,$$

где $l_\mathbf{c}$ для каждого $\mathbf{c}$ в общем случае своё. Так как $\mathbf{a}_i = \sum\limits_{l=1}^{k_A} \alpha_l \mathbf{g}_l^1$, $\mathbf{b}_i = \sum\limits_{r=1}^{k_B} \beta_r \mathbf{g}_r^2$, то из п. 3 леммы 1 получаем

$$\mathbf{c} = \sum_{i=1}^{l_\mathbf{c}} \gamma_i(\mathbf{a}_i \star \mathbf{b}_i) = \sum_{i=1}^{l_\mathbf{c}} \gamma_i \left(\sum_{l=1}^{k_A} \alpha_l \mathbf{g}_l^1 \star \sum_{r=1}^{k_B} \beta_r \mathbf{g}_r^2\right) = \sum_{i=1}^{l_\mathbf{c}} \gamma_i \left(\sum_{l=1}^{k_A} \sum_{r=1}^{k_B} \alpha_l \beta_r (\mathbf{g}_l^1 \star \mathbf{g}_r^2)\right).$$

Вектор $\sum\limits_{l=1}^{k_A}\sum\limits_{r=1}^{k_B}\alpha_l\beta_r(\mathbf{g}_l^1\star\mathbf{g}_r^2)$ представляет собой линейную комбинацию строк матрицы $G_{C_1}\star G_{C_2}$, поэтому вектор $\mathbf{c}$ также является линейной комбинацией строк этой матрицы. Следовательно, $C_1\star C_2\subseteq\mathcal{L}\left(G_{C_1}\star G_{C_2}\right)$. Обратное вложение очевидно. ■

Пусть $C_i$ — $[n_i,k_i,d_i]_q$-код, $G_{C_i}$ — его порождающая матрица, $i=1,2$. Тензорным произведением кодов $C_1$ и $C_2$ называется $[n_1n_2,k_1k_2,d_1d_2]_q$-код, обозначаемый $C_1\otimes C_2$, порождающая матрица которого имеет вид $G_{C_1\otimes C_2}=G_{C_1}\otimes G_{C_2}$. Дуальным к тензорному произведению кодов является код

$$(C_1\otimes C_2)^{\perp}=(\mathbb{F}_q^{n_1}\otimes C_2^{\perp})+(C_1^{\perp}\otimes\mathbb{F}_q^{n_2}). \tag{10}$$

По определению тензорного произведения получаем

$$C_1\otimes C_2\subseteq\mathbb{F}_q^{n_1}\otimes C_2=\underbrace{C_2\oplus\ldots\oplus C_2}_{n_1}. \tag{11}$$

**Лемма 5** [15]. Пусть $C$ — $[n,k]_q$-код, тогда

$$\dim(C^2)\leqslant\min\left\{n,k(k+1)/2\right\}. \tag{12}$$

В [15] для случайного линейного кода $C$ получены теоретические оценки вероятности выполнения равенства в (12).

Исследуем свойства произведения Шура — Адамара для кода $C_1\otimes C_2$.

**Лемма 6.** Пусть $C_i$ — $[n_i,k_i,d_i]_q$-код с порождающей матрицей $G_{C_i}$, $i=1,2$. Тогда

$$\dim((C_1\otimes C_2)^2)\leqslant\min\left\{n_2\dim(C_1^2),\ n_1\dim(C_2^2),\ k_1k_2(k_1k_2+1)/2\right\}. \tag{13}$$

***Доказательство.*** Неравенство $\dim((C_1\otimes C_2)^2)\leqslant n_1\dim(C_2^2)\leqslant n_1n_2$ вытекает из (11) и того, что $\dim(C_2^2)\leqslant n_2$. В [16] показано, что для любых матриц $A$ и $B$ всегда найдутся две такие матрицы перестановок $P_l$ и $P_r$ подходящих размеров, что

$$A\otimes B=P_l(B\otimes A)P_r. \tag{14}$$

Поэтому код $C_1\otimes C_2$ перестановочно эквивалентен некоторому подкоду кода $\mathbb{F}_q^{n_2}\otimes C_1$ и $\dim((C_1\otimes C_2)^2)\leqslant n_2\dim(C_1^2)$. Тогда (13) следует из (12). ■

Важной для дальнейшего является

**Теорема 1.** Пусть $C_1,D_1\subseteq\mathbb{F}_q^{n_1}$, $C_2,D_2\subseteq\mathbb{F}_q^{n_2}$. Тогда

$$(C_1\otimes C_2)\star(D_1\otimes D_2)=(C_1\star D_1)\otimes(C_2\star D_2).$$

***Доказательство.*** Как следует из леммы 4, для доказательства теоремы 1 достаточно показать, что для любых порождающих матриц $G_{C_1}$, $G_{C_2}$, $G_{D_1}$, $G_{D_2}$ выполняется равенство

$$\mathcal{L}\left((G_{C_1}\otimes G_{C_2})\star(G_{D_1}\otimes G_{D_2})\right)=\mathcal{L}\left((G_{C_1}\star G_{D_1})\otimes(G_{C_2}\star G_{D_2})\right).$$

Пусть $\mathbf{c}_i^1$, $\mathbf{c}_l^2$, $\mathbf{d}_j^1$, $\mathbf{d}_m^2$ — произвольные строки матриц $G_{C_1}$, $G_{C_2}$, $G_{D_1}$ и $G_{D_2}$ соответственно. Тогда из п. 5 леммы 1 получаем

$$(\mathbf{c}_i^1\otimes\mathbf{c}_l^2)\star(\mathbf{d}_j^1\otimes\mathbf{d}_m^2)=(\mathbf{c}_i^1\star\mathbf{d}_j^1)\otimes(\mathbf{c}_l^2\star\mathbf{d}_m^2). \tag{15}$$

Заметим, что в левой части равенства (15) стоит представление некоторой строки из $(G_{C_1} \otimes G_{C_2}) \star (G_{D_1} \otimes G_{D_2})$, а в правой — некоторой строки из $(G_{C_1} \star G_{D_1}) \otimes (G_{C_2} \star G_{D_2})$. Таким образом, каждая строка матрицы $(G_{C_1} \otimes G_{C_2}) \star (G_{D_1} \otimes G_{D_2})$ является строкой матрицы $(G_{C_1} \star G_{D_1}) \otimes (G_{C_2} \star G_{D_2})$, и наоборот. ■

Из (3) и теоремы 1 вытекает, что если $C_1 = \mathbb{F}_q^{n_1}$ и код $D_1$ не имеет фиктивных координат (или $D_1 = \mathbb{F}_q^{n_1}$ и код $C_1$ не имеет фиктивных координат), то

$$(C_1 \otimes D_1) \star (C_2 \otimes D_2) = \mathbb{F}_q^{n_1} \otimes C_2 \star D_2.$$

**Теорема 2.** Пусть $\tau_i \subseteq \underline{n_i}$ — множество фиктивных координат $[n_i, k_i]_q$-кода $C_i^\perp$, $i = 1, 2$. Тогда

1) $(C_1 \otimes C_2)^2 = C_1^2 \otimes C_2^2$;
2) $((C_1 \otimes C_2)^\perp)^2 = \mathbb{F}_q^{n_1} \otimes (C_2^\perp)^2 + \mathbb{F}_q^{n_1}(\underline{n_1} \setminus \tau_1) \otimes \mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2) + (C_1^\perp)^2 \otimes \mathbb{F}_q^{n_2}$.

***Доказательство.*** Утверждение 1 вытекает непосредственно из теоремы 1 при $D_1 = C_1$ и $D_2 = C_2$. Докажем утверждение 2. Из (10) и (8) получаем

$$\begin{aligned}((C_1 \otimes C_2)^\perp)^2 &= (\mathbb{F}_q^{n_1} \otimes C_2^\perp + C_1^\perp \otimes \mathbb{F}_q^{n_2})^2 = \\ &= (\mathbb{F}_q^{n_1} \otimes C_2^\perp)^2 + (\mathbb{F}_q^{n_1} \otimes C_2^\perp) \star (C_1^\perp \otimes \mathbb{F}_q^{n_2}) + (C_1^\perp \otimes \mathbb{F}_q^{n_2})^2.\end{aligned}$$

Из теоремы 1 следует, что $(\mathbb{F}_q^{n_1} \otimes C_2^\perp)^2 = \mathbb{F}_q^{n_1} \otimes (C_2^\perp)^2$, $(C_1^\perp \otimes \mathbb{F}_q^{n_2})^2 = (C_1^\perp)^2 \otimes \mathbb{F}_q^{n_2}$, а по лемме 2 имеем

$$(\mathbb{F}_q^{n_1} \star C_1^\perp) \otimes (C_2^\perp \star \mathbb{F}_q^{n_2}) = \mathbb{F}_q^{n_1}(\underline{n_1} \setminus \tau_1) \otimes \mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2).$$

Собрав вместе представления слагаемых, получаем утверждение 2 теоремы. ■

Из п. 1 теоремы 2 для натурального $s \geqslant 1$ получаем равенство

$$(C_1 \otimes C_2)^s = C_1^s \otimes C_2^s. \tag{16}$$

Коды $C$ и $D$ из $\mathbb{F}_q^n$ будем называть перестановочно эквивалентными и писать $C \sim D$, если найдётся такая перестановка $\sigma \in \mathcal{S}_n$, что $\sigma(C) = D$. Подгруппа $\mathcal{S}_n$ таких перестановок $\sigma$, что $\sigma(C) = C$, называется группой перестановочных автоморфизмов и обозначается здесь как $\mathrm{PAut}(C)$.

**Следствие 1.** Пусть $\tau_i \subseteq \underline{n_i}$ — множество фиктивных координат $[n_i, k_i]_q$-кода $C_i^\perp$, $i = 1, 2, 3$. Имеют место следующие свойства:

1) если $\tau_1 = \varnothing$, то $((C_1 \otimes C_2)^\perp)^2 \sim (\mathbb{F}_q^{n_2}(\tau_2) \otimes ((C_1^\perp)^2)^\perp)^\perp$;
2) если $\tau_2 = \varnothing$, то $((C_1 \otimes C_2)^\perp)^2 = (\mathbb{F}_q^{n_1}(\tau_1) \otimes ((C_2^\perp)^2)^\perp)^\perp$;
3) если $\tau_2 = \varnothing$, $n_1 = n_3$ и $\tau_1 = \tau_3$, то $((C_1 \otimes C_2)^\perp)^2 = ((C_3 \otimes C_2)^\perp)^2$;
4) если $\tau_1 = \tau_2 = \varnothing$, то $((C_1 \otimes C_2)^\perp)^2 = \mathbb{F}_q^{n_1 n_2}$.

***Доказательство.***

Докажем свойство 1. Из утверждения 2 теоремы 2 и условия $\tau_1 = \varnothing$ вытекает цепочка равенств

$$\begin{gathered}((C_1 \otimes C_2)^\perp)^2 = \mathbb{F}_q^{n_1} \otimes (C_2^\perp)^2 + \mathbb{F}_q^{n_1} \otimes \mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2) + (C_1^\perp)^2 \otimes \mathbb{F}_q^{n_2} = \\ = \mathbb{F}_q^{n_1} \otimes ((C_2^\perp)^2 + \mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2)) + (C_1^\perp)^2 \otimes \mathbb{F}_q^{n_2} = \mathbb{F}_q^{n_1} \otimes \mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2) + (C_1^\perp)^2 \otimes \mathbb{F}_q^{n_2}.\end{gathered}$$

Отсюда, последовательно применяя (10) и (14), получаем

$$((C_1 \otimes C_2)^\perp)^2 = (((C_1^\perp)^2)^\perp \otimes (\mathbb{F}_q^{n_2}(\underline{n_2} \setminus \tau_2))^\perp)^\perp \sim (\mathbb{F}_q^{n_2}(\tau_2) \otimes ((C_1^\perp)^2)^\perp)^\perp.$$

Свойство 2 доказывается по аналогии со свойством 1 с естественными упрощениями; свойство 3 вытекает из свойства 2.

Докажем свойство 4. Из п. 2 теоремы 2 и условия 4 получаем

$$((C_1 \otimes C_2)^{\perp})^2 = \mathbb{F}_q^{n_1} \otimes (C_2^{\perp})^2 + \mathbb{F}_q^{n_1} \otimes \mathbb{F}_q^{n_2} + (C_1^{\perp})^2 \otimes \mathbb{F}_q^{n_2} = \mathbb{F}_q^{n_1} \otimes \mathbb{F}_q^{n_2} = \mathbb{F}_q^{n_1 n_2}.$$

Следствие 1 доказано. ■

Рассмотрим некоторые примеры.

**Пример 1.** Пусть $\mathrm{GRS}_{k,n}$ — обобщённый $[n,k]_q$-код Рида — Соломона [17], $s \in \mathbb{N}$. Известно [4], что $\mathrm{GRS}_{k,n}^2 = \mathrm{GRS}_{\min\{2k-1,n\},n}$. Отсюда следует

$$\mathrm{GRS}_{k,n}^s = \mathrm{GRS}_{\min\{s(k-1)+1,n\},n}. \tag{17}$$

Из (16) следует, что $s$-я степень тензорного произведения обобщённых кодов Рида — Соломона есть тензорное произведение обобщённых кодов Рида — Соломона. Пусть $C_i = \mathrm{GRS}_{k_i,n_i}$, $i = 1,2$. Если существует такое $s$, что

$$\left\lceil \frac{n_1 - 1}{k_1 - 1} \right\rceil \leqslant s < \left\lfloor \frac{n_2 - 1}{k_2 - 1} \right\rfloor, \tag{18}$$

то из (16), (17) и $\mathrm{GRS}_{s(k_1-1)+1,n_1} = \mathrm{GRS}_{n_1,n_1} = \mathbb{F}_q^{n_1}$ получаем

$$(\mathrm{GRS}_{k_1,n_1} \otimes \mathrm{GRS}_{k_2,n_2})^s = (\mathrm{GRS}_{k_1,n_1})^s \otimes (\mathrm{GRS}_{k_2,n_2})^s = \mathbb{F}_q^{n_1} \otimes \mathrm{GRS}_{s(k_2-1)+1,n_2} \neq \mathbb{F}_q^{n_1} \otimes \mathbb{F}_q^{n_2}.$$

Другими словами, $s$-я степень тензорного произведения обобщённых кодов Рида — Соломона, при выполнении неравенств (18), равна прямой сумме нетривиальных обобщённых кодов Рида — Соломона. Так как обобщённый код Рида — Соломона не имеет фиктивных координат, из п. 4 следствия 1 получаем, что $((\mathrm{GRS}_{k_1,n_1} \otimes \mathrm{GRS}_{k_2,n_2})^{\perp})^s = \mathbb{F}_q^{n_1 n_2}$ для всех $s \geqslant 2$.

**Пример 2.** Пусть $\mathrm{RM}(r,m)$ — бинарный код Рида — Маллера порядка $r$ и длины $2^m$. Из результатов работы [5] следует, что $(\mathrm{RM}(r,m))^s = \mathrm{RM}(\min\{m, sr\}, m)$. Тогда из (16) получаем, что $s$-я степень тензорного произведения бинарных кодов Рида — Маллера есть тензорное произведение бинарных кодов Рида — Маллера, но большего порядка. Пусть $C_i = \mathrm{RM}(r_i, m_i)$, $i = 1,2$. Если существует такое $s \in \mathbb{N}$, что

$$\left\lceil \frac{2^{m_1}}{r_1} \right\rceil \leqslant s < \left\lfloor \frac{2^{m_2}}{r_2} \right\rfloor,$$

то из равенства $\mathrm{RM}(m_1, m_1) = \mathbb{F}_2^{2^{m_1}}$ вытекает, что $s$-я степень тензорного произведения бинарных кодов Рида — Маллера есть прямая сумма нетривиальных кодов Рида — Маллера большего порядка:

$$\begin{aligned}(\mathrm{RM}(r_1,m_1) \otimes \mathrm{RM}(r_2,m_2))^s &= (\mathrm{RM}(r_1,m_1))^s \otimes (\mathrm{RM}(r_2,m_2))^s = \\ &= \mathbb{F}_2^{2^{m_1}} \otimes \mathrm{RM}(sr_2, m_2) \neq \mathbb{F}_2^{2^{m_1}} \otimes \mathbb{F}_2^{2^{m_2}}.\end{aligned}$$

Учитывая, что бинарный код Рида — Маллера не имеет фиктивных координат, из п. 4 следствия 1 получаем, что $((\mathrm{RM}(r_1,m_1) \otimes \mathrm{RM}(r_2,m_2))^{\perp})^s = \mathbb{F}_2^{2^{m_1+m_2}}$ для всех $s \geqslant 2$.

## 2. Группа перестановочных автоморфизмов прямой суммы неразложимых кодов

### 2.1. Неразложимые коды

Говорят, что код $C$ *разложимый*, если он перестановочно эквивалентен прямой сумме двух или более нетривиальных кодов [18, 19]. В противном случае код называется неразложимым. Под *полным* разложением кода понимается представление перестановочно эквивалентного ему кода в виде прямой суммы неразложимых кодов. Разложимый код в этом случае перестановочно эквивалентен коду с порождающей матрицей, представимой в блочно-диагональном виде с двумя или более блоками на главной диагонали. В [6] построен эффективный алгоритм, позволяющий определить, является ли произвольный линейный код $C$ разложимым, и найти для перестановочно эквивалентного ему кода порождающую матрицу в блочно-диагональном виде. Из (9) вытекает

**Лемма 7.** Пусть $C$ — $n$-мерный код. Тогда

1) если $C$ — разложимый, то для любого натурального $s \geqslant 2$ код $C^s$ разложимый;
2) если $C$ — разложимый, то для любого $D \subseteq \mathbb{F}_q^n$ код $C \star D$ разложимый;
3) если $C^s$ — неразложимый, то и $C^i$ — неразложимый код для всех $i = 1, \ldots, s-1$.

С использованием леммы 7 доказывается

**Теорема 3** [11, лемма 5]. Бинарный код Рида — Маллера $\mathrm{RM}(r, m)$ для $r = 0, \ldots, m-1$ является неразложимым.

### 2.2. Группа перестановочных автоморфизмов

Д. Слепианом в работе [18] доказана следующая

**Теорема 4** [18, теорема 2]. Для каждого $[n, k]_q$-кода $X$ существует $m$ неразложимых кодов $C_1, \ldots, C_m$, таких, что код $X$ перестановочно эквивалентен прямой сумме $C = C_1 \oplus \ldots \oplus C_m$. Это разложение единственно в следующем смысле: если $X \sim C' = C'_1 \oplus \ldots \oplus C'_{m'}$, где $C'_1, \ldots, C'_{m'}$ — неразложимые коды, то $m = m'$ и $C_1 \sim C'_{i_1}, \ldots, C_m \sim C'_{i_m}$, где $i_1, \ldots, i_m$ — натуральные числа от 1 до $m$ в некотором порядке.

В [14] построен алгоритм разложения произвольного линейного кода в прямую сумму неразложимых подкодов. Однако, кроме разложения произвольного кода в прямую сумму кодов, интерес представляет описание группы перестановочных автоморфизмов прямой суммы неразложимых кодов.

Пусть $n \in \mathbb{N}$. Для $a, b \in \underline{n}$, $a < b$, обозначим $\mathcal{S}_n[a; b]$ подгруппу симметрической группы $\mathcal{S}_n$, такую, что всякая перестановка $\sigma$ из $\mathcal{S}_n[a; b]$ переставляет элементы из множества $\{a, \ldots, b\} \subseteq \underline{n}$, а остальные элементы из $\underline{n}$ оставляет на месте: $\sigma(i) \in \{a, \ldots, b\}$, если $i \in \{a, \ldots, b\}$, и $\sigma(i) = i$, если $i \notin \{a, \ldots, b\}$.

Пусть $i \in \underline{u}$, $C_i$ — $[n_i, k_i]_q$-код с группой перестановочных автоморфизмов $\mathrm{PAut}(C_i)$, $k = \sum\limits_{i=1}^{u} k_i$, $n = \sum\limits_{i=1}^{u} n_i$, $\mathbf{n}_i = \sum\limits_{j=1}^{i} n_j$, $\mathbf{n}_0 = 0$. Рассмотрим $[n, k]_q$-код $C = C_1 \oplus \ldots \oplus C_u$. Отметим, что для произвольного вектора

$$\mathbf{x} = (x_1, \ldots, x_{\mathbf{n}_1}, x_{\mathbf{n}_1+1}, \ldots, x_{\mathbf{n}_2}, \ldots, x_{\mathbf{n}_{u-1}+1}, \ldots, x_{\mathbf{n}_u}) \in C$$

выполняется условие $\mathbf{c}_i = (x_{\mathbf{n}_{i-1}+1}, \ldots, x_{\mathbf{n}_i}) \in C_i$ (см. (5)). Пусть

$$\tilde{C}_i = \Pi_{\{\mathbf{n}_{i-1}+1, \ldots, \mathbf{n}_i\}}(C_1 \oplus \ldots \oplus C_u)$$

(определение проекции $\Pi_{\{\mathbf{n}_{i-1}+1, \ldots, \mathbf{n}_i\}}(C_1 \oplus \ldots \oplus C_u)$ см. в (4)),

$$\mathrm{PAut}_{n, [\mathbf{n}_{i-1}+1; \mathbf{n}_i]}(\tilde{C}_i) = \mathrm{PAut}(\tilde{C}_i) \cap \mathcal{S}_n[\mathbf{n}_{i-1} + 1; \mathbf{n}_i].$$

Отсюда вытекает, что группы $\mathrm{PAut}(C_i)$ и $\mathrm{PAut}_{n,[\mathbf{n}_{i-1}+1;\mathbf{n}_i]}(\tilde{C}_i)$ естественно изоморфны; соответствующий изоморфизм обозначим так:

$$\xi_{n,C_i,\mathbf{n}_{i-1}+1,\mathbf{n}_i} : \mathrm{PAut}(C_i) \to \mathrm{PAut}_{n,[\mathbf{n}_{i-1}+1;\mathbf{n}_i]}(\tilde{C}_i).$$

Отметим, что $\mathrm{PAut}(C_i) = \mathcal{S}_{n_i}$ в случае $C_i = \mathbb{F}_q^{n_i}$, поэтому перестановки из группы $\xi_{n,\mathbb{F}_q^{n_i},\mathbf{n}_{i-1}+1,\mathbf{n}_i}(\mathcal{S}_{n_i})$ переставляют произвольным образом координаты с номерами из множества $\{\mathbf{n}_{i-1}+1,\ldots,\mathbf{n}_i\}$, а другие оставляют неподвижными. Нетрудно проверить, что произведение подгрупп

$$\mathrm{PAut}_{n,[\mathbf{n}_0+1;\mathbf{n}_1]}(\tilde{C}_1)\cdot\ldots\cdot\mathrm{PAut}_{n,[\mathbf{n}_{u-1}+1;\mathbf{n}_u]}(\tilde{C}_u)$$

не зависит от порядка следования и является подгруппой группы $\mathcal{S}_n$.

**Лемма 8.** Пусть $C_i$ — неразложимый $[n_i,k_i]_q$-код без фиктивных координат, $i = 1,2$, $C_1 \not\sim C_2$. Тогда

$$\mathrm{PAut}(C_1\oplus C_2) = \mathrm{PAut}_{n_1+n_2,[\mathbf{n}_0+1;\mathbf{n}_1]}(\tilde{C}_1)\cdot\mathrm{PAut}_{n_1+n_2,[\mathbf{n}_1+1;\mathbf{n}_2]}(\tilde{C}_2).$$

***Доказательство.*** Очевидно вложение

$$\mathrm{PAut}_{n_1+n_2,[\mathbf{n}_0+1;\mathbf{n}_1]}(\tilde{C}_1)\cdot\mathrm{PAut}_{n_1+n_2,[\mathbf{n}_1+1;\mathbf{n}_2]}(\tilde{C}_2) \subseteq \mathrm{PAut}(C_1\oplus C_2).$$

Докажем равенство. Предположим, что существует такая перестановка $\pi$ в группе $\mathrm{PAut}(C_1\oplus C_2)$, что

$$\pi \notin \mathrm{PAut}_{n_1+n_2,[\mathbf{n}_0+1;\mathbf{n}_1]}(\tilde{C}_1)\cdot\mathrm{PAut}_{n_1+n_2,[\mathbf{n}_1+1;\mathbf{n}_2]}(\tilde{C}_2). \tag{19}$$

Пусть $\mathbf{0}_i$ — нулевой вектор длины $n_i$, $i = 1,2$. Без потери общности, предположим, что $n_1 \geqslant n_2$. Из определения $\mathrm{PAut}(C_1\oplus C_2)$ вытекает, что

$$\Pi_{\{1,\ldots,n_1\}}(\pi(C_1\oplus C_2)) = \Pi_{\{1,\ldots,n_1\}}(C_1\oplus C_2) = C_1\oplus\{\mathbf{0}_2\}. \tag{20}$$

Рассмотрим множество номеров координат $\tau \subseteq \underline{n_1}$ вида

$$\tau = \{j \in \underline{n_1} : \pi^{-1}(j) \in \underline{n_1}\}$$

и его дополнение $\overline{\tau} = \underline{n_1}\setminus\tau$ до множества $\underline{n_1}$. По предположению, выполняется (19), поэтому из условия $n_1 \geqslant n_2$ вытекает

$$0 < |\tau|, |\overline{\tau}| < n_1.$$

Так как $\pi \in \mathrm{PAut}(C_1\oplus C_2)$, из определения $\tau$ и $\overline{\tau}$ получаем

$$\pi(C_1\oplus\{\mathbf{0}_2\}) \subseteq C_1\oplus C_2,\ \pi(\{\mathbf{0}_1\}\oplus C_2) \subseteq C_1\oplus C_2,$$
$$\Pi_\tau(\pi(C_1\oplus\{\mathbf{0}_2\})) \subseteq C_1\oplus\{\mathbf{0}_2\},\ \Pi_{\overline{\tau}}(\pi(\{\mathbf{0}_1\}\oplus C_2)) \subseteq C_1\oplus\{\mathbf{0}_2\}.$$

Тогда

$$\Pi_\tau(C_1\oplus C_2) = \Pi_\tau(\pi(C_1\oplus C_2)) = \Pi_\tau(\pi(C_1\oplus\{\mathbf{0}_2\})) \subseteq C_1\oplus\{\mathbf{0}_2\}; \tag{21}$$
$$\Pi_{\overline{\tau}}(C_1\oplus C_2) = \Pi_{\overline{\tau}}(\pi(C_1\oplus C_2)) = \Pi_{\overline{\tau}}(\pi(\{\mathbf{0}_1\}\oplus C_2)) \subseteq C_1\oplus\{\mathbf{0}_2\}. \tag{22}$$

Из (21) и (22) следует

$$\Pi_\tau(\pi(C_1 \oplus C_2)) + \Pi_{\overline{\tau}}(\pi(C_1 \oplus C_2)) = \Pi_\tau(C_1 \oplus C_2) + \Pi_{\overline{\tau}}(C_1 \oplus C_2) \subseteq C_1 \oplus \{\mathbf{0}_2\}. \quad (23)$$

Так как для любого линейного кода $D$ длины $n$ и любого множества $\tau \subseteq \underline{n}$ выполняется неравенство

$$\dim(\Pi_\tau(D)) + \dim(\Pi_{\overline{\tau}}(D)) \geqslant \dim(D),$$

то $\dim(\Pi_\tau(C_1 \oplus C_2)) + \dim(\Pi_{\overline{\tau}}(C_1 \oplus C_2)) \geqslant \dim(C_1)$, откуда с учётом (23) имеем

$$\Pi_\tau(C_1 \oplus C_2) + \Pi_{\overline{\tau}}(C_1 \oplus C_2) = C_1 \oplus \{\mathbf{0}_2\}. \quad (24)$$

Так как коды $C_1$ и $C_2$ не содержат фиктивных координат, то $\dim(\Pi_\tau(C_1 \oplus C_2)) > 0$ и $\dim(\Pi_{\overline{\tau}}(C_1 \oplus C_2)) > 0$. Отсюда и из (20) получам, что $\dim(\Pi_\tau(C_1 \oplus C_2)) < k_1$ и $\dim(\Pi_{\overline{\tau}}(C_1 \oplus C_2)) < k_1$. Тогда (с учётом (24)) код $C_1$ изоморфен внешней прямой сумме двух нетривиальных кодов, так как носители кодов $\Pi_\tau(C_1 \oplus C_2)$ и $\Pi_{\overline{\tau}}(C_1 \oplus C_2)$ не пересекаются. Однако это противоречит неразложимости кода $C_1$. ■

Из леммы 8 вытекает

**Лемма 9.** Пусть $u \in \mathbb{N}$, $C_i$ — неразложимый $[n_i, k_i]_q$-код $C_1$, $i = 1, \ldots, u$, $C_l \not\sim C_j$ для всех $l \neq j$. Тогда

$$\mathrm{PAut}(C_1 \oplus \ldots \oplus C_u) = \mathrm{PAut}_{n,[\mathbf{n}_0+1;\mathbf{n}_1]}(\tilde{C}_1) \cdot \ldots \cdot \mathrm{PAut}_{n,[\mathbf{n}_{u-1}+1;\mathbf{n}_u]}(\tilde{C}_u),$$

где $n = \sum\limits_{i=1}^{u} n_i$.

Рассмотрим группу $\mathcal{P}_n$ $(n \times n)$-матриц перестановок и естественный изоморфизм $\alpha_n : \mathcal{S}_n \to \mathcal{P}_n$. Для натуральных $n$ и $m$ в группе $\mathcal{P}_{nm}$ рассмотрим подгруппу

$$\mathcal{P}_{n,m} = \{B \otimes I_n : B \in \mathcal{P}_m\},$$

где $I_n$ — единичная $(n \times n)$-матрица. Пусть $\mathcal{S}_{n,m} = \alpha_{nm}^{-1}(\mathcal{P}_{n,m})$. Отметим, что группа $\mathcal{S}_{n,m}$ изоморфна группе $\mathcal{S}_m$, причём изоморфизм $\beta_{n,m} : \mathcal{S}_m \to \mathcal{S}_{n,m}$ можно записать в виде

$$\beta_{n,m}(\gamma) = \alpha_{nm}^{-1}(\alpha_n(\gamma) \otimes I_n), \gamma \in \mathcal{S}_m.$$

**Лемма 10.** Пусть $C_i = \sigma_i(C)$ — $[n_i, k_i]_q$-код, где $C$ — неразложимый $[n, k]_q$-код без фиктивных координат, $\sigma_i \in \mathcal{S}_n$, $i = 1, 2$. Тогда

$$\mathrm{PAut}(C_1 \oplus C_2) = \left\{ \begin{array}{c|c} (\hat{\sigma}_1 \circ \omega_1 \circ \dot{\sigma}_1) \circ & \hat{\sigma}_i = \xi_{2n,\mathbb{F}_q^n,\mathbf{n}_{i-1}+1,\mathbf{n}_i}(\sigma_i), \\ (\hat{\sigma}_2 \circ \omega_2 \circ \dot{\sigma}_2) \circ & \dot{\sigma}_i = \xi_{2n,\mathbb{F}_q^n,\mathbf{n}_{i-1}+1,\mathbf{n}_i}(\sigma_{\gamma^{-1}(i)}^{-1}), \\ \beta_{n,2}(\gamma) & \omega_i \in \mathrm{PAut}_{2n,[\mathbf{n}_{i-1}+1;\mathbf{n}_i]}(\tilde{C}), \\ & i = 1, 2,\ \gamma \in \mathcal{S}_2. \end{array} \right\} \quad (25)$$

***Доказательство.*** Сначала заметим, что множество, определённое в правой части равенства (25), является подгруппой группы $\mathcal{S}_{2n}$. Непосредственная проверка показывает, что любая перестановка из правой части (25) не меняет код $C_1 \oplus C_2$, а композиция двух перестановок из этого множества представима в таком же виде. Отсюда следует, что это множество является подгруппой и содержится в $\mathrm{PAut}(C_1 \oplus C_2)$. Для доказательства совпадения этих групп достаточно повторить рассуждения из доказательства леммы 8, положив $n_1 = n_2 = n$ и заменив $\mathrm{PAut}_{n_1+n_2,[\mathbf{n}_0+1,\mathbf{n}_1]}(\tilde{C}_1) \times$ $\times \mathrm{PAut}_{n_1+n_2,[\mathbf{n}_1+1,\mathbf{n}_2]}(\tilde{C}_2)$ на правую часть из равенства (25). ■

Лемма 10 может быть обобщена на случай суммы $u$ одинаковых кодов.

**Лемма 11.** Пусть $C_i = \sigma_i(C)$ — $[n_i, k_i]_q$-код, где $C$ — неразложимый $[n, k]_q$-код без фиктивных координат, $\sigma_i \in \mathcal{S}_n$, $i \in \underline{u}, u \in \mathbb{N}$. Тогда

$$\mathrm{PAut}(C_1 \oplus \ldots \oplus C_u) = \left\{ \begin{array}{c} (\hat{\sigma}_1 \circ \omega_1 \circ \dot{\sigma}_1)\circ \\ \ldots \\ (\hat{\sigma}_u \circ \omega_u \circ \dot{\sigma}_u)\circ \\ \beta_{n,u}(\gamma) \end{array} : \begin{array}{c} \hat{\sigma}_i = \xi_{un,\mathbb{F}_q^n,\mathbf{n}_{i-1}+1,\mathbf{n}_i}(\sigma_i), \\ \dot{\sigma}_i = \xi_{un,\mathbb{F}_q^n,\mathbf{n}_{i-1}+1,\mathbf{n}_i}(\sigma^{-1}_{\gamma^{-1}(i)}), \\ \omega_i \in \mathrm{PAut}_{un,[\mathbf{n}_{i-1}+1;\mathbf{n}_i]}(\tilde{C}), \\ i \in \underline{u},\ \gamma \in \mathcal{S}_u. \end{array} \right\} \tag{26}$$

**Следствие 2.** Пусть $u \in \mathbb{N}$, $C_1 = \ldots = C_u = \mathrm{RM}(1, m)$, тогда

$$\mathrm{PAut}(\mathbb{F}_2^u \otimes \mathrm{RM}(1, m)) = \ldots = \mathrm{PAut}(\mathbb{F}_2^u \otimes \mathrm{RM}(m-2, m)) =$$
$$= \mathrm{PAut}_{u2^m,[1,2^m]}(\tilde{C}_1) \cdot \ldots \cdot \mathrm{PAut}_{u2^m,[\mathbf{n}_{u-1}+1;\mathbf{n}_u]}(\tilde{C}_u)\, \mathcal{S}_{2^m,u}.$$

***Доказательство.*** Из (11) получаем, что для $j = 1, \ldots, m-2$

$$\mathbb{F}_2^u \otimes \mathrm{RM}(j, m) = \underbrace{\mathrm{RM}(j, m) \oplus \ldots \oplus \mathrm{RM}(j, m)}_{u}.$$

Как следует из теоремы 3, все бинарные коды Рида — Маллера неразложимые. Поэтому из леммы 11 (см. представление (26) группы перестановочных автоморфизмов перестановочно эквивалентных неразложимых кодов) вытекает, что

$$\mathrm{PAut}(\mathbb{F}_2^u \otimes \mathrm{RM}(1, m)) = \mathrm{PAut}_{u2^m,[1,2^m]}(\tilde{C}_1) \cdot \ldots \cdot \mathrm{PAut}_{u2^m,[\mathbf{n}_{u-1}+1;\mathbf{n}_u]}(\tilde{C}_u)\, \mathcal{S}_{2^m,u}.$$

Известно, что $\mathrm{PAut}(\mathrm{RM}(1, m)) = \ldots = \mathrm{PAut}(\mathrm{RM}(m-2, m))$ [20, с. 400, теорема 24]. Отсюда следует доказываемое утверждение. ■

**Теорема 5.** Пусть $u \in \mathbb{N}$, $l \in \underline{u}$, $r_l \in \mathbb{N}$, $D_l$ — неразложимый $[N_l, K_l]_q$-код без фиктивных координат, при этом $D_i \not\sim D_j$ для всех $i \neq j$, $\mathbf{r}_l = \sum\limits_{j=1}^{l} r_j$, $\mathbf{r}_0 = 0$, $\mathbf{N}_l = \sum\limits_{j=1}^{l} r_j N_j$, $\mathbf{N}_0 = 0$, $n = \sum\limits_{j=1}^{u} r_j N_j$, $\mathbf{n}_{l,i} = \mathbf{N}_{l-1} + i \cdot N_l$. Рассмотрим набор из $\mathbf{r}_u$ линейных $[n_i, r_i]_q$-кодов $C_i$, $i \in \underline{\mathbf{r}_u}$, где $C_{\mathbf{r}_{l-1}+1} = \sigma_{l,1}(D_l)$, …, $C_{\mathbf{r}_l} = \sigma_{l,r_l}(D_l)$ для $l \in \underline{u}$, $\sigma_{l,m} \in \mathcal{S}_{n_l}$, $m \in \underline{r_l}$. Тогда код $C$ вида

$$C = \underbrace{C_{\mathbf{r}_0+1} \oplus \ldots \oplus C_{\mathbf{r}_1}}_{r_1} \oplus \underbrace{C_{\mathbf{r}_1+1} \oplus \ldots \oplus C_{\mathbf{r}_2}}_{r_2} \oplus \ldots \oplus \underbrace{C_{\mathbf{r}_{u-1}+1} \oplus \ldots \oplus C_{\mathbf{r}_u}}_{r_u} \tag{27}$$

имеет группу перестановочных автоморфизмов $\mathrm{PAut}(C) = \mathcal{P}_1 \cdot \ldots \cdot \mathcal{P}_u$, где для $l \in \underline{u}$

$$\mathcal{P}_l = \left\{ \begin{array}{c} (\hat{\sigma}_{l,1} \circ \omega_{l,1} \circ \dot{\sigma}_{l,1})\circ \\ \ldots \\ (\hat{\sigma}_{l,r_l} \circ \omega_{l,r_l} \circ \dot{\sigma}_{l,r_l})\circ \\ \xi_{n,\mathbb{F}_q^{N_l r_l},\mathbf{N}_{l-1}+1,\mathbf{N}_l}(\beta_{N_l,r_l}(\gamma)) \end{array} : \begin{array}{c} \hat{\sigma}_{l,i} = \xi_{n,\mathbb{F}_q^{N_l},\mathbf{n}_{l,i-1}+1,\mathbf{n}_{l,i}}(\sigma_{l,i}), \\ \dot{\sigma}_{l,i} = \xi_{n,\mathbb{F}_q^{N_l},\mathbf{n}_{l,i-1}+1,\mathbf{n}_{l,i}}(\sigma^{-1}_{l,\gamma^{-1}(i)}), \\ \omega_{l,i} \in \mathrm{PAut}_{n,[\mathbf{n}_{l,i-1}+1;\mathbf{n}_{l,i}]}(\tilde{D}_l), \\ i \in \underline{r_l},\ \gamma \in \mathcal{S}_{r_l}. \end{array} \right\}$$

***Доказательство.*** Из (26) вытекает, что группа $\mathrm{PAut}(\oplus_{j=1}^{r_i} C_{\mathbf{r}_{i-1}+j}) \subseteq \mathcal{S}_{r_i N_i}$ изоморфна группе $\mathcal{P}_i \subseteq \mathcal{S}_n$ для $i = 1, \ldots, u$. Поэтому $\mathcal{P}_1 \cdot \ldots \cdot \mathcal{P}_u \subseteq \mathrm{PAut}(C)$. Предположим, что существует перестановка $\sigma$ из $\mathrm{PAut}(C) \setminus \{\mathcal{P}_1 \cdot \ldots \cdot \mathcal{P}_u\}$. Тогда такая перестановка в представлении (27) для $i \neq j$ меняет координаты между двумя слагаемыми-суммами

$$\underbrace{C_{\mathbf{r}_{i-1}+1} \oplus \ldots \oplus C_{\mathbf{r}_i}}_{r_i},\ \underbrace{C_{\mathbf{r}_{j-1}+1} \oplus \ldots \oplus C_{\mathbf{r}_j}}_{r_j}.$$

Но в силу леммы 9 таких перестановок не существует. ■

## Заключение

В работе [12] авторами предложена кодовая криптосистема типа Мак-Элиса $\mathrm{McE}(C_1 \otimes C_2)$ на основе тензорного произведения кодов $C_1$ и $C_2$. Проведённый в [12] анализ криптосистемы $\mathrm{McE}(C_1 \otimes C_2)$ позволил сделать вывод о её высокой стойкости к структурным атакам. При этом в [12] получено, что высокая стойкость достигается даже при использовании кодов $C_1$ и $C_2$, для которых известны структурные атаки для систем $\mathrm{McE}(C_1)$ и $\mathrm{McE}(C_2)$, например в случае использования двоичных кодов Рида — Маллера.

Результаты [4 – 11] показали, что произведение Шура — Адамара позволяет в ряде случаев построить эффективную структурную атаку для рассмотренных в этих работах криптосистем. В настоящей работе с целью уточнения стойкости к структурным атакам системы $\mathrm{McE}(C_1 \otimes C_2)$ из [12] исследуются свойства произведения Шура — Адамара для тензорного произведения $C_1 \otimes C_2$. Полученные результаты, в частности формула (16) из п. 1, позволяют с помощью произведения Шура — Адамара построить по публичной матрице системы $\mathrm{McE}(C_1 \otimes C_2)$ код, перестановочно эквивалентный прямой сумме кодов. Представляется, что это позволит, например, используя алгоритм из [14], найти такую матрицу перестановки $P_\pi$, с помощью которой публичная матрица системы $\mathrm{McE}(C_1 \otimes C_2)$ может быть преобразована к более простому для анализа виду. При этом если группа перестановочных автоморфизмов прямой суммы кодов имеет вид (26), то применение матрицы $P_\pi$, полученной по некоторой степени (в смысле произведения Шура — Адамара) публичной матрицы, к матрице публичного ключа будет корректным. Результаты по уточнению стойкости системы $\mathrm{McE}(C_1 \otimes C_2)$ планируется опубликовать отдельно.

Ю. В. Косолапов признателен рецензенту за внимательное прочтение статьи, в особенности за полезные советы по исправлению найденной ошибки в первоначальном варианте доказательства леммы 8 и обобщению полученных результатов.

## ЛИТЕРАТУРА

1. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory // DSN Progress Report. 1978. P. 42–44.
2. *Sendrier N. and Tillich J. P.* Code-Based Cryptography: New Security Solutions against a Quantum Adversary. ERCIM News. ERCIM, 2016.
3. *Alagic G., Alperin-Sheriff J., Apon D., et al.* Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process. US Department of Commerce, NIST, 2019.
4. *Wieschebrink C.* Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes // LNCS. 2010. V. 6061. P. 61–72.
5. *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.
6. *Deundyak V. M. and Kosolapov Yu. V.* On the strength of asymmetric code cryptosystems based on the merging of generating matrices of linear codes // XVI Intern. Symp. Prob. of Redundancy in Information and Control Systems. Russia, 2019. P. 143–148.
7. *Бородин М. А., Чижов И. В.* Классификация произведений Адамара подкодов коразмерности 1 кодов Рида — Маллера // Дискретная математика. 2020. Т. 32. № 1. С. 115–134.
8. *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.

9. *Vysotskaya V. and Chizhov I.* Equivalence classes of McEliece — Sidelnikov-type cryptosystems // Sixteenth Intern. Workshop Algebraic Combinat. Coding Theory. Svetlogorsk (Kaliningrad region), Russia, 2018. P. 121–124.
10. *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.
11. *Deundyak V. M., Kosolapov Yu. V., and Maystrenko I. A.* On the decipherment of Sidel'nikov-type cryptosystems // LNCS. 2020. V. 12087. P. 20–40.
12. *Deundyak V. M., Kosolapov Y. V., and Lelyuk E. A.* Decoding the tensor product of MLD codes and applications for code cryptosystems // Aut. Control Comp. Sci. 2019. V. 52. No. 7. P. 647–657.
13. *Randriambololona H.* On Products and Powers of Linear Codes under Componentwise Multiplication. arXiv:1312.0022. 2014.
14. *Деундяк В. М., Косолапов Ю. В.* Анализ стойкости некоторых кодовых криптосистем, основанный на разложении кодов в прямую сумму // Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование. 2019. Т. 12. № 3. С. 89–101.
15. *Cascudo I., Cramer R., Mirandola D., and Zemor G.* Squares of random linear codes // IEEE Trans. Inform. Theory. 2015. V. 61. No. 3. P. 1159–1173.
16. *Henderson H. V. and Searle S. R.* The vec-permutation matrix, the vec operator and Kronecker products: A review // Linear and Multilinear Algebra. 1981. V. 9. P. 271–288.
17. *Сидельников В. М.* Теория кодирования. М.: Физматлит, 2008. 324 с.
18. *Slepian D.* Some further theory of group codes // Bell Syst. Tech. J. 1960. V. 39. No. 5. P. 1219–1252.
19. *Assmus E. F.* The category of linear codes // IEEE Trans. Inform. Theory. 1998. V. 44. No. 2. P. 612–629.
20. *Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А.* Теория кодов, исправляющих ошибки. М.: Связь, 1979. 746 с.

## REFERENCES

1. *McEliece R. J.* A Public-Key Cryptosystem Based On Algebraic Coding Theory. DSN Progress Report, 1978, pp. 42–44.
2. *Sendrier N. and Tillich J. P.* Code-Based Cryptography: New Security Solutions against a Quantum Adversary. ERCIM News, ERCIM, 2016.
3. *Alagic G., Alperin-Sheriff J., Apon D., et al.* Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process. US Department of Commerce, NIST, 2019.
4. *Wieschebrink C.* Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. LNCS, 2010, vol. 6061, pp. 61–72.
5. *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.
6. *Deundyak V. M. and Kosolapov Yu. V.* On the strength of asymmetric code cryptosystems based on the merging of generating matrices of linear codes. XVI Intern. Symp. Prob. of Redundancy in Information and Control Systems, Russia, 2019, pp. 143–148.
7. *Borodin M. A. and Chizhov I. V.* Klassifikacija proizvedenij Adamara podkodov korazmernosti 1 kodov Rida — Mallera [Classification of Hadamard products of codimension 1 subcodes of Reed — Muller codes]. Diskret. Matem., 2020, vol. 32, no. 1, pp. 115–134. (in Russian)
8. *Vysotskaya V. V* Kvadrat koda Rida — Mallera i klassy ekvivalentnosti sekretnykh klyuchey kriptosistemy Mak-Elisa — Sidel'nikova [The Reed — Muller code square and equivalence

classes of McEliece — Sidelnikov cryptosystem private keys]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2017, no. 10, pp. 66–68. (in Russian)

9. *Vysotskaya V. and Chizhov I.* Equivalence classes of McEliece — Sidelnikov-type cryptosystems. Sixteenth Intern. Workshop Algebraic Combinat. Coding Theory, Svetlogorsk (Kaliningrad region), Russia, 2018, pp. 121–124.
10. *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)
11. *Deundyak V. M., Kosolapov Yu. V., and Maystrenko I. A.* On the decipherment of Sidel'nikov-type cryptosystems. LNCS, 2020, vol. 12087, pp. 20–40.
12. *Deundyak V. M., Kosolapov Y. V., and Lelyuk E. A.* Decoding the tensor product of MLD codes and applications for code cryptosystems. Aut. Control Comp. Sci, 2019, vol. 52, no. 7, pp. 647–657.
13. *Randriambololona H.* On Products and Powers of Linear Codes under Componentwise Multiplication. arXiv:1312.0022, 2014.
14. *Deundyak V. M. and Kosolapov Yu. V.* Analiz stoykosti nekotorykh kodovykh kriptosistem, osnovannyy na razlozhenii kodov v pryamuyu summu [The use of the direct sum decomposition algorithm for analyzing the strength of some McEliece type cryptosystems]. Vestn. JuUrGU. Ser. Matem. Modelirovanie i Programmirovanie, 2019, vol. 12, no. 3, pp. 89–101. (in Russian)
15. *Cascudo I., Cramer R., Mirandola D., and Zemor G.* Squares of random linear codes. IEEE Trans. Inform. Theory, 2015, vol. 61, no. 3, pp. 1159–1173.
16. *Henderson H. V. and Searle S. R.* The vec-permutation matrix, the vec operator and Kronecker products: A review. Linear and Multilinear Algebra, 1981, vol. 9, pp. 271–288.
17. *Sidel'nikov V. M.* Teorija kodirovanija [Coding Theory]. Moscow: Fizmatlit Publ., 2008. 324 p. (in Russian)
18. *Slepian D.* Some further theory of group codes. Bell Syst. Tech. J., 1960, vol. 39, no. 5, pp. 1219–1252.
19. *Assmus E. F.* The category of linear codes. IEEE Trans. Inform. Theory, 1998, vol. 44, no. 2, pp. 612–629.
20. *MacWilliams F. J. and Sloane N. J. A.* The Theory of Error-Correcting Codes. Amsterdam; New York, North-Holland Pub. Co., 1977. 762 p.

# ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.175.3

## ПЕРЕЧИСЛЕНИЕ ПОМЕЧЕННЫХ ЭЙЛЕРОВЫХ ПЕНТАЦИКЛИЧЕСКИХ ГРАФОВ

В. А. Воблый

*Всероссийский институт научной и технической информации РАН, г. Москва, Россия*

Эйлеров граф — это связный граф, у которого все степени вершин — чётные числа. Пентациклическим графом называется связный граф с $n$ вершинами и $n+4$ рёбрами. Получена явная формула для числа помеченных эйлеровых пентациклических графов с заданным числом вершин, а также найдена соответствующая асимптотика для числа таких графов с большим числом вершин. Доказано, что при равномерном распределении вероятностей вероятность того, что помеченный эйлеров пентациклический граф является блоком (кактусом), асимптотически равна 53/272 (63/272 соответственно).

**Ключевые слова:** *помеченный граф, эйлеров граф, пентациклический граф, блок, перечисление, асимптотика, вероятность.*

DOI 10.17223/20710410/50/6

## ENUMERATION OF LABELED EULERIAN PENTACYCLIC GRAPHS

V. A. Voblyi

*All-Russian Institut for Scientific and Technical Information, Moscow, Russia*

**E-mail:** vitvobl@yandex.ru

An Euler graph is a connected graph in which all degrees of vertices are even numbers. A pentacyclic graph is a connected graph with $n$ vertices and $n+4$ edges. We obtain an explicit formula for the number of labeled Euler pentacyclic graphs with a given number of vertices, and found the corresponding asymptotics for the number of such graphs with a large number of vertices. We prove that, given a uniform probability distribution, the probability that a labeled pentacyclic Euler graph is a block (cactus) is asymptotically 53/272 (63/272), respectively.

**Keywords:** *labeled graph, Eulerian graph, pentacyclic graph, block, enumeration, asymptotics, probability.*

### Введение

Рассматриваются неориентированные простые связные графы.

Для связного графа *точкой сочленения* называется его вершина, после удаления которой вместе с инцидентными ей рёбрами граф становится несвязным, а *мостом* — ребро, удаление которого приводит к несвязности графа. *Блок* — это связный граф без точек сочленения, а также максимальный связный нетривиальный подграф, не имеющий точек сочленения. *Чётным* графом называется граф, у которого все степени

вершин — чётные числа. *Эйлеров граф* — это связный чётный граф [1, с. 22]. *Цикломатическим числом* связного графа называется увеличенная на единицу разность между числом рёбер графа и числом его вершин. *k-Циклический граф* — это связный граф с цикломатическим числом $k$.

Эйлеровы графы находят применение в исследовании операций (задача китайского почтальона) [2], биоинформатике [3] и криптографии [4].

Р. Рид [5] нашёл производящую функцию для числа помеченных чётных графов с заданными числами вершин и рёбер, из которой производящая функция для помеченных эйлеровых графов может быть получена логарифмированием. С. Тазава перечислил по числу вершин помеченные эйлеровы блоки [6]. В [7] найдены явные формулы и асимптотика для числа помеченных эйлеровых бициклических и трициклических графов с заданным числом вершин; в [8] перечислены по числу вершин помеченные эйлеровы тетрациклические графы. В [9] получена явная формула для числа помеченных эйлеровых пентациклических блоков.

В данной работе получена явная формула для числа помеченных эйлеровых пентациклических графов с заданным числом вершин и найдена асимптотика для числа таких графов с большим числом вершин. Доказано, что при равномерном распределении вероятностей помеченный эйлеров пентациклический граф является блоком (кактусом) с вероятностью, асимптотически равной 53/272 (63/272).

## 1. Перечисление графов

**Теорема 1.** Число $E_5(n)$ помеченных эйлеровых пентациклических графов с $n$ вершинами при $n \geqslant 7$ равно

$$E_5(n) = \frac{n!}{29030400}(1360n^7 - 21490n^6 + 101563n^5 + 83930n^4 - 1999655n^3 + \\ +4004840n^2 + 2158812n + 3457440). \tag{1}$$

***Доказательство.*** Пусть $B(n,k)$ — число помеченных $k$-циклических блоков с $n$ вершинами; $B_k(z) = \sum\limits_{n=0}^{\infty} B(n,k)\dfrac{z^n}{n!}$ — соответствующая экспоненциальная производящая функция. В [10] для числа $\bar{S}(n,k)$ помеченных связных графов без мостов с $n$ вершинами и цикломатическим числом $k$ при $k \geqslant 1$ доказана формула

$$\bar{S}(n,k) = \frac{(n-1)!}{nk!}[z^{-1}]Y_k(nB_1'(z), \ldots, nk!B_k'(z))z^{-n}. \tag{2}$$

Здесь $[z^{-1}]$ — оператор формального вычета [11, с. 25]; а $Y_k(x_1, \ldots, x_k)$ — многочлены разбиений (многочлены Белла), для которых известна формула [12, с. 173]

$$Y_m(x_1, \ldots, x_m) = \sum_{\pi(m)} \frac{m!}{k_1! \ldots k_m!}\left(\frac{x_1}{1!}\right)^{k_1} \ldots \left(\frac{x_m}{m!}\right)^{k_m},$$

где суммирование проводится по всем разбиениям $\pi(m)$ числа $m$:

$$k_1 + 2k_2 + \ldots + mk_m = m, \quad k_i \geqslant 0, \quad i = 1, \ldots, m.$$

Будем считать теперь, что числа $\bar{S}(n,k)$ и функции $B_k(z)$ относятся к классу помеченных эйлеровых графов. Формула (2) может быть неверна для подкласса связных графов [13]. Класс графов называется *блочно-устойчивым*, если граф принадлежит

этому классу тогда и только тогда, когда каждый блок графа принадлежит этому классу [14]. Для блочно-устойчивого класса графов формула (2) верна [13]. Известно, что класс эйлеровых графов является блочно-устойчивым [15].

Так как [9, с. 246]

$$Y_5(x_1,x_2,x_3,x_4,x_5) = x_1^5 + 10x_2x_1^3 + 10x_3x_1^2 + 15x_2^2x_1 + 5x_4x_1 + 10x_3x_2 + x_5 = \\ = x_1^5 + 10x_3x_1^2 + 5x_4x_1 + x_5$$

и $x_i = ni!B_i'(z)$, имеем

$$E_5(n) = \frac{(n-1)!}{120}[z^{-1}]\Big(n^4(B_1'(z))^5 + 60n^2B_3'(z)(B_1'(z))^2 + 120nB_1'(z)B_4'(z)(z) + 120B_5'(z))\Big)z^{-n}\ .$$

Интегрируя по частям последнее слагаемое, найдём

$$E_5(n) = B(n,5) + \frac{n!}{120}[z^{-1}]\Big(n^3(B_1'(z))^5 + 60nB_3'(z)(B_1'(z))^2 + 120B_1'(z)B_4'(z)(z)\Big)z^{-n}\ . \quad (3)$$

В [9] получена формула

$$B(n,5) = \frac{n!(n-3)(n-5)}{29030400}(265n^5 + 2680n^4 - 17157n^3 - 82816n^2 + 335252n + 351456).$$

Так как эйлеров унициклический блок — это простой цикл, то $B(n,1) = \dfrac{(n-1)!}{2}$. В [7] доказано, что $B(n,2) = 0$, $B(n,3) = \dfrac{n!}{288}(n-3)(n-4)(n+7)$, а в [8] найдена формула

$$B(n,4) = \frac{n!}{5760}(n-2)(n-4)(n-5)(n^2+11n+18).$$

Таким образом, имеем

$$B_1(z) = \sum_{n=3}^{\infty}\frac{1}{2}(n-1)!\frac{z^n}{n!}, \quad B_1'(z) = \frac{z^2}{2(1-z)}, \quad B_2'(z) = 0,$$

$$B_3(z) = \sum_{n=5}^{\infty}\frac{n!}{288}(n-3)(n-4)(n+7)\frac{z^n}{n!} = \frac{z^5(4-3z)}{48(1-z)^4}, \quad B_3'(z) = \frac{3z^6-11z^5+10z^4}{24(1-z)^5},$$

$$B_4(z) = \sum_{n=6}^{\infty}B(n,4)\frac{z^n}{n!} = \frac{8z^6-12z^7+6z^8-z^9}{48(1-z)^6}, \quad B_4'(z) = \frac{z^9-7z^8+20z^7-28z^6+16z^5}{16(1-z)^7}.$$

Суммирование рядов для $B_3(z)$ и $B_4(z)$, а также дифференцирование выполнено с помощью пакета программ Maple.

Представляя выражение (3) в виде

$$E_5(n) = B(n,5) + P(n) + Q(n) + R(n), \quad (4)$$

с помощью известного разложения $(1-z)^{-p-1} = \sum\limits_{i=0}^{\infty}\binom{i+p}{p}z^i$ ([12, с. 141]) получим

$$P(n) = \frac{n!}{120}n^3[z^{-1}](B_1'(z))^5z^{-n} = \frac{n!n^3}{120}[z^{-1}]\frac{z^{10}}{32(1-z)^5}z^{-n} =$$

$$= \frac{n!n^3}{3840}[z^{-1}]\sum_{i=0}^{\infty}\binom{i+4}{4}z^{i+10-n} = \frac{n!n^3}{3840}\binom{n-7}{4} = \frac{n!n^3}{92160}(n-7)(n-8)(n-9)(n-10);$$

$$Q(n) = \frac{n!}{120}[z^{-1}]60nB_3'(z)(B_1'(z))^2 z^{-n} = \frac{n!n}{192}[z^{-1}]\frac{3z^{10} - 11z^9 + 10z^8}{(1-z)^9}z^{-n} =$$
$$= \frac{n!n}{192}[z^{-1}]\sum_{i=0}^{\infty}\left(3\binom{i+6}{6}z^{i+10-n} - 11\binom{i+6}{6}z^{i+9-n} + 10\binom{i+6}{6}z^{i+8-n}\right) =$$
$$= \frac{n!n}{192}\left(3\binom{n-5}{6} - 11\binom{n-4}{6} + 10\binom{n-3}{6}\right) =$$
$$= \frac{n!n}{69120}(n-5)(n-6)(n-7)(n-8)(n^2+8n-3);$$

$$R(n) = \frac{n!}{120}[z^{-1}]120B_1'(z)B_4'(z)(z)z^{-n} = \frac{n!}{32}[z^{-1}]\frac{z^{11} - 7z^{10} + 20z^9 - 28z^8 + 16z^7}{(1-z)^8}z^{-n} =$$
$$= \frac{n!}{32}[z^{-1}]\sum_{i=0}^{\infty}\left(\binom{i+7}{7}z^{i+11-n} - 7\binom{i+7}{7}z^{i+10-n} + 20\binom{i+7}{7}z^{i+9-n} -\right.$$
$$\left. -28\binom{i+7}{7}z^{i+8-n} + 16\binom{i+7}{7}z^{i+7-n}\right) = \frac{n!}{32}\left(\binom{n-5}{7} - 7\binom{n-4}{7} +\right.$$
$$\left. +20\binom{n-3}{7} - 28\binom{n-2}{7} + 16\binom{n-1}{7}\right) =$$
$$= \frac{n!}{161280}(n-5)(n-6)(n-7)(2n^4 + 15n^3 - 35n^2 - 30n + 48).$$

Подставляя в (4) выражения для $B(n,5), P(n), Q(n), R(n)$, получим формулу (1). ■

В таблице представлены числа $E_5(n)$, вычисленные с помощью теоремы 1 и пакета программ Maple. При $n = 6$ пентациклический граф является блоком, поэтому $E_5(6) = B(6,5) = 90$.

| $n$ | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|
| $E_5(n)$ | 90 | 5061 | 199528 | 6519492 | 191434320 | 5317946855 | 143972662680 |

## 2. Асимптотика и вероятность

Из теоремы 1 сразу получаем следствие 1.

**Следствие 1.** При $n \to \infty$ верно асимптотическое равенство

$$E_5(n) \sim \frac{17n^7}{362880}n! \,.$$

Зададим на множестве помеченных эйлеровых графов с $n$ вершинами равномерное распределение вероятностей.

**Следствие 2.** Пусть $P_n$ — вероятность того, что помеченный эйлеров пентациклический граф с $n$ вершинами является блоком. При $n \to \infty$ верна асимптотика $P_n \sim \dfrac{53}{272}$ .

***Доказательство.*** $P_n = \dfrac{B(n,5)}{E_5(n)} \sim \dfrac{265}{29030400}\dfrac{362880}{17} = \dfrac{53}{272}$ при $n \to \infty$. ■

*Кактусом* называется связный граф, в котором нет рёбер, лежащих более чем на одном простом цикле [1, с. 93]. Все блоки кактуса — рёбра или простые циклы.

**Следствие 3.** Пусть $\bar{P}_n$ — вероятность того, что помеченный эйлеров пентациклический граф с $n$ вершинами является кактусом. При $n \to \infty$ верна асимптотика $\bar{P}_n \sim \dfrac{63}{272}$ .

***Доказательство.*** Пусть $Ca(n,k)$ — число помеченных эйлеровых $k$-циклических кактусов с $n$ вершинами. В [10] получена формула

$$Ca(n,k) = \frac{(n-1)!n^{k-1}}{k!2^k}\binom{n-k-1}{k-1},$$

из которой имеем $Ca(n,5) = \dfrac{n^3 n!}{3840}\binom{n-7}{4} = \dfrac{n!n^3}{92160}(n-7)(n-8)(n-9)(n-10) \sim \dfrac{n!n^7}{92160}$,
$\bar{P}_n = \dfrac{Ca(n,5)}{E_5(n)} \sim \dfrac{n!n^7}{92160}\dfrac{362880}{17n!n^7} = \dfrac{63}{272}$ при $n \to \infty$. ■

ЛИТЕРАТУРА


1. *Харари Ф., Палмер Э.* Перечисление графов. М.: Мир, 1977. 324 с.
2. *Kumar A.* A study on Euler graph and it's applications // Int. J. Math. Trends Technol. 2017. V. 43. No. 1. P. 9–14.
3. *Pancoska P., Moravek Z., and Moll U. M.* Rational design of DNA sequences for nanotechnology, microarrays and molecular computers using Eulerian graphs // Nucleic Acids Res. 2004. V. 32. No. 15. P. 4630–4645.
4. *Amudha P., Sagayaraj A. C. C., and Sheela A. C. S.* An application of graph theory in cryptography // Int. J. Pure Appl. Math. 2018. V. 119. No. 13. P. 375–383.
5. *Read R. C.* Euler graphs on labelled nodes // Canad. J. Math. 1962. V. 14. P. 482–486.
6. *Tazawa S.* Enumeration of labelled 2-connected Euler graphs // J. Combinat. Inform. System Sci. 1998. V. 23. No. 1–4. P. 407–414.
7. *Воблый В. А.* Перечисление помеченных связных бициклических и трициклических эйлеровых графов // Матем. заметки. 2012. Т. 92. № 5. С. 678–683.
8. *Воблый В. А., Мелешко А. К.* Перечисление помеченных эйлеровых тетрациклических графов //Дискретн. анализ и исслед. операций. 2014. Т. 21. № 5. С. 17–22.
9. *Воблый В. А.* О числе помеченных эйлеровых пентациклических блоков //Материалы заочного семинара XIX Международ. конф. «Проблемы теоретической кибернетики». Казань, 2020. С. 41–44.
10. *Воблый В. А.* О перечислении помеченных связных графов с заданными числами вершин и ребер //Дискретн. анализ и исслед. операций. 2016. Т. 3. № 2. С. 5–20.
11. *Гульден Я., Джексон Д.* Перечислительная комбинаторика. М.: Наука, 1990. 504 с.
12. *Риордан Дж.* Комбинаторные тождества. М.: Наука, 1981. 256 с.
13. *Воблый В. А.* Второе соотношение Риддела и следствия из него // Дискретн. анализ и исслед. операций. 2019. Т. 26. № 1. С. 20–32.
14. *Mc Diarmid C. and Scott A.* A random graphs from a block stable class. II //Europ. J. Combin. 2016. V. 58. P. 96–106.
15. *Воблый В. А., Мелешко А. К.* Перечисление помеченных полноблочно-кактусных графов // Дискретн. анализ и исслед. операций. 2014. Т. 21. № 2. С. 24–32.


REFERENCES


1. *Harary F. and Palmer E. M.* Graphical Enumeration. N.Y., London, Academic Press, 1973.
2. *Kumar A.* A study on Euler graph and it's applications. Int. J. Math. Trends Technol., 2017, vol. 43, no. 1, pp. 9–14.

3. *Pancoska P., Moravek Z., and Moll U. M.* Rational design of DNA sequences for nanotechnology, microarrays and molecular computers using Eulerian graphs. Nucleic Acids Res., 2004, vol. 32, no. 15, pp. 4630–4645.
4. *Amudha P., Sagayaraj A. C. C., and Sheela A. C. S.* An application of graph theory in cryptography. Int. J. Pure Appl. Math., 2018, vol. 119, no. 13, pp. 375–383.
5. *Read R. C.* Euler graphs on labelled nodes. Canad. J. Math., 1962, vol. 14, pp. 482–486.
6. *Tazawa S.* Enumeration of labelled 2-connected Euler graphs. J. Combinat. Inform. System Sci., 1998, vol. 23, no. 1–4, pp. 407–414.
7. *Voblyi V. A.* Perechislenie pomechennyh svjaznyh biciklicheskih i triciklicheskih jejlerovyh grafov [Enumeration of labeled connected bicyclic and tricyclic Eulerian graphs]. Matematicheskie Zametki, 2012, vol. 92, no. 5, pp. 678–683. (in Russian)
8. *Voblyi V. A. and Meleshko A. K.* Perechislenie pomechennyh jejlerovyh tetraciklicheskih grafov [Enumeration of labeled Eulerian tetracyclic graphs]. Diskretn. Anal. Issled. Oper., 2014, vol. 21, no. 5, pp. 17–22. (in Russian)
9. *Voblyi V. A.* O chisle pomechennyh jejlerovyh pentaciklicheskih blokov [On the number of labeled Eulerian pentacyclic blocks]. Proc. Intern. Conf. "Problemy Teoreticheskoj Kibernetiki", Kazan', 2020, pp. 41–44. (in Russian)
10. *Voblyi V. A.* Enumeration of labeled connected graphs with given order and size. J. Appl. Ind. Math., 2016, vol. 10, no. 2, pp. 302–310.
11. *Goulden I. P. and Jackson D. M.* Combinatorial Enumeration. Dover Publ., 2004.
12. *Riordan J.* Combinatorial Identities. New York, Wiley, 1968.
13. *Voblyi V. A.* The second Riddell relation and its consequences. J. Appl. Ind. Math., 2019, vol. 13, no. 1, pp. 168–174.
14. *Mc Diarmid C. and Scott A.* A random graphs from a block stable class. II. Europ. J. Combin., 2016, vol. 58, pp. 96–106.
15. *Voblyi V. A. and Meleshko A. K.* The number of labeled block-cactus graphs. J. Appl. Industr. Math., 2014, vol. 8, no. 3, pp. 422–427.

UDC 519.17 DOI 10.17223/20710410/50/7

# THE CHROMATICITY OF THE JOIN OF TREE AND NULL GRAPH

L. X. Hung

*Hanoi University for Natural Resources and Environment, Hanoi, Vietnam*

**E-mail:** lxhung@hunre.edu.vnl

The chromaticity of the graph $G$, which is join of the tree $T_p$ and the null graph $O_q$, is studied. We prove that $G$ is chromatically unique if and only if $1 \leqslant p \leqslant 3$, $1 \leqslant q \leqslant 2$; a graph $H$ and $T_p + O_{p-1}$ are $\chi$-equivalent if and only if $H = T'_p + O_{p-1}$, where $T'_p$ is a tree of order $p$; $H$ and $T_p + O_p$ are $\chi$-equivalent if and only if $H \in \{T'_p + O_p, T''_{p+1} + O_{p-1}\}$, where $T'_p$ is a tree of order $p$, $T''_{p+1}$ is a tree of order $p+1$. We also prove that if $p \leqslant q$, then $\chi'(G) = ch'(G) = \Delta(G)$; if $\Delta(G) = |V(G)| - 1$, then $\chi'(G) = ch'(G) = \Delta(G)$ if and only if $G \neq K_3$.

**Keywords:** *chromatic number, chromatically equivalent, chromatically unique graph, chromatic index, list-chromatic index.*

## 1. Introduction

All graphs considered in the paper are finite undirected graphs without loops or multiple edges. If $G$ is a graph, then $V(G)$, $E(G)$ (or $V$ and $E$ in short) and $\overline{G}$ denote its vertex set, edge set and its complementary graph, respectively. The set of all neighbours of a subset $S \subseteq V(G)$ is denoted by $N_G(S)$ (or $N(S)$ in short). If $S = \{v\}$, then $N(S)$ is denoted by $N(v)$. For a vertex $v \in V(G)$, the degree of $v$ is denoted by $\deg_G(v)$ (or $\deg(v)$), it equals $|N_G(v)|$. The subgraph of $G$ induced by $W \subseteq V(G)$ is denoted by $G[W]$. Let $R$ be a subset of edges in $G$, $|R| = r$; denote by $G - R$ the graph obtained by deleting all edges in $R$ from $G$.

The null graphs and complete graphs of order $n$ are denoted by $O_n$ and $K_n$, respectively. The $K_3$ is called a *triangle*. Let $t_1(G)$, $t_2(G)$, and $t_3(G)$ be the numbers of triangles, of induced subgraphs $C_4$, and of complete subgraphs $K_4$ in $G$, respectively. Unless otherwise indicated, our graph-theoretic terminology follows [1].

An *acyclic* graph, one not containing any cycles, is called *forest*. A connected forest is called a *tree*, a tree of order $n$ is denoted by $T_n$.

A graph $G = (V, E)$ is called *r-partite graph* if $V$ admits a partition into $r$ classes $V = V_1 \cup V_2 \cup \ldots \cup V_r$ such that the subgraphs of $G$ induced by $V_i$, $i = 1, \ldots, r$, are empty. If $r = 2$, then $G$ is called *bipartite graph*, if $r = 3$, then $G$ is called *tripartite graph*. An $r$-partite graph in which every two vertices from different partition classes are adjacent is called *complete r-partite graph* and is denoted by $K_{|V_1|,|V_2|,\ldots,|V_r|}$. The complete $r$-partite graph $K_{|V_1|,|V_2|,\ldots,|V_r|}$ with $|V_1| = |V_2| = \ldots = |V_r| = s$ is denoted by $K_s^r$.

Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be two graphs such that $V_1 \cap V_2 = \varnothing$. Their *union* $G = G_1 \cup G_2$ has, as expected, $V(G) = V_1 \cup V_2$ and $E(G) = E_1 \cup E_2$. Their *join* is denoted $G_1 + G_2$ and consists of $G_1 \cup G_2$ and all edges joining $V_1$ with $V_2$.

Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be two graphs. We call $G_1$ and $G_2$ *isomorphic*, and write $G_1 \cong G_2$, if there exists a bijection $f : V_1 \to V_2$ with $uv \in E_1$ if and only if $f(u)f(v) \in E_2$ for all $u, v \in V_1$.

Let $G = (V, E)$ be a graph and $\lambda$ is a positive integer.

A $\lambda$-*coloring* of $G$ is a bijection $f : V(G) \to \{1, 2, \ldots, \lambda\}$ such that $f(u) \neq f(v)$ for any adjacent vertices $u, v \in V(G)$. The smallest positive integer $\lambda$ such that $G$ has a $\lambda$-coloring is called the *chromatic number* of $G$ and is denoted by $\chi(G)$. We say that a graph $G$ is $n$-*chromatic* if $n = \chi(G)$.

Let $V(G) = \{v_1, v_2, \ldots, v_n\}$, two $\lambda$-colorings $f$ and $g$ are considered different if and only if $f(v_k) \neq g(v_k)$ for some $k \in \{1, 2, \ldots, n\}$. Let $P(G, \lambda)$ (or simply $P(G)$ if there is no danger of confusion) denote the number of distinct $\lambda$-colorings of $G$. It is well-known that for any graph $G$, $P(G, \lambda)$ is a polynomial in $\lambda$, called the *chromatic polynomial* of $G$. The notion of chromatic polynomials was first introduced by Birkhoff [2] in 1912 as a quantitative approach to tackle the four-color problem. Two graphs $G$ and $H$ are called *chromatically equivalent* (or, in short, $\chi$-*equivalent*), and we write $G \sim H$, if $P(G, \lambda) = P(H, \lambda)$. A graph $G$ is called *chromatically unique* ($\chi$-*unique*) if $G' \cong G$ (i.e., $G'$ is isomorphic to $G$) for any graph $G'$ such that $G' \sim G$. For examples, all cycles are $\chi$-unique [3]. The notion of $\chi$-unique graphs was first introduced and studied by Chao and Whitehead [4] in 1978. The readers can see the surveys [3, 5, 6] for more information on $\chi$-unique graphs.

An edge coloring of a graph $G$ can be defined similarly. Namely, an *edge* $\lambda$-*coloring* of a graph $G$ is a mapping $f : E(G) \to \{1, 2, \ldots \lambda\}$ such that two adjacent edges have distinct images. The *chromatic index* of $G$, denoted by $\chi'(G)$, is the smallest positive integer $\lambda$ such that $G$ has an edge $\lambda$-coloring. In 1964, Vizing [7] proved that $\chi'(G)$ is equal to either $\Delta(G)$ or $\Delta(G) + 1$, where $\Delta(G)$ is the maximum degree of $G$. A graph $G$ is said to be *Class one* (resp., *Class two*) if $\chi'(G) = \Delta(G)$ (resp., $\Delta(G) + 1$). For examples, all cycles $C_n$ with $n$ even are Class one; all cycles $C_n$ with $n$ odd are Class two. Let $(L(e))_{e \in E(G)}$ be a family of sets. We call an edge coloring $f$ of $G$ with $f(e) \in L(e)$ for all $e \in E(G)$ a *list edge coloring* from the lists $L(e)$. The least integer $k$ such that $G$ has an edge coloring from any family of lists of size $k$ is the *list-chromatic index* of $G$ and is denoted by $ch'(G)$. The idea of list colorings of graphs is due independently to V. G. Vizing [8] and to P. Erdös, A. L. Rubin, and H. Taylor [9].

In [10] we have characterized chromatically uniqueness of the graph $K_2^r + O_k$, in [11] we have characterized chromatically uniqueness of the graph $G = K_2^n + K_r$, and in [6] we have determined chromatic index and characterized chromatically uniqueness split graphs.

In this paper, we study the chromaticity of $G$, which is join of the tree $T_p$ and the null graph $O_q$. We prove that $G$ is chromatically unique if and only if $1 \leqslant p \leqslant 3$, $1 \leqslant q \leqslant 2$; $H$ and $T_p + O_{p-1}$ are $\chi$-equivalent if and only if $H = T'_p + O_{p-1}$, where $T'_p$ is a tree of order $p$; $H$ and $T_p + O_p$ are $\chi$-equivalent if and only if $H \in \{T'_p + O_p, T''_{p+1} + O_{p-1}\}$, where $T'_p$ is a tree of order $p$, $T''_{p+1}$ is a tree of order $p + 1$. We also prove that if $p \leqslant q$, then $\chi'(G) = ch'(G) = \Delta(G)$; if $\Delta(G) = |V(G)| - 1$, then $\chi'(G) = ch'(G) = \Delta(G)$ if and only if $G \neq K_3$.

## 2. Vertex colorings

For a graph $G$ and a positive integer $k$, a partition $\{A_1, A_2, \ldots, A_k\}$ of $V(G)$ is called *a* $k$-*independent partition in* $G$ if each $A_i$ is a non-empty independent set of $G$. Let $\alpha(G, k)$ denote the number of $k$-independent partitions in $G$. Hence, $P(G, \lambda) = \sum\limits_{1 \leqslant k \leqslant n} \alpha(G, k)(\lambda)_k$, where $(\lambda)_k = \lambda(\lambda - 1) \ldots (\lambda - k + 1)$.

The polynomial $\sigma(G, x) = \sum\limits_{1 \leqslant k \leqslant n} \alpha(G, k)x^k$ is called *the* $\sigma$-*polynomial of* $G$.

The polynomial $h(G, x) = \sum\limits_{1 \leqslant k \leqslant n} \alpha(\overline{G}, k)x^k$ is called *the adjoint polynomial of* $G$.

Let $K_p^+$ be the vertex gluing of $K_p$ and $K_2$.

For convenience, denote $\sigma(G,x)$ by $\sigma(G)$, $h(G,x)$ by $h(G)$, and $G \cong H$ by $G = H$. The following lemmas will be used to prove our main results.

**Lemma 1** [3]. If $G = K_n$ is the complete graph on $n$ vertices, then $\chi(G) = n$ and $G$ is $\chi$-unique.

**Lemma 2.** If $G = K_{n_1,n_2,\ldots,n_r}$ is the complete $r$-partite graph, then $\chi(G) = r$.

**Lemma 3** [12]. Let $G$ and $H$ be two $\chi$-equivalent graphs. Then

(i) $|V(G)| = |V(H)|$;
(ii) $|E(G)| = |E(H)|$;
(iii) $\chi(G) = \chi(H)$;
(iv) $G$ is connected if and only if $H$ is connected;
(v) $G$ is 2-connected if and only if $H$ is 2-connected;
(vi) $t_1(G) = t_1(H)$;
(vii) $t_2(G) - 2t_3(G) = t_2(H) - 2t_3(H)$;
(viii) $\alpha(G,k) = \alpha(H,k)$ for each $k = 1, 2, \ldots$

**Lemma 4** [12].

(i) All trees of the same order are $\chi$-equivalent. Further, the graph $G$ of order $n$ is a tree if and only if $P(G,\lambda) = \lambda(\lambda-1)^{n-1}$;
(ii) A tree $T_n$ is $\chi$-unique if and only if $1 \leqslant n \leqslant 3$;
(iii) If $G = T_n$ is a tree of order $n$, then $\chi(G) = 2$.

**Lemma 5** [11]. The graph $G = K_2^m + K_n$ is $\chi$-unique.

**Lemma 6** [13]. Let $G$ and $H$ be two disjoint graphs. Then

(i) $\sigma(G+H,x) = \sigma(G,x)\sigma(H,x)$;
(ii) $h(G \cup H,x) = h(G,x)h(H,x)$.

**Lemma 7** [14]. Let $G$ and $H$ be two graphs. Then

(i) $P(G,\lambda) = P(H,\lambda)$ if and only if $\sigma(G,x) = \sigma(H,x)$;
(ii) $P(G,\lambda) = P(H,\lambda)$ if and only if $h(\overline{G},x) = h(\overline{H},x)$.

**Lemma 8.** If $p \geqslant 2$, then $\chi(T_p + O_q) = 3$.

***Proof.*** If $p \geqslant 2$, then the complete graph $K_3$ is a subgraph of $G = T_p + O_q$. So $\chi(G) \geqslant 3$. Let $V(G) = V_1 \cup V_2$ is a partition of $V(G)$ such that $G[V_1] = T_p$, $G[V_2] = O_q$. The graph $G[V_1]$ is a tree, by (iii) of Lemma 4, $G[V_1]$ has a coloring $f_1$ using two colors $1, 2$. Set mapping

$$f : V(G) \to \{1, 2, 3\}$$

such that $f(v) = f_1(v)$ if $v \in V_1$, $f(v) = 3$ if $v \in V_2$. Then $f$ is a 3-coloring of $G$, i.e., $\chi(G) \leqslant 3$. Thus, $\chi(G) = 3$. ■

**Theorem 1.** $G = T_p + O_q$ is $\chi$-unique if and only if $1 \leqslant p \leqslant 3$, $1 \leqslant q \leqslant 2$.

***Proof.*** First we prove the necessity. Suppose that $G = T_p + O_q$ is $\chi$-unique. Suppose the contrary, that $p \geqslant 4$. Set $G^1 = (K^1 \cup I^1, E^1)$ with

$$K^1 = \{v_1, v_2, \ldots, v_p\}, \quad I^1 = \{u_1, u_2, \ldots, u_q\},$$
$$E^1 = \{v_1v_2, v_1v_3, \ldots, v_1v_p\} \cup \{v_iu_j : i = 1, 2, \ldots p, j = 1, 2, \ldots q\}.$$

Set $G^2 = (K^2 \cup I^2, E^2)$ with

$$K^2 = \{v_1, v_2, \ldots, v_p\}, \quad I^2 = \{u_1, u_2, \ldots, u_q\},$$
$$E^2 = \{v_1v_2, v_2v_3, \ldots, v_{p-1}v_p\} \cup \{v_iu_j : i = 1, 2, \ldots p, j = 1, 2, \ldots q\}.$$

By (i) of Lemma 4, (i) of Lemma 6, and (i) of Lemma 7, it follows that

$$P(G^1,\lambda) = P(G^2,\lambda) = P(G,\lambda).$$

It is not difficult to see that

$$\Delta(G^1) = \max\{\deg(u) : u \in V(G^1)\} = \deg(v_1) = p+q-1$$

and

$$\Delta(G^2) = \max\{\deg(u) : u \in V(G^2)\} = \max\{p, q+2\}.$$

If $q \geqslant 2$, then $\max\{p, q+2\} < p+q-1$, it follows that $\Delta(G^2) < \Delta(G^1)$. So $G^1 \ncong G^2$ and $G$ is not $\chi$-unique, a contradiction.

If $q = 1$, then $\Delta(G^2) = \Delta(G^1) = p$. It is not difficult to see that

$$|\{u \in V(G^1) : \deg_{G^1}(u) = p\}| = 2 \quad \text{and} \quad |\{u \in V(G^2) : \deg_{G^2}(u) = p\}| = 1.$$

It follows that $G^1 \ncong G^2$ and $G$ is not $\chi$-unique, a contradiction. Thus, $1 \leqslant p \leqslant 3$.

Suppose that $q \geqslant 3$. For $p = 3$, we set $G^3 = (K^3 \cup I^3, E^3)$ with

$$K^3 = \{v_1, v_2, v_3\}, \quad I^3 = \{u_1, u_2, \ldots, u_q\},$$
$$E^3 = \{v_1v_2, v_2v_3\} \cup \{v_iu_j : i = 1,2,3, j = 1,2,\ldots q\},$$

and set $G^4 = (K^4 \cup I^4, E^4)$ with

$$K^4 = \{v_1, v_2, v_3\}, \quad I^4 = \{u_1, u_2, \ldots, u_q\},$$
$$E^4 = \{v_2u_1, u_1u_2, u_2u_3, \ldots, u_{q-1}u_q\} \cup \{v_1v_2, v_2v_3\} \cup \{v_1u_j, v_3u_j : j = 1,2,\ldots,q\}.$$

It is not difficult to see that $G = G^3 = T_3 + O_q$ and $G^4 = T_{q+1} + O_2$. By (i) of Lemma 4, (i) of Lemma 6, and (i) of Lemma 7, we have

$$\begin{aligned}
\sigma(G^3,x) &= \sigma(T_3 + O_q, x) = \\
&= \sigma(T_3,x)\sigma(O_q,x) = \\
&= \sigma(O_1 + O_2,x)\sigma(O_q,x) = \\
&= \sigma(O_1,x)\sigma(O_2,x)\sigma(O_q,x) = \\
&= \sigma(O_1,x)\sigma(O_q,x)\sigma(O_2,x) = \\
&= \sigma(O_1 + O_q,x)\sigma(O_2,x) = \\
&= \sigma(T_{q+1},x)\sigma(O_2,x) = \\
&= \sigma(T_{q+1} + O_2,x) = \\
&= \sigma(G^4,x).
\end{aligned}$$

It follows that $P(G^3,\lambda) = P(G^4,\lambda)$. Otherwise,

$$\Delta(G^3) = \max\{\deg(u) : u \in V(G^3)\} = \deg(v_2) = q+2,$$
$$\Delta(G^4) = \max\{\deg(u) : u \in V(G^2)\} = \deg(v_1) = \deg(v_3) = q+1.$$

So $G^3 \ncong G^4$ and $G$ is not $\chi$-unique, a contradiction.

For $p = 2$, we set $G^5 = (K^5 \cup I^5, E^5)$ with

$$K^5 = \{v_1, v_2\}, \quad I^5 = \{u_1, u_2, \ldots, u_q\},$$
$$E^5 = \{v_1v_2\} \cup \{v_iu_j : i = 1,2, j = 1,2,\ldots q\},$$

and set $G^6 = (K^6 \cup I^6, E^6)$ with

$$K^6 = \{v_1, v_2\}, \quad I^6 = \{u_1, u_2, \ldots, u_q\},$$
$$E^6 = \{v_2u_1, u_1u_2, u_2u_3, \ldots, u_{q-1}u_q\} \cup \{v_1v_2\} \cup \{v_1u_j : j = 1, 2, \ldots q\}.$$

It is clear that $P(G^5, \lambda) = P(G^6, \lambda)$ and

$$|\{u \in V(G^5) : \deg_{G^5}(u) = q + 1\}| = |\{v_1, v_2\}| = 2,$$
$$|\{u \in V(G^6) : \deg_{G^6}(u) = q + 1\}| = |\{v_1\}| = 1.$$

So $G^5 \ncong G^6$ and $G$ is not $\chi$-unique, a contradiction.

If $p = 1$, then $G$ is a tree $T_n$ with $n = q+1 \geqslant 4$. By (ii) of Lemma 4, $G$ is not $\chi$-unique, a contradiction.

Now we prove the sufficiency. If $p = 1$ and $q = 1$, then $G = K_2$, if $p = 2$ and $q = 1$, then $G = K_3$. By Lemma 1, $G$ is $\chi$-unique.

If $p = 1$ and $q = 2$, then $G = T_3$. By (ii) of Lemma 4, $G$ is $\chi$-unique.

If $p = 2$ and $q = 2$ or $p = 3$ and $q = 1$, then $G = K_2^1 + K_2$, if $p = 3$ and $q = 2$, then $G = K_2^2 + K_1$. By Lemma 5, $G$ is $\chi$-unique. ■

**Theorem 2.**

(i) $H$ and $T_p + O_{p-1}$ are $\chi$-equivalent if and only if $H = T'_p + O_{p-1}$, where $T'_p$ is a tree of order $p$;

(ii) $H$ and $T_p + O_p$ are $\chi$-equivalent if and only if $H \in \{T'_p + O_p, T''_{p+1} + O_{p-1}\}$, where $T'_p$ is a tree of order $p$, $T''_{p+1}$ is a tree of order $p + 1$.

***Proof.*** If $p = 2$, then, by Theorem 1, $T_p + O_{p-1}$ and $T_p + O_p$ are $\chi$-unique. It follows that the theorem is obviously true. Hence we may assume that $p \geqslant 3$.

Suppose that $H$ and $G = T_p + O_q$ are $\chi$-equivalent, where $p - 1 \leqslant q \leqslant p$. By (iii) of Lemma 3 and Lemma 8, $\chi(H) = 3$. So $H$ is a tripartite graph. We may assume that $D = K_{a,b,c}$ and $R = \{e_1, e_2, \ldots, e_r\} \subseteq E(D)$ such that $H = D - R$ and $a \leqslant b \leqslant c$. It is clear that

$$r = |E(D) - |E(H)| = |E(D)| - |E(G)| = ab + ac + bc - pq - p + 1$$

and

$$a + b + c = |V(D)| = |V(H)| = |V(G)| = p + q.$$

Denote by $t_1(e_i)$ the number of triangles containing the edge $e_i$ in $D$ for every $i = 1, 2, \ldots, r$. It is not difficult to see that $t_1(e_i) \leqslant c$ for every $i = 1, 2, \ldots, r$. Then

$$t_1(H) \geqslant t_1(D) - rc,$$

and the equality holds only if $t_1(e_i) = c$ for every $i = 1, 2, \ldots, r$.

By (vi) of Lemma 3, $t_1(G) = t_1(H)$, it follows that

$$t_1(D) - t_1(G) = t_1(D) - t_1(H) \leqslant rc.$$

Since $t_1(D) = abc, t_1(G) = (p - 1)q$, we have

$$\begin{aligned} f(c) &= t_1(D) - t_1(G) - rc = \\ &= abc - (p - 1)q - (ab + ac + bc - pq - p + 1)c = \\ &= abc - (p - 1)q - [ab + (p + q - c)c - pq - p + 1]c = \\ &= (c - 1)(c - p + 1)(c - q) \leqslant 0. \end{aligned}$$

By $p \geqslant 3$ and $p-1 \leqslant q \leqslant p$, it follows that $c \geqslant (p+q)/3 > 1$. Let $\{V_1, V_2, V_3\}$ be the 3-independent partition in $H$ such that $|V_1| = a$, $|V_2| = b$, $|V_3| = c$. It is not difficult to see that if $f(c) = 0$, then $t_1(e_i) = c$ for every $i = 1, 2, \ldots, r$, so edge $e_i \in R$ has one end vertex in $V_1$ and another end vertex in $V_2$. It follows that $H = H[V_1 \cup V_2] + O_q$

(i) If $q = p-1$, then $G = T_p + O_{p-1}$. In this case, $f(c) = (c-1)(c-p+1)^2 \geqslant 0$, $f(c) = 0$ if and only if $c = p-1$. So $t_1(e_i) = c = p-1$ for every $i = 1, 2, \ldots, r$. By (i) of Lemma 6 and (i) of Lemma 7, we have

$$\begin{aligned}
\sigma(H,x) &= \sigma(H[V_1 \cup V_2] + O_{p-1}, x) = \\
&= \sigma(H[V_1 \cup V_2], x)\sigma(O_{p-1}, x) = \\
&= \sigma(G, x) = \\
&= \sigma(T_p + O_{p-1}, x) = \\
&= \sigma(T_p, x)\sigma(O_{p-1}, x).
\end{aligned}$$

It follows that $\sigma(H[V_1 \cup V_2], x) = \sigma(T_p, x)$. So $P(H[V_1 \cup V_2], \lambda) = P(T_p, \lambda)$. By (i) of Lemma 4, $H[V_1 \cup V_2] = T'_p$, where $T'_p$ is a tree of order $p$. Thus, $H = T'_p + O_{p-1}$.

It is not difficult to see that if $H = T'_p + O_{p-1}$, then $P(H, \lambda) = P(T_p + O_{p-1}, \lambda)$.

(ii) If $q = p$, then $G = T_p + O_p$. So $f(c) \leqslant 0$ if and only if $p-1 \leqslant c \leqslant p$, $f(z) = 0$ if and only if $c = p-1$ or $c = p$. Now we consider separately two cases.

C a s e 1 : $c = p$. We have

$$\begin{aligned}
\sigma(H,x) &= \sigma(H[V_1 \cup V_2] + O_p, x) = \\
&= \sigma(H[V_1 \cup V_2], x)\sigma(O_p, x) = \\
&= \sigma(G, x) = \\
&= \sigma(T_p + O_p, x) = \\
&= \sigma(T_p, x)\sigma(O_p, x).
\end{aligned}$$

It follows that $\sigma(H[V_1 \cup V_2], x) = \sigma(T_p, x)$. So $P(H[V_1 \cup V_2], \lambda) = P(T_p, \lambda)$. By (i) of Lemma 4, $H[V_1 \cup V_2] = T'_p$, where $T'_p$ is a tree of order $p$. Thus, $H = T'_p + O_p$.

C a s e 2 : $c = p-1$. We have

$$\begin{aligned}
\sigma(H,x) &= \sigma(H[V_1 \cup V_2] + O_{p-1}, x) = \\
&= \sigma(H[V_1 \cup V_2], x)\sigma(O_{p-1}, x) = \\
&= \sigma(G, x) = \\
&= \sigma(T_p + O_p, x) = \\
&= \sigma(T_p, x)\sigma(O_p, x) = \\
&= \sigma(O_1 + O_{p-1}, x)\sigma(O_p, x) = \\
&= \sigma(O_1, x)\sigma(O_{p-1}, x)\sigma(O_p, x) = \\
&= \sigma(O_1, x)\sigma(O_p, x)\sigma(O_{p-1}, x) = \\
&= \sigma(O_1 + O_p, x)\sigma(O_{p-1}, x) = \\
&= \sigma(T_{p+1}, x)\sigma(O_{p-1}, x).
\end{aligned}$$

It follows that $\sigma(H[V_1 \cup V_2], x) = \sigma(T_{p+1}, x)$. So $P(H[V_1 \cup V_2], \lambda) = P(T_{p+1}, \lambda)$. By (i) of Lemma 4, $H[V_1 \cup V_2] = T''_{p+1}$, where $T''_{p+1}$ is a tree of order $p+1$. Thus, $H = T''_{p+1} + O_{p-1}$. It is not difficult to see that if $H \in \{T'_p + O_p, T''_{p+1} + O_{p-1}\}$, then $P(H, \lambda) = P(T_p + O_p, \lambda)$. ■

## 3. Edge colorings

We need the following lemmas 9–13 to prove our results.

**Lemma 9** [15]. Every bipartite graph $G$ satisfies $\chi'(G) = \Delta(G)$.

**Lemma 10** [15]. $ch'(G) \geqslant \chi'(G)$ for all graphs $G$.

**Lemma 11** [15]. Every bipartite graph $G$ satisfies $ch'(G) = \chi'(G)$.

**Lemma 12** [16]. If $G$ is a graph of order $2n+1$ and $\Delta(G) = 2n$, then $G$ is Class one if and only if $|E(\overline{G})| \geqslant n$.

**Lemma 13** [12]. If $G = T_n$ is a tree of order $n$, then $|E(G)| = n - 1$.

**Theorem 3.** If $p \leqslant q$, then graph $G = T_p + O_p$ satisfies

(i) $\chi'(G) = \Delta(G)$;
(ii) $ch'(G) = \Delta(G)$.

***Proof.*** Let $V(G) = V_1 \cup V_2$ is a partition of $V(G)$ such that $G[V_1] = T_p$, $G[V_2] = O_q$, $V_1 = \{v_1, v_2, \ldots, v_p\}$, $V_2 = \{u_1, u_2, \ldots, u_q\}$. Set $G_1 = G[V_1]$ and $G_2 = G - E(G[V_1])$. It is not difficult to see that $G_1$ and $G_2$ are bipartite graphs, $\Delta(G_2) = q = \deg_{G_2}(v)$ for every vertex $v \in V_1$ and $\Delta(G) = \Delta(G_1) + \Delta(G_2) = \Delta(G_1) + q = \deg(v)$ for some vertex $v \in V_1$.

(i) By (iii) of Lemma 4, $\chi(G_1) = 2$, so $G_1$ is a bipartite graph. By Lemma 9, $G_1$ has an edge coloring $f_1$ using $\Delta(G_1)$ colors $1, 2, \ldots, \Delta(G_1)$. Again by Lemma 9, $G_2$ has an edge coloring $f_2$ using $\Delta(G_2) = q$ colors $\Delta(G_1) + 1, \ldots, \Delta(G_1) + q$. Since $\Delta(G) = \deg(v)$ for some vertex $v \in V_1$, it is clear that the mapping

$$f : E(G) \to \{1, 2, \ldots, \Delta(G_1), \Delta(G_1) + 1, \ldots, \Delta(G_1) + q\}$$

such that $f(e) = f_1(e)$ if $e \in E(G_1)$ and $f(e) = f_2(e)$ if $e \in E(G_2)$ is an edge coloring of $G$. Since $\Delta(G) = \Delta(G_1) + q$, it follows that $\chi'(G) = \Delta(G)$.

(ii) By Lemma 10 and (i), we have $ch'(G) \geqslant \Delta(G) = \Delta(G_1) + q$. Now we prove that $ch'(G) \leqslant \Delta(G)$. Let $L(e)$ be the lists of colors of $e \in E(G)$ such that $|L(e)| = \Delta(G)$.

Let $L_1(e) \subseteq L(e)$ such that $|L_1(e)| = \Delta(G_1)$ for every $e \in E(G_1)$. Since $G_1$ is a bipartite graph, by Lemma 9 and Lemma 11, there exists $g_1$ being a list edge coloring of $G_1$ with the lists of colors $L_1(e)$ for every $e \in E(G_1)$.

For every $i = 1, 2, \ldots, p$, the subgraph induced by the edges of $G_1$ incident with $v_i$ is denoted by $G_1(v_i)$. It is clear that $|g_1(G_1(v_i))| \leqslant \Delta(G_1)$. For every $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$, set $L'(v_iu_j) = L(v_iu_j) \backslash g_1(G_1(v_i))$. It follows that $|L'(v_iu_j)| \geqslant \Delta(G) - \Delta(G_1) = q$. Let $L_2(v_iu_j) \subseteq L'(v_iu_j)$ such that $|L_2(v_iu_j)| = \Delta(G_2) = q$ for every $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$. By Lemma 11, there exists $g_2$ being a list edge coloring of $G_2$ with the lists of colors $L_2(v_iu_j)$ for every $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$. Let $g$ be the edge coloring of $G$ such that $g(e) = g_1(e)$ if $e \in E(G_1)$ and $g(e) = g_2(e)$ if $e \in E(G_2)$. Then $g$ is a list edge coloring of $G$ with the lists of colors $L(e)$ for every $e \in E(G)$, i.e., $ch'(G) \leqslant \Delta(G)$. Thus, $ch'(G) = \Delta(G)$. ■

**Theorem 4.** Let $G = T_p + O_p$ be a graph with $\Delta(G) = p + q - 1$. Then

$$\chi'(G) = ch'(G) = \Delta(G)$$

if and only if $G \neq K_3$.

***Proof.*** Let $V(G) = V_1 \cup V_2$ is a partition of $V(G)$ such that $G[V_1] = T_p$, $G[V_2] = O_q$, $V_1 = \{v_1, v_2, \ldots, v_p\}$, $V_2 = \{u_1, u_2, \ldots, u_q\}$. Set $G_1 = G[V_1]$ and $G_2 = G - E(G[V_1])$. It is not difficult to see that $G_1$ and $G_2$ are bipartite graphs and $\Delta(G) = p + q - 1 = \deg(v)$ for some vertex $v \in V_1$. It follows that $\Delta(G_1) = p - 1$.

Suppose that $\chi'(G) = ch'(G) = \Delta(G)$. We have $chi'(K_3) = ch'(K_3) = 3$. So $G \neq K_3$.

Now suppose that $G \neq K_3$. If $p \leqslant q$, then by Theorem 3, $\chi'(G) = ch'(G) = \Delta(G)$. So we may assume that $p > q$. If $p = 2$, then $q = 1$, so $G = K_3$, a contradiction. It follows that $p \geqslant 3$. Without loss of generality we may assume that $\Delta(G_1) = \deg_{G_1}(v_1)$, so $\Delta(G) = \deg(v_1)$. Since $\Delta(G_1) = p-1$, it is not difficult to see that $E(G_1) = \{v_1v_2, v_1v_3, \ldots, v_1v_p\}$. We consider separately two cases.

C a s e 1: $p = q + 1$.

If $q = 1$, then $p = 2$, so $G = K_3$, a contradiction. So we may assume that $q \geqslant 2$. By Lemma 13, it is not difficult to see that $|E(\overline{G})| = q^2 - q$. Since $q \geqslant 2$, it follows that $|E(\overline{G})| \geqslant q$. By Lemma 12, $G$ is Class one.

By Lemma 10, $ch'(G) \geqslant \chi'(G) = \Delta(G)$. Let $L(e)$ be the lists of colors of $e \in E(G)$ such that $|L(e)| = \Delta(G)$. Set $G_3 = G - E(G[\{v_2, v_3, \ldots, v_p\} \cup V_2])$ and $G_4 = G[\{v_2, v_3, \ldots, v_p\} \cup V_2]$. It is clear that $G_3$ and $G_4$ are bipartite graphs with $\Delta(G_3) = \deg(v_1) = \Delta(G)$ and $\Delta(G_4) = q$. By Lemma 9 and Lemma 11, there exists $g_3$ being a list edge coloring of $G_3$ with the lists of colors $L(e)$ for every $e \in E(G_3)$. For every $i = 2, 3, \ldots, p$, $j = 1, 2, \ldots, q$, set $L'(v_iu_j) = L(v_iu_j) \setminus \{g_3(v_1v_i), g_3(v_1u_j)\}$. It follows that $|L'(v_iu_j)| \geqslant \Delta(G) - 2 = p + q - 3 \geqslant q$ for every $i = 2, 3, \ldots, p$, $j = 1, 2, \ldots, q$. Let $L_4(v_iu_j) \subseteq L'(v_iu_j)$ such that $|L_4(v_iu_j)| = q$ for every $i = 2, 3, \ldots, p$, $j = 1, 2, \ldots, q$. By Lemma 11, there exists $g_4$ being a list edge coloring of $G_4$ with the lists of colors $L_2(v_iu_j)$ for every $i = 2, 3, \ldots, p$, $j = 1, 2, \ldots, q$. Let $g$ be the edge coloring of $G$ such that $g(e) = g_3(e)$ if $e \in E(G_3)$ and $g(e) = g_4(e)$ if $e \in E(G_4)$. Then $g$ is a list edge coloring of $G$ with the lists of colors $L(e)$ for every $e \in E(G)$, i.e., $ch'(G) \leqslant \Delta(G)$. Thus, $ch'(G) = \Delta(G)$.

C a s e 2: $p \geqslant q + 2$.

It is clear that $G[v_1 \cup V_2] = T'_{q+1}$, where $T'_{q+1}$ is a tree of order $q + 1$. Therefore, $G = T'_{q+1} + O_{p-1}$. Since $q + 1 < p - 1$, by Theorem 3, $\chi'(G) = ch'(G) = \Delta(G)$. ■

## Conclusion

The coloring problems are interesting topics in graph theory. Coloring graphs found application in many practical problems, for example, coding theory or security. Clearly, to estimate the chromatic as well as the chromatic uniqueness is very important. So far there have been many research results on this topic for different graph layers. However, the problem has not been generally solved, and further research is needed. This paper explores some of the coloring problems with graph $G$, which is join of the tree $T_p$ and the null graph $O_q$, contributes to enriching the research results on the coloring problems.

## REFERENCES

1. *Behzad M. and Chartrand G.* Introduction to the Theory of Graphs. Boston, Allyn and Bacon, 1971.
2. *Birkhoff G. D.* A determinant formula for the number of ways of coloring a map. Ann. Math., 1912, vol. 14 (2), pp. 42–46.
3. *Koh K. M. and Teo K. L.* The search for chromatically unique graphs. Graphs Combin., 1990, no. 6, pp. 259–285.
4. *Chao C. Y. and Whitehead Jr. E. G.* On chromatic equivalence of graphs. Lecture Notes Math., 1978, vol. 642, pp. 121–131.
5. *Koh K. M. and Teo K. L.* The search for chromatically unique graphs II. Discrete Math., 1997, vol. 172, pp. 59–78.
6. *Tan N. D. and Hung L. X.* On colorings of split graphs. Acta Mathematica Vietnammica, 2006, vol. 31, no. 3, pp. 195–204.

7. *Vizing V. G.* Ob otsenke khromaticheskogo klassa $p$-grafa [On an estimate of the chromatic class of a $p$-graph]. Discret. Analiz, 1964, no. 3, pp. 23–30. (in Russian)
8. *Vizing V. G.* Raskraska vershin grafa v predpisannye tsveta [Coloring the vertices of a graph in prescribed colors]. Diskret. Analiz, 1976, no. 29, pp. 3–10. (in Russian)
9. *Erdös P., Rubin A. L., and Taylor H.* Choosability in graphs. Proc. West Coast Conf. Combin., Graph Theory, and Computing, Arcata, CA, September 1979, no. 26, pp. 125–157.
10. *Hung L. X.* List-chromatic number and chromatically unique of the graph $K_2^r + O_k$. Selecciones Matemáticas, Universidad Nacional de Trujillo, 2019, vol. 06(01), pp. 26–30.
11. *Hung L. X.* Colorings of the graph $K_2^m + K_n$. J. Siberian Federal University. Mathematics & Physics, 2020, vol. 13, no. 3, pp. 297–305.
12. *Read R. C.* An introduction to chromatic polynomials. J. Combin. Theory, 1968, no. 4, pp. 52–71.
13. *Brenti F.* Expansions of chromatic polynomial and log-concavity. Trans. Amer. Math. Soc., 1992, vol. 332, pp. 729–756.
14. *Liu R. Y.* A new method to find the chromatic polynomial of a graph and its applications. Kexue Tongbao, 1987, vol. 32, pp. 1508–1509.
15. *Diestel R.* Graph Theory. N.Y., Springer Verlag, 2000.
16. *Plantholt M.* The chromatic index of graphs with a spanning star. J. Graph Theory, 1981, no. 5, pp. 5–13.

# МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 004.056.5

## АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ХЭШ-ФУНКЦИЙ ДЛЯ ОБФУСКАЦИИ ПРОГРАММНОГО КОДА

Р. К. Лебедев

*Новосибирский государственный университет, г. Новосибирск, Россия*

Рассмотрены особенности применения хэш-функций для запутывания программного кода, а также проблемы использования в этих целях существующих хэш-функций. С учётом этих особенностей и проблем предлагается метод автоматической генерации хэш-функций, основанный на подходе генетического программирования. Предложены методы оценки устойчивости хэш-функций к автоматическим атакам поиска первого прообраза, основанным на использовании SMT-решателей, и к случайным коллизиям. Проведена оценка генерируемых функций, а также предложен метод быстрого обнаружения слабых экземпляров, позволяющий значительно повысить устойчивость получаемых хэш-функций к атакам.

**Ключевые слова:** *обфускация, хэш-функция, генетическое программирование, лавинный эффект, SMT-решатель.*

DOI 10.17223/20710410/50/8

## AUTOMATIC GENERATION OF HASH FUNCTIONS FOR PROGRAM CODE OBFUSCATION

R. K. Lebedev

*Novosibirsk State University, Novosibirsk, Russia*

**E-mail:** n0n3m4@gmail.com

The specifics of hash function applications in code obfuscation are considered, as well as the disadvantages of currently existing hash functions for this purpose. Considering these specifics and disadvantages, an automatic hash function generation method is proposed, that is based on a genetic programming approach. Methods of measuring the hash function resilience to automated preimage attacks based on SMT solvers usage and random collision resistance are also proposed. Generated functions were evaluated and a method of weak function detection is proposed, that allows to increase the resilience of generated functions to attacks notably.

**Keywords:** *obfuscation, hash function, genetic programming, avalanche effect, SMT solver.*

## Введение

Обфускация — процесс запутывания кода программ с целью затруднения их анализа. Анализ исполняемого кода программ применяется как потенциальными нарушителями авторского права, так и антивирусными аналитиками, поэтому как методы обфускации, так и методы противодействия ей становятся темой многих научных работ [1, 2]. Одним из наиболее важных свойств обфускации является сохранение функциональности программы: для любых входных данных результат выполнения обфусцированной и оригинальной программы должны совпадать. От обфусцированной программы обычно требуют близкого к оригинальной программе размера и быстродействия, однако конкретные требования могут варьироваться в широких диапазонах в зависимости от задач [3].

Среди применяемых для обфускации подходов используются и основанные на криптографии, в том числе на криптографических хэш-функциях. Например, в [4] представлен подход, использующий свойство односторонности таких хэш-функций для запутывания условных переходов в программе.

Однако широко используемые хэш-функции, такие, как MD5, SHA-1 и SHA-2, зачастую могут быть распознаны в машинном коде по используемым в их реализации константам (например, `0xd76aa478` для MD5). Из-за этого эффективность их использования в обфускации подвергается критике [1]: аналитик сможет узнать функцию и сделать выводы об используемом методе обфускации, даже не разбирая её код. Современные инструменты обратной разработки программ также позволяют распознать популярные функции исключительно по машинному коду, без использования констант [5]. Существуют и методы обнаружения криптографических функций, основанные на динамическом анализе [6].

Проблемой применения распространённых хэш-функций в обфускации является не только тот факт, что атакующий получит представление о сути механизма обфускации, но и то, что переборные атаки на такие хэш-функции могут осуществляться при помощи готовых оптимизированных инструментов с очень высокой производительностью (порядка $10^{10}$ попыток в секунду) [7]. Поскольку многие константы в исходном коде программ являются 32-битными целочисленными значениями или строками, содержащими осмысленные слова, атака полным перебором в ряде случаев будет практически применимой.

Целью данной работы является решение упомянутых проблем путём автоматической генерации большого семейства ранее неизвестных случайных хэш-функций, которые можно использовать в процессе обфускации.

Использование таких хэш-функций призвано обеспечить следующие свойства:

1) Затруднение ручного анализа кода защищённых программ при помощи дизассемблеров и декомпиляторов. Такие функции не будут автоматически распознаны, даже сам факт того, что определённая часть кода является хэш-функцией, аналитику придётся устанавливать вручную.
2) Невозможность использования существующего программного обеспечения для атак полным перебором.
3) Возможность использования множества различных функций при обфускации разных мест программы, что кратно увеличит требуемые усилия по ручному анализу и созданию инструментов для переборных атак.

Среди существующих результатов по автоматической генерации хэш-функций наибольший интерес представляет работа [8]. Некоторые из описанных в ней подходов

нашли применение и в данном исследовании, хотя авторы [8] преследовали другую цель, в результате чего найденная ими хэш-функция gp-hash оказалась не слишком сложной для ручного анализа, а также тривиально обратимой. Продолжением [8] является работа [9], в которой описано создание блочного шифра Wheedham на основе сети Фейстеля при помощи нелинейных функций, подобных найденным в [8]. При помощи конструкции Миагучи — Пренеля из этого шифра, в свою очередь, была создана криптографическая хэш-функция MPW-512. Однако в силу использования готовых конструкций она не является сложной для обнаружения и анализа в исполняемом коде, что делает её малоподходящей для нужд обфускации. Кроме того, [8, 9] имеют целью поиск одной лучшей функции, а не семейства, что в значительной степени отличает их от данной работы.

## 1. Требования к хэш-функциям для обфускации

### 1.1. Механизм обфускации

Рассмотрим хэш-функции, применимые для механизма обфускации, описанного в [4]. Его идея состоит в следующем: значения констант в коде условий заменяются результатами выполнения хэш-функций, благодаря чему восстановление их первоначальных значений становится вычислительно сложной задачей. Пример использования подобного механизма обфускации некоторой функции проверки лицензионного ключа приведён в листингах 1 и 2.

```
1 int check_serial(char* serial) {
2     if (!strcmp(serial, "S0M3S3CR3TK3Y"))
3         return 1;
4     else
5         return 0;
6 }
```

Листинг 1. Функция до обфускации

```
1 int check_serial(char* serial) {
2     // 0x8cd28b8294f34457 == hash("S0M3S3CR3TK3Y")
3     if (hash(serial) == 0x8cd28b8294f34457)
4         return 1;
5     else
6         return 0;
7 }
```

Листинг 2. Функция после обфускации

### 1.2. Сравнение с криптографическими хэш-функциями

Хотя к криптографическим хэш-функциям предъявляется множество различных требований [10], для целей обфускации не все из них строго обязательны [4]. Рассмотрим основные требования к криптографическим хэш-функциям и их применимость в данной работе:

1) Стойкость к поиску первого прообраза — сложность нахождения для данного $h$ такого $m$, что $f(m) = h$. Это требование очень важно для качественной обфускации, так как неразрывно связано с необратимостью хэш-функции. При невыполнении данного требования атакующий сможет восстановить исходные значения констант, нивелируя эффективность обфускации.

2) Стойкость к поиску второго прообраза — сложность нахождения для данного $m_1$ такого $m_2$, что $f(m_1) = f(m_2)$ и $m_1 \neq m_2$. В отличие от стойкости к поиску первого прообраза, данное требование не столь критично для обфускации. Существование таких коллизий неизбежно приведёт к нарушению свойства сохранения функциональности, но не повлияет негативно на стойкость программы к обратной разработке. Заметим, что в контексте сохранения функциональности наиболее интересны случайные коллизии, а не намеренно подобранные в результате криптографической атаки (если речь не идёт об участках кода, связанных с аутентификацией или другой обработкой недоверенного ввода, строго требующей отсутствия коллизий).
3) Стойкость к коллизиям — сложность нахождения таких $m_1$ и $m_2$, что $f(m_1) = f(m_2)$ и $m_1 \neq m_2$. Из данного требования следует стойкость к поиску второго прообраза, поэтому его также можно считать необязательным для специально выбранных $m_1$ и $m_2$. Однако именно это требование удобно использовать для проверки полученной функции при случайных $m_1$ и $m_2$. Если оно соблюдается, можно утверждать о сохранении функциональности с достаточной для стабильной работы обфусцированной программы точностью.

### 1.3. Специфические для обфускации требования

В то время как для криптографических хэш-функций наличие понятного и открытого описания является желательным согласно принципу Керхгоффса, в обфускации ценно запутывание любого рода, способное замедлить обратную разработку [11]. Соответственно для целей работы принцип Керхгоффса можно нарушить, не только скрыв от аналитика исходное сообщение, но и сделав саму функцию максимально непрозрачной. Для этого следует потребовать отсутствия закономерностей в её описании, например повторяющихся математических операций. Хорошим примером функции, нарушающей данное условие, является функция gp-hash, представленная в [8], в которой операция циклического сдвига вправо применяется к аргументу 18 раз подряд, что позволяет заменить эту цепочку операций одиночным сдвигом на 18 бит, сильно упростив код функции. Реализация этой функции и её упрощённый вариант представлены в листингах 3 и 4, где `ror` и `ror_18` — операция циклического сдвига вправо на 1 и 18 бит соответственно.

```
1 int gphash(int h, int m) {
2     return 0x6CF575C5 * ror(ror(ror(ror(ror(ror(ror(ror(ror
    (ror(ror(ror(ror(ror(ror(ror(ror(ror(0x6CF575C5 * (h + m
    ))))))))))))))))));
3 }
```

Листинг 3. Реализация исходного варианта функции gp-hash

```
1 int gphash(int h, int m) {
2     return 0x6CF575C5 * ror_18(0x6CF575C5 * (h + m));
3 }
```

Листинг 4. Упрощенная реализация функции gp-hash

Выполнение хэш-функций подразумевается в каждом из обфусцируемых условий, соответственно необходимо обеспечить высокую скорость их работы. Среди способов добиться этого можно отметить следующие:

1) Отсутствие раундовой структуры. Так как с ростом числа раундов пропорционально растёт количество исполняемых инструкций, не имеет смысла использовать множество раундов, если функция способна обеспечить требуемые свойства за один. Наряду с этим наличие повторяющихся раундов нарушает требование максимальной запутанности кода.
2) Использование в реализации операций, которые можно скомпилировать в одну машинную инструкцию целевой процессорной архитектуры. Некоторые операции, имеющие достаточно простую математическую запись (например, деление с остатком), могут требовать для реализации выполнения множества машинных инструкций, что делает их менее привлекательными для использования.

### 1.4. Итоговые требования к хэш-функциям для обфускации

Объединив требования к хэш-функциям, применимым в обфускации, поставленные в п. 1.2 и 1.3, получим следующий список:

1) стойкость к поиску первого прообраза — необратимость хэш-функции;
2) стойкость к случайным коллизиям — малая вероятность совпадения значений хэш-функции для случайных сообщений;
3) запутанность описания функции — отсутствие в описании хэш-функции закономерностей, позволяющих упростить её исследование;
4) высокая скорость работы — скорость работы, сравнимая с распространёнными некриптографическими хэш-функциями.

### 1.5. Количественные оценки

Одним из важных свойств криптографических хэш-функций является лавинный эффект — значительное изменение результата при изменении малого количества бит в исходном сообщении. Это свойство связано с понятием диффузии Шеннона и препятствует установлению связи между входными и выходными значениями, а значит, может оказаться полезным при обеспечении стойкости к поиску первого прообраза.

Математически лавинный эффект определяется следующим образом: при изменении одного бита входного сообщения в выходном значении должна измениться в среднем половина бит:

$$\forall x, y\,(\mathrm{d}(x, y) = 1 \;\Rightarrow \mathrm{d}(F(x), F(y)) \approx N/2).$$

Здесь $\mathrm{d}(x, y)$ — расстояние Хэмминга между $x$ и $y$; $F(x)$ — функция, обладающая лавинным эффектом; $N$ — размер её результата в битах.

Существует и более строгое условие — строгий лавинный критерий [12], при соблюдении которого автоматически соблюдается и условие лавинного эффекта. Он звучит так: при изменении одного бита входного сообщения каждый бит выходного сообщения должен измениться с вероятностью $1/2$; его можно представить в другом виде: общее количество изменившихся бит должно соответствовать биномиальному распределению с параметрами $(N, 1/2)$:

$$\forall x, y\,(\mathrm{d}(x, y) = 1 \;\Rightarrow\; \mathrm{d}(F(x), F(y)) \sim B(N, 1/2)). \tag{1}$$

Для данной работы выбрано именно это условие как более строгое. Соответствие полученного распределения образцовому оценивалось при помощи критерия согласия Пирсона.

Отсутствие закономерностей в описании хэш-функций, в свою очередь, связано с понятием колмогоровской сложности — минимально возможным алгоритмом, гене-

рирующим описание хэш-функции. Так как колмогоровская сложность является невычислимой, в качестве её замены можно использовать алгоритмы сжатия общего назначения, например Deflate, как предложено в [13]. Чем больший размер имеет функция в сжатом виде, тем больше оценка минимальной длины её описания, а значит, тем сложнее она для упрощения и анализа.

## 2. Алгоритм поиска хэш-функций для обфускации

### 2.1. Структура генерируемых хэш-функций

В качестве структуры хэш-функций выбран упрощённый вариант структуры Меркла — Дамгора: ввиду того, что стойкость к намеренному поиску коллизий не является обязательной для целей обфускации, этап с дополнением сообщения его длиной можно пропустить. Тогда функция будет выглядеть следующим образом:

$$H_0 = IV, \quad H_i = F(H_{i-1}, m_i), \quad i = 1, 2, \ldots \tag{2}$$

Здесь $m_i$ — части исходного сообщения $M$ одинакового размера; $F(h, m)$ — функция сжатия; $IV$ — некоторое начальное значение функции. Значение $H_n$ для сообщения из $n$ частей является результатом хэш-функции от всего сообщения $M$. В рамках данной работы длина $m_i$ и $H_i$ в битах считается равной.

Таким образом, задача поиска хэш-функций сводится к задаче поиска функций сжатия $F(h, m)$.

### 2.2. Генетическое программирование

Цель работы состоит в автоматической генерации функций, поэтому используемый алгоритм должен быть способен осуществлять поиск в пространстве всех возможных функций. Одним из немногих подходов, пригодных для решения задач подобного рода, является подход генетического программирования [14]. Обычно алгоритмы генетического программирования оперируют функциями, представленными в виде деревьев, где внутренние узлы являются операциями, а листья — операндами (или терминалами). На каждом шаге поиска решения к некоторой популяции функций применяются генетические операторы мутации и скрещивания, после чего в соответствии с требованиями, сформулированными в виде оптимизируемой функции приспособленности, формируется следующее поколение. После некоторого количества итераций поиск завершается и лучшая функция популяции выбирается решением задачи.

В качестве программной реализации выбрана библиотека DEAP для языка Python, содержащая готовые реализации описанного подхода и требующая только задания требований конкретной задачи [15].

### 2.3. Набор терминалов и операций

В соответствии со структурой хэш-функций (2) терминалами являются значение функции $h$, полученное на предыдущем этапе, и $m$ — текущий блок хэшируемого сообщения. В качестве операций выбраны операции, которые быстро выполняются на современных процессорах и являются достаточно простыми, с целью соблюдения предъявленного в п. 1.3 требования к производительности.

В результате выбраны следующие операции:

1) or(x, y), and(x, y), xor(x, y) — побитовые операции ИЛИ, И и исключающее ИЛИ;
2) add(x, y), mul(x, y) — беззнаковые сложение и умножение;
3) not(x) — побитовое отрицание;
4) ror_1(x), ror_half(x) — операции циклического сдвига вправо на 1 бит и на половину размера переменной: хотя вторую операцию можно выразить через

первую, многократный повтор операции противоречит условию максимальной запутанности из п. 1.3;

5) dynror(x, y) — циклический сдвиг x вправо на y бит; хотя эта операция и выглядит менее тривиально, чем предыдущие, на архитектуре x86, широко используемой на существующих компьютерах, она кодируется одной инструкцией `ROR r/m8, CL`.

### 2.4. Функция приспособленности

Искомая функция $F(h, m)$ применяется к двум аргументам, поэтому условие (1) уточнено следующим образом: потребуем соответствия функции строгому лавинному критерию для обеих переменных одновременно (как если бы $h$ и $m$ были одним аргументом $x$ двойного размера для функции $F(x)$).

Для количественного измерения соответствия условию (1) последовательно осуществляются следующие шаги:

1) генерация пары $h_0$, $m_0$ при помощи генератора псевдослучайных чисел;
2) подсчёт значения $F_0 = F(h_0, m_0)$;
3) изменение одного случайного бита в одной из переменных $h_0$ или $m_0$, новая пара значений после этой операции обозначается как $h_1$, $m_1$;
4) подсчёт значения $F_1 = F(h_1, m_1)$;
5) подсчёт и запоминание числа различающихся в $F_0$ и $F_1$ бит $\mathrm{d}(F_0, F_1)$.

После проведения достаточного числа итераций полученная выборка $\mathrm{d}(F_0, F_1)$ проверяется на соответствие распределению $B(N, 1/2)$ при помощи критерия согласия Пирсона: получаем значение $\chi^2$:

$$\chi^2 = \sum_{i=0}^{N} \frac{(O_i - E_i)^2}{E_i}.$$

Здесь $O_i$ — наблюдаемое количество итераций, где изменилось $i$ бит, а $E_i$ — ожидаемое количество для распределения $B(N, 1/2)$. В данной работе использован генератор случайных чисел MT19937 (вихрь Мерсенна), а число итераций, обеспечивающее достаточную точность оценки лавинного эффекта, было экспериментально установлено равным 1024 (оно должно быть значительно больше числа бит функции).

Для оценки запутанности кода полученной функции необходимо подвергнуть его сжатию. Для этого сначала необходимо преобразовать дерево функции $F(h, m)$ в линейное представление, что можно сделать средствами библиотеки DEAP. Затем каждый элемент (операция или терминал) заменяется численным представлением — индексом данной операции или терминала, в результате чего получается список чисел. Так как общее количество операций и терминалов не превышает $2^8$, каждый элемент занимает один байт, а значит, список можно тривиально преобразовать в байтовую строку. Полученная строка подвергается сжатию при помощи алгоритма Deflate, а длина сжатых данных является характеристикой $S$ функции $F(h, m)$.

Для обеспечения выполнения обоих условий из п. 1.5 значение $\chi^2$ должно быть подвергнуто минимизации, а $S$ — максимизации. Поэтому возникает вопрос объединения данных характеристик в одну функцию, оптимизируемую при помощи генетического программирования. Поскольку условие строгого лавинного критерия менее тривиально, решено отдать ему приоритет и только при достижении им некоторого целевого значения $\chi^2_{\mathrm{tgt}}$ обеспечивать отсутствие в функции закономерностей.

Так как $S$ намного меньше $\chi^2$ для условий данной работы, итоговая функция приспособленности, подвергающаяся минимизации, имеет следующий вид:

$$F_{\text{opt}} = \max(\chi^2 - \chi^2_{\text{tgt}}, 0) - S.$$

В качестве значения $\chi^2_{\text{tgt}}$ можно взять характеристику $\chi^2$ любой современной криптографической хэш-функции, стойкость которой к атакам поиска первого прообраза не вызывает опасений. В данной работе использована функция SHAKE256 из семейства SHA-3, удобная тем, что позволяет получить результат произвольной длины [16].

### 2.5. Описание алгоритма

Приведём разработанный метод в алгоритме 1. Здесь *measureSAC* и *measureComplexity* — функции оценки лавинного эффекта и запутанности кода (см. п. 2.4); *createPopulation* и *generateOffspring* — функции создания популяции и генерации потомков, предоставленные библиотекой генетического программирования DEAP. Функция *check* служит для дополнительной проверки функций на применимость в обфускации, принцип её работы описан далее в п. 4.4.

**Алгоритм 1.** Алгоритм поиска хэш-функций для обфускации

*params* := параметры генетического программирования;
$score_{\text{tgt}}$ := целевое значение функции;
$\chi^2_{\text{tgt}} := measureSAC(\text{SHAKE256})$;
*population* := *createPopulation*(*params*).
**Повторять**
  *population* := *generateOffspring*(*population*, *params*).
  **Для всех** $ind \in population$
    $ind.score := \max(measureSAC(ind) - \chi^2_{\text{tgt}}, 0) - measureComplexity(ind)$.
  $score_{\min} := \min\limits_{ind \in population}(ind.score)$.
**Пока** $score_{\min} \geqslant score_{\text{tgt}}$
$ind_{\text{best}} := \arg\min\limits_{ind \in population}(ind.score)$.
**Если** $check(ind_{\text{best}})$, **то**
  **Вывести** $ind_{\text{best}}$.

## 3. Проверка применимости полученных хэш-функций в обфускации

### 3.1. Метод автоматического анализа на обратимость

В программном обеспечении для осуществления деобфускации активно применяются подходы символьного исполнения и SMT-решатели [17, 18]. Использующие их инструменты angr и S2E способны автоматически генерировать входные данные, необходимые для выполнения различных ветвей условий программы, а значит, эти инструменты могут быть использованы и для атак на механизм обфускации, описанный в п. 1.1. По этой причине крайне важна проверка полученных функций на устойчивость к SMT-решателям. Проверка хэш-функций осуществлялась при помощи следующих инструментов:

1) Microsoft Z3 (версия 4.8.7) — наиболее важный решатель для целей данной работы, потому что именно он используется в инструменте автоматического анализа кода angr по умолчанию [17, 19];

2) CVC4 (версия 1.8) — решатель, который также поддерживается инструментом angr [20];
3) Yices2 (версия 2.6.2) — победитель трека QF_BV (application) в соревнованиях SMT-COMP 2018 [21];
4) Boolector (версия 3.2.1) — победитель трека QF_ABV в соревнованиях SMT-COMP 2018 и SMT-COMP 2019 [22].

Основой предлагаемого метода проверки хэш-функций является сравнение времени поиска $m$ для определённых $H = F(h, m)$ и $h$ при помощи SMT-решателей и полного перебора: если первое из времён больше (или решение вовсе не найдено), можно говорить об устойчивости функции к автоматическому анализу, так как SMT-решатели не способны ускорить процесс поиска.

Соответственно для проверки функции на обратимость достаточно осуществить следующие операции:

1) найти $T_{\text{bf}}$ — худшее возможное время атаки полным перебором. Это значение соответствует суммарному времени подсчёта $F(h, m)$ для всех возможных $m$ при фиксированном $h$;
2) найти $T_{\text{smt}}$ — время, которое лучший из SMT-решателей затратит на решение задачи нахождения $m$ по значению $H = F(h, m)$ для известного $h$;
3) сравнить $T_{\text{smt}}$ и $T_{\text{bf}}$ и сделать вывод об устойчивости функции, если $T_{\text{smt}} > T_{\text{bf}}$.

Для реализации программы перебора использовался язык Си в сочетании с компилятором GCC версии 10.1 с уровнем оптимизации O3. Перебор осуществлялся в однопоточном режиме.

Время анализа при помощи SMT-решателей измерялось путём их параллельного запуска на разных ядрах процессора. Время, через которое первый из SMT-решателей справлялся с задачей, являлось искомым $T_{\text{smt}}$. Длительность работы SMT-решателей сложно спрогнозировать, а время экспериментов не может быть неограниченным, поэтому максимальное время их работы ограничивалось некоторым $T_{\text{smtlim}}$, по достижении которого задача объявлялась неразрешимой за допустимое время.

### 3.2. Метод анализа устойчивости к случайным коллизиям

Традиционным способом анализа устойчивости к случайным коллизиям является использование парадокса дней рождения: при хэшировании случайных значений идеальной хэш-функцией длины $N$ бит через $\approx 2^{N/2}$ операций будет обнаружена коллизия. Данный способ исследован Д. Кнутом в [23], в результате чего установлено выражение для среднего количества операций до первой коллизии: $C_{\text{ideal}} = 1 + Q(2^N)$. Функция $Q(x)$, в свою очередь, изучена в работе [24], где для неё предложено следующее приближение:

$$Q(x) \approx \sqrt{\frac{\pi x}{2}} - \frac{1}{3} + \frac{1}{12}\sqrt{\frac{\pi}{2x}} - \frac{4}{135x} + \dots$$

Именно это приближение использовано в данной работе.

Для тестируемых функций среднее количество хэшированій до коллизии $C_{\text{bday}}$ может отличаться в меньшую сторону от $C_{\text{ideal}}$ в силу их неидеальности, например в случае, если реальная область значений хэш-функции меньше чем $\{0, \dots, 2^N - 1\}$. Оценить количество потерянных бит области значений $N_{\text{lost}}$ можно при помощи выражения $C_{\text{bday}} = 1 + Q(2^{N - N_{\text{lost}}})$

Предложим следующее условие устойчивости к случайным коллизиям: если для хэш-функции $N_{\text{lost}}$ мало в сравнении с $N$, то она является достаточно устойчивой

к случайным коллизиям. Метод анализа устойчивости к случайным коллизиям состоит из следующих шагов:

1) поиск $C_{\text{bday}}$ — количества хэшированний до первой коллизии. Для этого необходимо генерировать значения $H$ хэш-функции от случайных сообщений до первого совпадения с одним из ранее полученных значений, число выполненных итераций будет искомым значением. Здесь $H = F(F(IV, r_1), r_2)$ в соответствии со структурой хэш-функций (2), где $r_1$ и $r_2$ — случайные значения; $IV$ — некоторое постоянное значение. Для увеличения точности следует провести несколько экспериментов, усредняя $C_{\text{bday}}$;
2) подсчёт $N_{\text{lost}}$ с использованием выражения $C_{\text{bday}} = 1 + Q(2^{N-N_{\text{lost}}})$. Осуществляется численно, так как $Q(x)$ в любом случае является приближением;
3) сравнение $N_{\text{lost}}$ и $N$: если $N_{\text{lost}} \ll N$, то функция устойчива к случайным коллизиям.

Заметим, что в данном методе хэшируется сообщение из двух блоков ($r_1$ и $r_2$), а не из одного. Это связано с тем, что размер аргумента $m$ совпадает с размером функции $F(h, m)$, а значит, вероятность повтора случайных $m$ достаточно высока, чтобы повлиять на итоговое $C_{\text{bday}}$: она столь же высока, как и вероятность равенства выходных значений идеальной хэш-функции. Нас же интересуют только коллизии, где $f(m_1) = f(m_2)$, но $m_1 \neq m_2$, поэтому необходимо хэшировать сообщения длины, как минимум вдвое большей, чем длина значения функции: тогда вероятность равенства исходных сообщений становится пренебрежимой.

## 4. Экспериментальная проверка хэш-функций

### 4.1. Условия экспериментов

Для того чтобы полный перебор, требуемый для оценки устойчивости к автоматическому анализу, был практически осуществимым, в качестве длины аргументов и результата функции решено выбрать 32 бита. Хотя 32-битные хэш-функции, очевидно, мало подходят для защиты от подбора первоначальных значений, полученные с их использованием результаты могут быть распространены на функции большей разрядности.

Время выполнения итоговой функции определяется общим количеством операций, поэтому решено не вносить ограничений на высоту дерева функции в процессе генетического программирования, а использовать ограничение на общее число узлов. С целью обеспечения производительности, близкой к существующим некриптографическим хэш-функциям, это значение выбрано равным 32. Тогда для 32-битных функций для хэширования одного байта сообщения потребуется не более 8 инструкций (учитывая использованный в п. 2.3 набор операций), для сравнения — широко распространённой функции FNV требуется не меньше трёх, без учёта более частых условных переходов [25].

В отличие от традиционного использования генетического программирования для задач оптимизации, было решено не ограничивать общее число поколений, заменив его целевым значением функции приспособленности, равным $-30$, что соответствует достижению лавинных характеристик, сравнимых с характеристиками криптографических хэш-функций (в данном случае SHAKE256) и высокой степени запутанности (здесь целевое $S$ зависит от разрешенного максимального числа узлов).

Итоговые параметры, использованные для генетического программирования, приведены в табл. 1. Для экспериментов применялся компьютер с ОС Ubuntu Linux 18.04, процессором Intel Core i7-5820K (4 ГГц) и 32 ГБ ОЗУ.

Таблица 1

**Параметры генетического программирования**

| **Параметр** | **Значение** |
|---|---|
| Размер популяции | 200 |
| Вероятность скрещивания | 0,9 |
| Вероятность мутации | 0,1 |
| Максимальное число узлов | 32 |
| Набор операций | or, and, xor, add, mul, not, ror_1, ror_half, dynror |
| Набор терминалов | $h$, $m$ |
| Функция приспособленности | $\max(\chi^2 - \chi^2_{\text{tgt}}, 0) - S$ |
| Целевое значение функции | $-30$ |
| Отбор | Турнирный отбор размера 3 |
| Эволюционная стратегия | (1+1)-стратегия |

### 4.2. Результаты автоматического анализа

При помощи генетического программирования было сгенерировано 256 функций, подвергнутых проверке. Среднее наихудшее время полного перебора для сгенерированных функций оказалось равным $T_{\text{bf}} = 6{,}65\,\text{с}$. Так как SMT-решатели могут выполнять исследуемую функцию медленнее, чем оптимизированная программа на языке C, было решено взять максимальное время их работы равным $T_{\text{smtlim}} = 180\,\text{с}$. Наибольший интерес с практической точки зрения имеют времена решений меньшие, чем $T_{\text{bf}}$, поэтому значение $T_{\text{smtlim}}$ может быть выбрано приблизительно.

С учётом данных значений осуществлена проверка при помощи SMT-решателей, результаты которой приведены в табл. 2 и на рис. 1. Можно видеть, что, несмотря на ограниченное пространство поиска, в 159 случаях SMT-решатели не смогли найти ответ за время $T_{\text{smtlim}}$. Количество функций, в которых время решения $T_{\text{smt}}$ оказалось меньше $T_{\text{bf}}$, составило 53. Для этих функций можно считать, что алгоритм поиска хэш-функций совершенно не справился с задачей.

Таблица 2

**Результаты проверки 32-битных функций при помощи SMT-решателей**

| **Категория** | **Количество** |
|---|---|
| Функции, для которых $T_{\text{smt}} < T_{\text{bf}}$ | 53 |
| Функции, для которых $T_{\text{smt}} < T_{\text{smtlim}}$ | 97 |
| Функции, для которых $T_{\text{smt}} \geqslant T_{\text{smtlim}}$ | 159 |

Таким образом, в 62 % случаев алгоритм смог сгенерировать 32-битные функции, которые не поддаются анализу при помощи SMT-решателей за допустимое время, причём в 17 % оставшихся случаев время решения оказалось больше, чем худшее время полного перебора. Проанализировав гистограмму на рис. 1, заметим, что наибольшее количество решений происходит за небольшое время, причём за пределами 120 с количество решений сократилось до нуля. Если совместить поиск функций с их быстрой проверкой в течение времени $T_{\text{bf}}$, можно повысить долю неразрешимых за время $T_{\text{smtlim}}$ функций до 78 %.

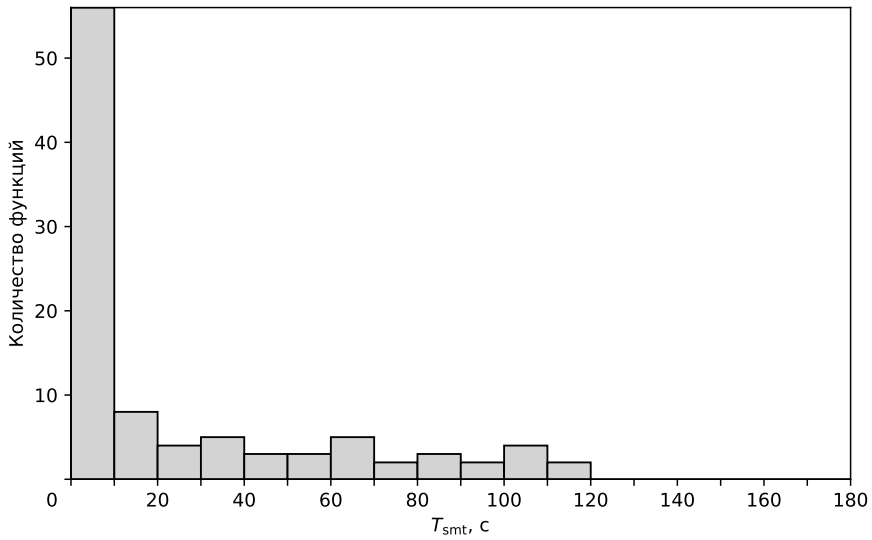


Рис. 1. Гистограмма количества 32-битных функций, для которых SMT-решатели нашли решение за указанное время, без учёта случаев $T_{\text{smt}} > T_{\text{smtlim}}$

### 4.3. Устойчивость к случайным коллизиям

Поскольку устойчивость к случайным коллизиям является второстепенным свойством после устойчивости к автоматическому анализу, проверке подвергались только те функции, которые не были обращены при помощи SMT-решателей за время меньшее $T_{\text{bf}}$. За счёт небольшой области значений функций описанная в п. 3.2 проверка может быть выполнена с достаточной для экспериментов производительностью. Для каждой функции измерялось среднее значение $C_{\text{bday}}$ за 512 попыток, после чего для него рассчитывалось приблизительное значение $N_{\text{lost}}$.

Для 203 проверенных функций среднее значение $C_{\text{bday}}$ составило 78080, наименьшее равно 63641. Отсюда среднее число потерянных бит $N_{\text{lost}} = 0{,}15$ и наихудшее — 0,74. Таким образом, в наихудшем случае хэш-функция имеет не больше коллизий, чем идеальная хэш-функция с длиной значения 31 бит.

### 4.4. Функции большей разрядности

Хотя многие 32-битные функции оказались устойчивы к анализу при помощи SMT-решателей, они уязвимы к переборным атакам (которые атакущий может провести на обычном процессоре в силу малого пространства поиска), вследствие чего было решено проанализировать функции с длинами значений, практически применимыми в обфускации: 64 и 128 бит. Для таких функций использование полного перебора затруднительно, поэтому в отличие от сценария, описанного в п. 3.1, в этих экспериментах имеет значение не сравнение $T_{\text{bf}}$ и $T_{\text{smt}}$, а принципиальная достижимость решения за какое бы то ни было разумное время.

Из соображений продолжительности эксперимента значение $T_{\text{smtlim}}$ оставлено прежним (180 с), для каждой размерности сгенерировано по 256 функций. Результаты проверки для 64- и 128-битных функций представлены на рис. 2 и 3 соответственно. Явление, наблюдавшееся для 32-битных функций, здесь ещё более выражено: для 64-битных функций максимальное время решения, меньшее $T_{\text{smtlim}}$, составило 17 с, а для 128-битных — 8 с. Таким образом, можно отсеивать крайне слабые функции, запуская сразу после их генерации дополнительную проверку, заключающуюся в анализе при помощи SMT-решателей в течение 17 и 8 с соответственно: если решение в течение этого времени найдено, функцию можно считать слабой. Этот способ позволяет повы-

сить долю неразрешимых за время $T_{\text{smtlim}}$ функций до 100 % и может быть реализован в функции *check* алгоритма 1.

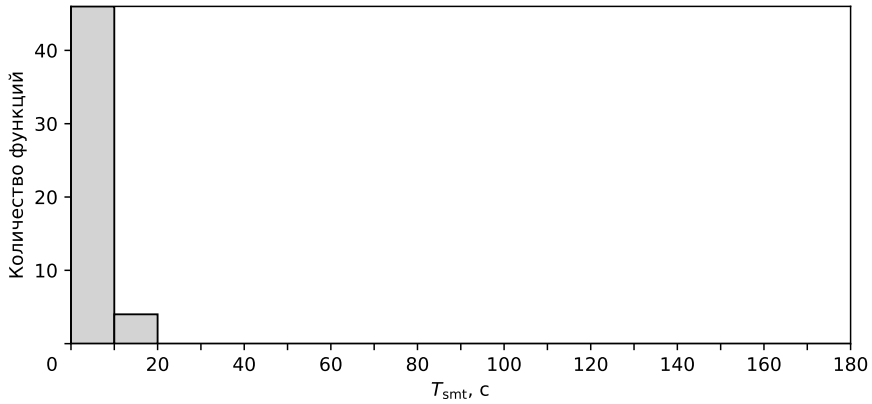


Рис. 2. Гистограмма количества 64-битных функций, для которых SMT-решатели нашли решение за указанное время, без учёта случаев $T_{\text{smt}} > T_{\text{smtlim}}$

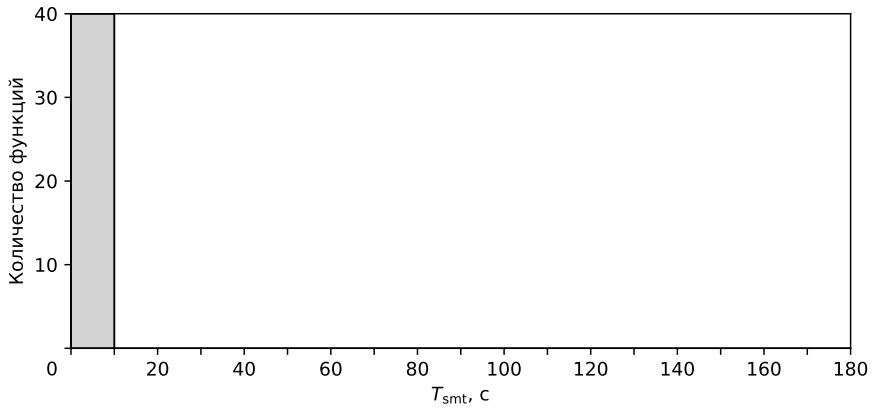


Рис. 3. Гистограмма количества 128-битных функций, для которых SMT-решатели нашли решение за указанное время, без учёта случаев $T_{\text{smt}} > T_{\text{smtlim}}$

Ввиду того, что может возникнуть вопрос адекватности выбора $T_{\text{smtlim}}$, было решено проверить несколько случайных функций, устойчивых к анализу за время $T_{\text{smtlim}}$, в течение намного большего времени. Для этого было выбрано по 16 соответствующих 64- и 128-битных функций, каждая из них подверглась анализу при помощи SMT-решателей в течение 4 ч. В результате эксперимента решение не было найдено ни для одной из функций, что позволяет утверждать, что данные функции являются устойчивыми к атакам с практической точки зрения.

## Заключение

Сформулированы требования к хэш-функциям, применимым для использования в обфускации, и показаны различия между ними и требованиями, предъявляемыми к криптографическим хэш-функциям. Данные требования выражены в виде количественных оценок. Предложен метод поиска хэш-функций для обфускации с использованием генетического программирования и этих оценок. Найденные функции проверены на устойчивость к атаке поиска первого прообраза при помощи SMT-решателей и на устойчивость к случайным коллизиям.

Полученные результаты позволяют утверждать, что генерируемые предложенным алгоритмом функции применимы для защиты от инструментов автоматического анализа кода, а также от обратной разработки в целом. Хотя сам по себе алгоритм не всегда генерирует устойчивые к автоматическому анализу функции, в сочетании со способом исключения слабых функций он даёт хорошие результаты. С небольшим общим временем генерации функций алгоритм генерирует 64- и 128-битные хэш-функции, устойчивые к анализу в течение достаточно длительного времени во всех проверенных случаях.

## ЛИТЕРАТУРА


1. *Banescu S., Collberg C. S., Ganesh V., et al.* Code obfuscation against symbolic execution attacks // Proc. 32nd Ann. Conf. Computer Security Appl. 2016. P. 189–200.
2. *Udupa S., Debray S., and Madou M.* Deobfuscation: Reverse Engineering Obfuscated Code. 12th Working Conf. WCRE'05. 2005. 10 p.
3. *Junod P., Rinaldini J., Wehrli J., and Michielin J.* Obfuscator-LLVM — software protection for the masses // IEEE/ACM 1st Intern. Workshop Software Protection. 2015. P. 3–9.
4. *Sharif M., Lanzi A., Giffin J., and Lee W.* Impeding malware analysis using conditional code obfuscation // Proc. NDSS. San Diego, California, USA, 2008.
5. `https://www.hex-rays.com/products/ida/lumina/` — IDA: Lumina server. 2020.
6. *Xu D., Ming J., and Wu D.* Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping // IEEE Symp. Security Privacy (SP). Los Alamitos, CA, USA, 2017. P. 921–937.
7. *Qiu W., Gong Z., Guo Y., et al.* GPU-based high performance password recovery technique for hash functions // J. Inform. Sci. Eng. 2016. V. 32. No. 1. P. 97–112.
8. *Estebanez C., Hernandez-Castro J., Ribagorda A., and Isasi P.* Finding state-of-the-art non-cryptographic hashes with genetic programming // LNCS. 2006. V. 4193. P. 818–827.
9. *Tapiador J., Hernandez-Castro J., Peris-Lopez P., and Ribagorda A.* Automated design of cryptographic hash schemes by evolving highly-nonlinear functions // J. Inform. Sci. Eng. 2008. V. 24. No. 5. P. 1485–1504.
10. *Menezes A., van Oorschot P., and Vanstone S.* Handbook of Applied Cryptography. CRC Press, 1996. 661 p.
11. *Collberg C. S. and Thomborson C.* Watermarking, tamper-proofing, and obfuscation — tools for software protection // IEEE Trans. Software Eng. 2002. V. 28. No. 8. P. 735–746.
12. *Webster A. F. and Tavares S. E.* On the design of S-boxes // LNCS. 1985. V. 218. P. 523–534.
13. *Cilibrasi R. and Vitanyi P. M. B.* Clustering by compression // IEEE Trans. Inform. Theory. 2005. V. 51. No. 4. P. 1523–1545.
14. *Koza J. R.* Genetic programming as a means for programming computers by natural selection // Stat. Comput. 1994. V. 4. No. 2. P. 87–112.
15. *Fortin F. A., De Rainville F. M., Gardner M. A. G., et al.* DEAP: Evolutionary algorithms made easy // J. Machine Learning Res. 2012. V. 13. No. 1. P. 2171–2175.
16. *Dworkin M. J.* SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology, 2015.
17. *Shoshitaishvili Y., Wang R., Salls C., et al.* SOK: (State of) The art of war: Offensive techniques in binary analysis // IEEE Symp. Security Privacy (SP). 2016. P. 138–157.
18. *Chipounov V., Kuznetsov V., and Candea G.* S2E: A platform for in-vivo multi-path analysis of software systems // ACM SIGPLAN Notices. 2011. V. 46. No. 3. P. 265–278.
19. *De Moura L. and Bjorner N.* Z3: An efficient SMT solver // LNCS. 2008. V. 4963. P. 337–340.

20. *Barrett C., Conway C. L., Deters M., et al.* CVC4 // Intern. Conf. Comput. Aided Verification. 2011. P. 171–177.
21. *Dutertre B.* Yices 2.2 // Intern. Conf. Comput. Aided Verification. 2014. P. 737–744.
22. *Brummayer R. and Biere A.* Boolector: An efficient SMT solver for bit-vectors and arrays // LNCS. 2009. V. 5505. P. 174–177.
23. *Knuth D. E.* The Art of Computer Programming. V. 3: Sorting and Searching. 2nd Ed. Addison-Wesley Professional, 1998. 800 p.
24. *Flajolet P., Grabner P. J., Kirschenhofer P., and Prodinger H.* On Ramanujan's Q-function // J. Computat. Appl. Math. 1995. V. 58. No. 1. P. 103–116.
25. `http://www.isthe.com/chongo/tech/comp/fnv/` — FNV Hash. 1991.

## REFERENCES

1. *Banescu S., Collberg C. S., Ganesh V., et al.* Code obfuscation against symbolic execution attacks. Proc. 32nd Ann. Conf. Computer Security Appl., 2016, pp. 189–200.
2. *Udupa S., Debray S., and Madou M.* Deobfuscation: Reverse Engineering Obfuscated Code. 12th Working Conf. WCRE'05, 2005. 10 p.
3. *Junod P., Rinaldini J., Wehrli J., and Michielin J.* Obfuscator-LLVM — software protection for the masses. IEEE/ACM 1st Intern. Workshop Software Protection, 2015, pp. 3–9.
4. *Sharif M., Lanzi A., Giffin J., and Lee W.* Impeding malware analysis using conditional code obfuscation. Proc. NDSS. San Diego, California, USA, 2008.
5. `https://www.hex-rays.com/products/ida/lumina/` — IDA: Lumina server. 2020.
6. *Xu D., Ming J., and Wu D.* Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping. IEEE Symp. Security Privacy (SP), Los Alamitos, CA, USA, 2017, pp. 921–937.
7. *Qiu W., Gong Z., Guo Y., et al.* GPU-based high performance password recovery technique for hash functions. J. Inform. Sci. Eng., 2016, vol. 32, no. 1, pp. 97–112.
8. *Estebanez C., Hernandez-Castro J., Ribagorda A., and Isasi P.* Finding state-of-the-art non-cryptographic hashes with genetic programming. LNCS, 2006, vol. 4193, pp. 818–827.
9. *Tapiador J., Hernandez-Castro J., Peris-Lopez P., and Ribagorda A.* Automated design of cryptographic hash schemes by evolving highly-nonlinear functions. J. Inform. Sci. Eng., 2008, vol. 24, no. 5, pp. 1485–1504.
10. *Menezes A., van Oorschot P., and Vanstone S.* Handbook of Applied Cryptography. CRC Press, 1996. 661 p.
11. *Collberg C. S. and Thomborson C.* Watermarking, tamper-proofing, and obfuscation — tools for software protection. IEEE Trans. Software Eng., 2002, vol. 28, no. 8, pp. 735–746.
12. *Webster A. F. and Tavares S. E.* On the design of S-boxes. LNCS, 1985, vol. 218, pp. 523–534.
13. *Cilibrasi R. and Vitanyi P. M. B.* Clustering by compression. IEEE Trans. Inform. Theory, 2005, vol. 51, no. 4, pp. 1523–1545.
14. *Koza J. R.* Genetic programming as a means for programming computers by natural selection. Stat. Comput., 1994, vol. 4, no. 2, pp. 87–112.
15. *Fortin F. A., De Rainville F. M., Gardner M. A. G., et al.* DEAP: Evolutionary algorithms made easy. J. Machine Learning Res., 2012, vol. 13, no. 1, pp. 2171–2175.
16. *Dworkin M. J.* SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology, 2015.
17. *Shoshitaishvili Y., Wang R., Salls C., et al.* SOK: (State of) The art of war: Offensive techniques in binary analysis. IEEE Symp. Security Privacy (SP), 2016, pp. 138–157.
18. *Chipounov V., Kuznetsov V., and Candea G.* S2E: A platform for in-vivo multi-path analysis of software systems. ACM SIGPLAN Notices, 2011, vol. 46, no. 3, pp. 265–278.

19. *De Moura L. and Bjorner N.* Z3: An efficient SMT solver. LNCS, 2008, vol. 4963, pp. 337–340.
20. *Barrett C., Conway C. L., Deters M., et al.* CVC4. Intern. Conf. Comput. Aided Verification, 2011, pp. 171–177.
21. *Dutertre B.* Yices 2.2. Intern. Conf. Comput. Aided Verification, 2014, pp. 737–744.
22. *Brummayer R. and Biere A.* Boolector: An efficient SMT solver for bit-vectors and arrays. LNCS, 2009, vol. 5505, pp. 174–177.
23. *Knuth D. E.* The Art of Computer Programming. vol. 3: Sorting and Searching. 2nd Ed. Addison-Wesley Professional, 1998. 800 p.
24. *Flajolet P., Grabner P. J., Kirschenhofer P., and Prodinger H.* On Ramanujan's Q-function. J. Computat. Appl. Math., 1995, vol. 58, no. 1, pp. 103–116.
25. `http://www.isthe.com/chongo/tech/comp/fnv/` — FNV Hash. 1991.

УДК 510.52

# О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ О СУММЕ ПОДМНОЖЕСТВ ДЛЯ ПОЛУГРУПП ЦЕЛОЧИСЛЕННЫХ МАТРИЦ[1]

А. Н. Рыбалов

*Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия*

В 2003 г. Каповичем, Мясниковым, Шуппом и Шпильрайном была предложена теория генерической вычислимости и сложности вычислений. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Проблема о сумме подмножеств является классической комбинаторной проблемой, изучаемой многие десятилетия. Мясников, Николаев и Ушаков в 2015 г. ввели аналог этой проблемы для произвольных групп (полугрупп). Оказалось, что для некоторых классов групп, таких, как гиперболические и нильпотентные группы, эта проблема разрешима за полиномиальное время. Для других, например групп Баумслага — Солитера и группы унимодулярных целочисленных матриц второго порядка $SL_2(\mathbb{Z})$, эта проблема NP-полна. Из работ Гуревича, Каи, Фукса, Козена и Лиу следует, что проблема о сумме подмножеств для группы $SL_2(\mathbb{Z})$ и для моноида $SL_2(\mathbb{N})$ полиномиально разрешима для почти всех входов. В работе изучается генерическая сложность проблемы о сумме подмножеств для полугрупп матриц произвольного порядка с целыми неотрицательными элементами. Эта проблема является NP-полной, а потому при условии $\mathrm{P} \neq \mathrm{NP}$ нет полиномиального алгоритма, решающего её для всех входов. Доказывается, что проблема является генерически разрешимой за полиномиальное время. Предлагается полиномиальный генерический алгоритм, основанный на методе динамического программирования.

**Ключевые слова:** *генерическая сложность, проблема о сумме подмножеств, полугруппы целочисленных матриц.*

DOI 10.17223/20710410/50/9

# ON GENERIC COMPLEXITY OF THE SUBSET SUM PROBLEM FOR SEMIGROUPS OF INTEGER MATRICES

A. N. Rybalov

*Sobolev Institute of Mathematics, Omsk, Russia*

**E-mail:** alexander.rybalov@gmail.com

Generic-case approach to algorithmic problems was suggested by Miasnikov, Kapovich, Schupp, and Shpilrain in 2003. This approach studies behavior of an algorithm on typical (almost all) inputs and ignores the rest of inputs. The subset sum problem is a classic combinatorial problem that has been studied for many decades. Myasnikov, Nikolaev and Ushakov in 2015 introduced an analogue of this problem for arbitrary groups (semigroups). For some classes of groups, such as hyperbolic and nilpotent groups, this problem is solvable in polynomial time. For others, for example, Baumslag — Solitaire groups, group of second order integer unimodular matrices $SL_2(\mathbb{Z})$, this problem is NP-complete. From the works of Gurevich, Kai, Fuchs,

[1]Работа поддержана грантом РНФ № 18-71-10028.

Cosen, and Liu, it follows that the subset sum problem for the group $SL_2(\mathbb{Z})$ and for the monoid $SL_2(\mathbb{N})$ is polynomially solvable for almost all inputs. In the paper, we study the generic complexity of the subset sum problem for semigroups of matrices of arbitrary order with integer non-negative elements. This problem is NP-complete, and therefore for it, provided P $\neq$ NP, there is no polynomial algorithm that solves it for all inputs. We present a polynomial generic algorithm based on the dynamic programming and prove that this problem is generically solvable in polynomial time.

**Keywords:** *generic complexity, the subset sum problem, integer matrix semigroups.*

## Введение

Проблема о сумме подмножеств является классической комбинаторной проблемой, изучаемой многие десятилетия. Она содержится в классическом списке из двадцати одной NP-полной проблемы в знаменитой работе [1]. Её формулировка состоит в следующем. Дано множество натуральных чисел $A = \{a_1, \ldots, a_n\}$ и натуральное число $S$. Все числа записаны в двоичном виде. Необходимо определить, можно ли в множестве $A$ выбрать подмножество чисел, которые в сумме дают число $S$. Эта задача имеет родство с другой классической проблемой оптимизации — о рюкзаке, её иногда считают частным случаем проблемы о рюкзаке. Большую популярность проблема о сумме подмножеств приобрела в криптографии, где неоднократно предлагались криптосистемы, основанные на ней [2, 3]. Отметим, что Н. Н. Кузюрин доказал полиномиальную разрешимость в среднем некоторых проблем рюкзачного типа [4].

А. Мясников, А. Николаев и А. Ушаков ввели аналоги проблем о рюкзаке и о сумме подмножеств для произвольных групп (полугрупп) [5]. Это является некоторым обобщением классических проблем, так как в классическом случае входные данные берутся из группы целых чисел $\mathbb{Z}$ с операцией сложения, заданной с помощью бесконечной системы порождающих $\{2^m : m = 0, 1, 2, \ldots\}$. В [5] изучена вычислительная сложность этих проблем для различных групп: доказаны полиномиальная разрешимость для гиперболических групп и NP-полнота для групп Баумслага — Солитера. Отмечена связь с классической проблемой о вхождении в подгруппу. Сложность в среднем ограниченной проблемы вхождения для группы $SL_2(\mathbb{Z})$ унимодулярных целочисленных $(2 \times 2)$-матриц изучалась А. Бласcом и Ю. Гуревичем [6]. С использованием идей Гуревича из [7] в [8] предложен полиномиальный в среднем алгоритм для ограниченной проблемы вхождения в модулярной группе $PSL_2(\mathbb{Z})$. Полиномиальный в среднем алгоритм для ограниченной проблемы вхождения для полугруппы унимодулярных натуральнозначных $(2 \times 2)$-матриц $SL_2(\mathbb{N})$ предложен в [9]. Эти алгоритмы без особого труда могут быть переделаны в эффективные алгоритмы, решающие проблемы рюкзака и суммы подмножеств для упомянутых групп и полугрупп матриц. Однако они не работают для полугрупп неунимодулярных матриц и матриц порядка больше 2, так как существенно используют структуру полугруппы $SL_2(\mathbb{N})$ как свободного двупорождённого моноида.

В 2003 г. в [10] предложена теория генерической вычислимости и сложности вычислений. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют генерическое множество; понятие «почти все» формализуется введением естественной меры на множестве входных данных. С точки зрения практики алгоритмы, быстро решающие проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов.

В данной работе изучается генерическая сложность проблемы о сумме подмножеств для полугрупп матриц произвольного порядка с целыми неотрицательными элементами. Эта проблема в классическом смысле является NP-полной, а потому при условии P $\neq$ NP нет полиномиального алгоритма, решающего её для всех входов. Доказывается, что проблема является генерически разрешимой за полиномиальное время. Предлагается полиномиальный генерический алгоритм, основанный на методе динамического программирования.

## 1. Генерические алгоритмы

Пусть $I$ — множество всех входов, $I_n$ — множество входов размера $\leqslant n$. Для любого подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где $S_n = S \cap I_n$ — множество входов из $S$ размера $\leqslant n$. *Асимптотической плотностью* $S$ назовём предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество $S$ называется *генерическим*, если $\rho(S) = 1$, и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что $S$ генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов $I$ и множеством выходов $J \cup \{?\}$ $(? \notin J)$ называется *генерическим*, если

1) $\mathcal{A}$ останавливается на всех входах из $I$;
2) множество $\{x \in I : \mathcal{A}(x) = ?\}$ является пренебрежимым.

Здесь через $\mathcal{A}(x)$ обозначается результат работы алгоритма $\mathcal{A}$ на входе $x$. Генерический алгоритм $\mathcal{A}$ *вычисляет функцию* $f : I \to J$, если для всех $x \in I$ имеет место

$$\mathcal{A}(x) \neq ?\ \Rightarrow\ f(x) = \mathcal{A}(x).$$

Ситуация $\mathcal{A}(x) = ?$ означает, что $\mathcal{A}$ не может вычислить функцию $f$ на аргументе $x$. Но условие 2 гарантирует, что $\mathcal{A}$ корректно вычисляет $f$ на почти всех входах (входах из генерического множества).

## 2. Полугруппа натуральнозначных матриц

Обозначим через $\mathbb{N}$ множество натуральных чисел, начинающееся с единицы, а через $\omega$ — множество натуральных чисел с нулём. Будем работать в основном с полугруппой матриц $Mat(k, \omega)$ порядка $k$ с целыми неотрицательными элементами с обычной операцией умножения матриц. Иногда будем использовать полугруппу $Mat(k, \mathbb{N})$. Порядок матриц $k$ фиксирован. Очевидно, $Mat(k, \mathbb{N}) \subseteq Mat(k, \omega)$. Элементы $Mat(k, \omega)$ будем представлять матрицами из целых неотрицательных чисел. Размер целого положительного числа $a$, обозначаемый $\mathrm{size}(a)$, — это длина его двоичной записи. Таким образом, $\mathrm{size}(a) = n$, если $2^{n-1} \leqslant a < 2^n$. Отдельно положим $\mathrm{size}(0) = 0$. Легко видеть, что для любых натуральных $a, b \neq 0$ выполнено $\mathrm{size}(ab) = \mathrm{size}(a) + \mathrm{size}(b)$ и $\mathrm{size}(a) \leqslant \mathrm{size}(a + b)$. Под размером матрицы $M = ||a_{ij}||$ будем понимать $\mathrm{size}(M) = \max\limits_{i,j=1,\ldots,k} \{\mathrm{size}(a_{ij})\}$.

**Лемма 1.** Для любого $n$ имеет место

$$|Mat(k,\omega)_n| = 2^{nk^2},\ |Mat(k,\mathbb{N})_n| = (2^n-1)^{k^2}.$$

***Доказательство.*** На каждом из $k^2$ мест матрицы из множества $Mat(k,\omega)_n$ может стоять число от 0 до $(2^n-1)$, то есть $2^n$ вариантов. Всего получаем $2^{nk^2}$ различных матриц. Аналогично производится подсчёт $|Mat(k,\mathbb{N})_n|$. ■

Докажем несколько фактов о полугруппах матриц, которые используются при построении и обосновании генерического полиномиального алгоритма для проблемы о сумме подмножеств.

**Лемма 2.**

1) Пусть $Z$ — множество матриц в $Mat(k,\omega)$, содержащих хотя бы один нулевой элемент. Тогда $Z$ пренебрежимо в $Mat(k,\omega)$.
2) Пусть $S \subseteq Mat(k,\mathbb{N})$. Если $S$ пренебрежимо в $Mat(k,\mathbb{N})$, то $S$ пренебрежимо в $Mat(k,\omega)$.

***Доказательство.***

1) Оценим число матриц из $Z_n$. Нуль можно поставить на одно из $k^2$ мест, остальные $k^2-1$ мест заполняются произвольно. Таким образом, $|Z_n| \leqslant k^2 2^{n(k^2-1)}$ и

$$\rho(Z) = \lim_{n\to\infty} \frac{|Z_n|}{|Mat(k,\omega)_n|} \leqslant \lim_{n\to\infty} \frac{k^2 2^{n(k^2-1)}}{2^{nk^2}} = \lim_{n\to\infty} \frac{k^2}{2^n} = 0.$$

2) Пусть

$$\rho(S) = \lim_{n\to\infty} \frac{|S_n|}{|Mat(k,\mathbb{N})_n|} = \lim_{n\to\infty} \frac{|S_n|}{(2^n-1)^{k^2}} = 0.$$

Тогда

$$\rho(S) = \lim_{n\to\infty} \frac{|S_n|}{|Mat(k,\omega)_n|} = \lim_{n\to\infty} \frac{|S_n|}{2^{nk^2}} \leqslant \lim_{n\to\infty} \frac{|S_n|}{(2^n-1)^{k^2}} = 0.$$

Лемма 2 доказана. ■

**Лемма 3.** Множество матриц из $Mat(k,\omega)$ с определителем 0 пренебрежимо.

***Доказательство.*** Рассмотрим сначала множество $Mat(k,\mathrm{GF}(p))$ квадратных матриц порядка $k$ с элементами из поля $\mathrm{GF}(p)$, где $p$ — простое число. Очевидно, что $|Mat(k,\mathrm{GF}(p))| = p^{k^2}$. Найдём среди всех таких матриц долю обратимых матриц, у которых определитель отличен от нуля в $\mathrm{GF}(p)$. Такие матрицы образуют группу $\mathrm{GL}(k,\mathrm{GF}(p))$. Известно [11], что $|\mathrm{GL}(k,\mathrm{GF}(p))| = \prod_{i=1}^{k}(p^k - p^{i-1})$. Отсюда искомая доля равна

$$P = \frac{|\mathrm{GL}(k,\mathrm{GF}(p))|}{|Mat(k,\mathrm{GF}(p))|} = \frac{\prod_{i=1}^{k}(p^k - p^{i-1})}{p^{k^2}} = \prod_{i=1}^{k}(1-p^{-i}).$$

Обозначим через $Mat_0(k,\mathrm{GF}(p))$ множество матриц из $Mat(k,\mathrm{GF}(p))$ с определителем 0. Доля таких матриц равна

$$\frac{|Mat_0(k,\mathrm{GF}(p))|}{|Mat(k,\mathrm{GF}(p))|} = 1 - P = 1 - \prod_{i=1}^{k}(1-p^{-i}).$$

Вернёмся к $Mat(k,\omega)$. Согласно лемме 2, достаточно доказать, что матрицы из $Mat(k,\mathbb{N})$ с нулевым определителем образуют пренебрежимое множество в $Mat(k,\mathbb{N})$.

Зафиксируем размер матриц $n$. Выберем простое число $p$, являющееся делителем $2^n-1$. Пусть $\varphi_p : \{1,\dots,2^n-1\} \to \mathrm{GF}(p)$ — отображение, определяемое для любого $a \in \{0,\dots,2^n-1\}$ как $\varphi_p(a) = a \bmod p$. Соответствующее индуцированное отображение для матриц тоже будем обозначать $\varphi_p$. Заметим, что прообраз каждого элемента из $\mathrm{GF}(p)$ при отображении $\varphi_p$ состоит ровно из $(2^n-1)/p$ чисел из отрезка $1,\dots,2^n-1$. Поэтому для любого множества матриц $\mathcal{S} \subseteq Mat(k,\mathrm{GF}(p))$ имеет место

$$|\varphi_p^{-1}(\mathcal{S})| = \Big(\frac{2^n-1}{p}\Big)^{k^2}|\mathcal{S}|.$$

Пусть теперь $\mathcal{Z}$ — множество матриц из $Mat(k,\mathbb{N})_n$ с определителем 0. Так как $\mathcal{Z} \subseteq \varphi_p^{-1}(Mat_0(k,\mathrm{GF}(p))$, имеем

$$\frac{|\mathcal{Z}|}{|Mat(k,\mathbb{N})_n|} \leqslant \frac{|\varphi_p^{-1}(M_0(k,\mathrm{GF}(p))|}{|Mat(k,\mathbb{N})_n|} = \frac{\Big(\frac{2^n-1}{p}\Big)^{k^2}|Mat_0(k,\mathrm{GF}(p))|}{\Big(\frac{2^n-1}{p}\Big)^{k^2}|Mat(k,\mathrm{GF}(p))|} =$$
$$= \frac{|Mat_0(k,\mathrm{GF}(p))|}{|Mat(k,\mathrm{GF}(p))|} = 1 - \prod_{i=1}^{k}(1-p^{-i}).$$

При увеличении $p$ значение справа стремится к нулю. Осталось доказать, при увеличении размера $n$ значение $p$ тоже увеличивается. Напомним, что простое число $p$ выбиралось как делитель $2^n-1$. По классической теореме Жигмонди [12], начиная с $m > 6$ каждый элемент последовательности $\{2^m-1 :\ m = 1,2,\dots\}$ имеет простой делитель, на который не делятся предыдущие члены этой последовательности. Поэтому с ростом $n$ можно выбирать простые делители $p$ числа $2^n-1$ так, чтобы значение $p$ неограничено увеличивалось. ■

**Лемма 4.** Пусть для матриц $A = ||a_{ij}||$ и $B = ||b_{ij}||$ из $Mat(k,\omega)$ имеет место $\mathrm{size}(A), \mathrm{size}(B) \leqslant n$ и $\mathrm{size}(AB) \leqslant n$. Тогда для любых $1 \leqslant i,j \leqslant k$ имеет место $\mathrm{size}(a_{ij}) + \mathrm{size}(b_{ji}) \leqslant n$.

***Доказательство.*** Пусть $AB = ||c_{ij}||$. Тогда для любого $i = 1,\dots,k$

$$c_{ii} = a_{i1}b_{1i} + a_{i2}b_{2i} + \dots + a_{ij}b_{ji} + \dots + a_{ik}b_{ki}.$$

Так как $\mathrm{size}(c_{ii}) \leqslant n$, для каждого $j = 1,\dots,k$ и положительных $a_{ij}, b_{ji}$ имеем $\mathrm{size}(a_{ij}b_{ji}) \leqslant n$, т. е. $\mathrm{size}(a_{ij}) + \mathrm{size}(b_{ji}) \leqslant n$. Если $a_{ij} = 0$ или $b_{ji} = 0$, то требуемое неравенство следует из условия $\mathrm{size}(a_{ij}), \mathrm{size}(b_{ji}) \leqslant n$. ■

**Лемма 5.** Пусть $S_n$ — множество матриц $M$ из $Mat(k,\omega)$ размера $n$, таких, что матричное уравнение $M = XY$ имеет более $p(n) = (2(n+1))^{k^2+1}$ различных решений $X, Y \in Mat(k,\omega)$, таких, что $\mathrm{size}(X), \mathrm{size}(Y) \leqslant n$. Тогда

$$\frac{|S_n|}{|Mat(k,\omega)_n|} < \frac{1}{2(n+1)}.$$

***Доказательство.*** Пусть $K$ — число различных пар матриц $X, Y$, таких, что $\mathrm{size}(X), \mathrm{size}(Y) \leqslant n$ и $\mathrm{size}(XY) \leqslant n$. Тогда из определения множества $S_n$ следует

$$(2(n+1))^{k^2+1}|S_n| \leqslant K. \tag{1}$$

Оценим сверху число $K$. Пусть $X = ||x_{ij}||$, $Y = ||y_{ij}||$. По лемме 4 для любых $i, j$ выполнено $\mathrm{size}(x_{ij}) + \mathrm{size}(y_{ji}) \leqslant n$. Поэтому число $K$ не превосходит числа пар таких матриц $X, Y$, что $\mathrm{size}(x_{ij}) + \mathrm{size}(y_{ji}) \leqslant n$ для $1 \leqslant i, j \leqslant k$, т. е.

$$K \leqslant |\{(X,Y) : \mathrm{size}(x_{ij}) + \mathrm{size}(y_{ji}) \leqslant n,\ 1 \leqslant i,j \leqslant k\}| =$$
$$= \prod_{1\leqslant i,j\leqslant k} |\{(a,b) : \mathrm{size}(a) + \mathrm{size}(b) \leqslant n\}| = \Big(|\{(a,b) : \mathrm{size}(a) + \mathrm{size}(b) \leqslant n\}|\Big)^{k^2} =$$
$$= \Big(\sum_{m=0}^{n} |\{(a,b) : \mathrm{size}(a) + \mathrm{size}(b) = m\}|\Big)^{k^2} =$$
$$= \Big(\sum_{m=0}^{n}\sum_{l=0}^{m} |\{(a,b) : \mathrm{size}(a) = l,\ \mathrm{size}(b) = m-l\}|\Big)^{k^2} =$$
$$= \Big(\sum_{m=0}^{n}\sum_{l=0}^{m} 2^l \cdot 2^{m-l}\Big)^{k^2} = \Big(\sum_{m=0}^{n} m2^m\Big)^{k^2} < \Big((n+1)\sum_{m=0}^{n} 2^m\Big)^{k^2} =$$
$$= \Big((n+1)(2^{n+1}-1)\Big)^{k^2} < (2(n+1))^{k^2}2^{nk^2} = (2(n+1))^{k^2}|Mat(k,\omega)_n|.$$

Таким образом, получили оценку

$$K < (2(n+1))^{k^2}|Mat(k,\omega)_n|. \tag{2}$$

Допустим теперь противное, то есть что $\dfrac{|S_n|}{|Mat(k,\omega)_n|} > \dfrac{1}{2(n+1)}$. Тогда

$$(2(n+1))^{k^2+1}|S_n| > (2(n+1))^{k^2}|Mat(k,\omega)_n|.$$

Но это неравенство и оценки (1) и (2) дают противоречие

$$(2(n+1))^{k^2}|Mat(k,\omega)_n| < (2(n+1))^{k^2+1}|S_n| \leqslant K < (2(n+1))^{k^2}|Mat(k,\omega)_n|.$$

Лемма 5 доказана. ■

### 3. Проблема о сумме подмножеств над $Mat(k,\omega)$

Сформулируем проблему о сумме подмножеств для полугруппы $Mat(k,\omega)$. Даны матрицы $M_1, M_2, \ldots, M_n$ из $Mat(k,\omega)$, каждая размером не более $n$, и матрица $M$ из $\mathcal{M}$ размера не более $n^2$. Определить, существуют ли степени $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \in \{0,1\}$, такие, что имеет место

$$M_1^{\varepsilon_1} M_2^{\varepsilon_2} \ldots M_n^{\varepsilon_n} = M.$$

Размер входа $(M_1, M_2, \ldots, M_n, M)$ полагаем равным $n$. Условимся, что если матрица $M = E$ (единичная), то $M$ есть произведение пустого подмножества матриц, а потому проблема суммы подмножеств с такой матрицей $M$ разрешима. Это допущение послужит для удобства описания генерического алгоритма в дальнейшем.

Следующее утверждение говорит, что для этой проблемы при условии P $\neq$ NP не существует полиномиального алгоритма, который решает её для всех входов.

**Лемма 6.** Проблема о сумме подмножеств для $Mat(k,\omega)$ NP-полна.

***Доказательство.*** Докажем, что к данной проблеме полиномиально сводится классическая проблема о сумме подмножеств для натуральных чисел. Определим

функцию $F : \omega \to Mat(k, \omega)$ следующим образом. Для любого $a \in \omega$ положим

$$F(a) = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & a \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

Соответствующая полиномиальная сводимость сопоставляет входу $(a_1, a_2, \dots, a_n, S)$ классической проблемы о сумме подмножеств вход $(F(a_1), F(a_2), \dots, F(a_n), F(S))$ проблемы о сумме подмножеств для $Mat(k, \omega)$. Легко проверить, что для любого набора чисел $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n \in \{0, 1\}$ равенство

$$a_1\varepsilon_1 + a_2\varepsilon_2 + \dots + a_n\varepsilon_n = S$$

выполняется тогда и только тогда, когда $F(a_1)^{\varepsilon_1} F(a_2)^{\varepsilon_2} \dots F(a_n)^{\varepsilon_n} = F(S)$. ■

Следующая теорема говорит, что для данной проблемы существует генерический полиномиальный алгоритм, решающий её для почти всех входов.

**Теорема 1.** Проблема о сумме подмножеств для полугруппы $Mat(k, \omega)$ является генерически разрешимой за полиномиальное время.

***Доказательство.*** Опишем, как работает генерический полиномиальный алгоритм $\mathcal{A}$ на входе $(M_1, \dots, M_n, M)$ размера $n$.

1) Вычисляем определитель матрицы $M$. Если он равен 0, выдаём ответ «?». Из леммы 3 следует, что множество входов $(M_1, \dots, M_n, M)$, у которых $\det(M) = 0$, пренебрежимо.
2) Вычисляем определители матриц $M_1, \dots, M_n$. Выбрасывам те матрицы, у которых определитель равен 0, — они не могут участвовать в произведении, которое равно $M$, так как $\det(M) \neq 0$. На последующих шагах считаем, что все матрицы $M_1, \dots, M_n, M$ невырождены.
3) В дальнейшей работе алгоритм $\mathcal{A}$ осуществляет рекурсивные вызовы алгоритма $\mathcal{A}$ на других входных данных. Будем контролировать число таких вызовов, для чего заведём счётчик вызовов $R$, который сначала равен 0.
4) Если счётчик $R$ стал равен $np(n^2)$, где $p$ — полином из леммы 5, то останавливаем все рекурсивные вызовы и выдаём ответ «?».
5) Если $M = E$, то останавливаем все запущенные рекурсивные вызовы алгоритма $\mathcal{A}$ и выдаём ответ «ДА».
6) Для каждой матрицы $M_i$, $i = 1, \dots, n$, решаем матричное уравнение $M_i X = M$. Оно сводится к системе линейных уравнений, которое решаем методом Гаусса в рациональных числах. Если решение получается в натуральных числах, то делаем следующее:
   а) проверяем, был ли запущен рекурсивный вызов $\mathcal{A}(M_{i+1}, \dots, M_n, X)$ ранее;
   б) если нет, то запускаем рекурсивно алгоритм $\mathcal{A}$ на входе $(M_{i+1}, \dots, M_n, X)$ и увеличиваем счетчик рекурсивных вызовов: $R := R + 1$.

   Это делаем для каждого матричного уравнения $M_i X = M$, разрешимого в натуральных числах.

7) Если ни одна из этих систем неразрешима в натуральных числах, то останавливаем текущий рекурсивный вызов алгоритма $\mathcal{A}$ и выдаем ответ «НЕТ РЕШЕНИЯ НА ДАННОЙ ПОДЗАДАЧЕ».
8) Если все запущенные рекурсивные вызовы в какой-то момент остановились и выдали ответ «НЕТ РЕШЕНИЯ НА ДАННОЙ ПОДЗАДАЧЕ», то выдаём ответ «НЕТ».

Докажем полиномиальность алгоритма $\mathcal{A}$. Каждая вычислительная процедура (метод Гаусса, вычисление определителя), используемая внутри алгоритма, работает за полиномиальное время. Кроме того, число рекурсивных вызовов алгоритма $\mathcal{A}$ ограничено полиномиально — оно не превосходит $np(n^2)$, где $p$ — полином из леммы 5.

Докажем теперь генеричность алгоритма $\mathcal{A}$. Из леммы 5 следует, что множество входов $(M_1, \ldots, M_n, M)$ проблемы суммы подмножеств размера $n$, в которых число решений матричного уравнения $M = XY$ в матрицах из $\mathcal{M}$ не превосходит $p(n^2)$, является генерическим. Заметим, что для любого такого входа $(M_1, \ldots, M_n, M)$ алгоритм $\mathcal{A}$ выдаст ответ, отличный от ответа «?». Для того чтобы убедиться в этом, оценим число возможных подзадач $(M_i, \ldots, M_n, M')$, для которых происходят рекурсивные вызовы алгоритма $\mathcal{A}$. Для всех матриц, кроме последней, имеется не более $n$ вариантов выбора. Для матрицы $M'$ всегда найдётся матрица $X$, такая, что $M = XM'$. Значит, число вариантов выбора матрицы $M'$ не может быть больше числа решений матричного уравнения $M = XY$ в матрицах из $Mat(k, \omega)$, то есть, согласно лемме 5, $p(n^2)$. Итого получаем не более $np(n^2)$ вариантов для возможных подзадач $(M_i, \ldots, M_n, M')$. А так как в алгоритме счетчик числа рекурсивных вызовов ограничен как раз значением $np(n^2)$, то все эти рекурсивные вызовы происходят и соответствующие подзадачи решаются. Поэтому на таких входах алгоритм обязательно выдаёт ответ, отличный от «?». ■

## ЛИТЕРАТУРА


1. *Karp R.* Reducibility among combinatorial problems // R. E. Miller and J. W. Thather (eds). Complexity of Computer Computations. IBM Research Symposia Ser. 1972. P. 85–103.
2. *Hellman M. and Merkle R.* Hiding information and signatures in trapdoor knapsacks // IEEE Trans. Inform. Theory. 1978. V. 24. No. 5. P. 525–530.
3. *Chor B. and Rivest R.* A knapsack-type public key cryptosystem based on arithmetic in finite fields // IEEE Trans. Inform. Theory. 1988. V. 34. No. 5. P. 901–909.
4. *Кузюрин Н. Н.* Полиномиальный в среднем алгоритм в целочисленном линейном программировании // Сибирский журнал исследования операций. 1994. Т. 1. № 3. С. 38–48.
5. *Miasnikov A., Nikolaev A., and Ushakov A.* Knapsack problems in groups // Math. Comput. 2015. V. 84. P. 987–1016.
6. *Blass A. and Gurevich Yu.* Matrix transformation is complete for the average case // SIAM J. Computing. 1995. V. 24. No. 1. P. 24–39.
7. *Gurevich Yu.* Matrix decomposition problem is complete for the average case // Proc. 31st Ann. Symp. Foundations of Computer Science. 1990. P. 802–811.
8. *Cai J., Fuchs W., Kozen D., and Liu Z.* Efficient average-case algorithms for the modular group // Proc. 35th Ann. Symp. Foundations of Computer Science. 1994. P. 143–152.
9. *Cai J. and Liu Z.* The bounded membership problem of the monoid $SL_2(N)$ // Math. Systems Theory. 1996. V. 29. P. 573–587.
10. *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.
11. *Каргаполов М. И., Мерзляков Ю. И.* Основы теории групп. М.: Наука, 1982. 288 с.

12. *Zsigmondy K.* Zur Theorie der Potenzreste // Monatshefte für Math. u. Phys. 1882. V. 3. P. 265–284.

REFERENCES

1. *Karp R.* Reducibility among combinatorial problems. R. E. Miller and J. W. Thather (eds.), Complexity of Computer Computations. IBM Research Symposia Ser., 1972, pp. 85–103.
2. *Hellman M. and Merkle R.* Hiding information and signatures in trapdoor knapsacks. IEEE Trans. Inform. Theory, 1978, vol. 24, no. 5, pp. 525–530.
3. *Chor B. and Rivest R.* A knapsack-type public key cryptosystem based on arithmetic in finite fields. IEEE Trans. Inform. Theory, 1988, vol. 34, no. 5, pp. 901–909.
4. *Kuzyurin N. N.* Polinomialnyi v srednem algoritm v tselochilennom lineinom programmirovanii [Polynomial on average algorithm in integer linear programming]. Sib. J. Operation Research, 1994, vol. 1, no. 3, pp. 38–48. (in Russian)
5. *Miasnikov A., Nikolaev A., and Ushakov A.* Knapsack problems in groups. Math. Comput., 2015, vol. 84, pp. 987–1016.
6. *Blass A. and Gurevich Yu.* Matrix transformation is complete for the average case. SIAM J. Computing, 1995, vol. 24, no. 1, pp. 24–39.
7. *Gurevich Yu.* Matrix decomposition problem is complete for the average case. Proc. 31st Ann. Symp. Foundations of Computer Science, 1990, pp. 802–811.
8. *Cai J., Fuchs W., Kozen D., and Liu Z.* Efficient average-case algorithms for the modular group. // Proc. 35th Ann. Symp. Foundations of Computer Science, 1994, pp. 143–152.
9. *Cai J. and Liu Z.* The bounded membership problem of the monoid $SL_2(N)$. Math. Systems Theory, 1996, vol. 29, pp. 573–587.
10. *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.
11. *Kargapolov M. I. and Merzlyakov Yu. I.* Osnovy teorii grupp [Elements of the Group Theory]. Moscow, Nauka Publ., 1982. 288 p. (in Russian)
12. *Zsigmondy K.* Zur Theorie der Potenzreste. Monatshefte für Math. u. Phys., 1882, vol. 3, pp. 265–284.

# СВЕДЕНИЯ ОБ АВТОРАХ

**АГИБАЛОВ Геннадий Петрович** — доктор технических наук, профессор, главный научный сотрудник лаборатории компьютерной криптографии Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **agibalov@mail.tsu.ru**

**АЛЕКСЕЕВ Евгений Константинович** — кандидат физико-математических наук, начальник отдела криптографических исследований ООО «КРИПТО-ПРО», г. Москва. E-mail: **alekseev@cryptopro.ru**

**БОРОВКОВА Ирина Вячеславовна** — аспирантка Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **iborovkova95@gmail.com**

**ВОБЛЫЙ Виталий Антониевич** — доктор физико-математических наук, ведущий научный сотрудник Всероссийского института научной и технической информации РАН, г. Москва. E-mail: **vitvobl@yandex.ru**

**ДЕУНДЯК Владимир Михайлович** — кандидат физико-математических наук, доцент, доцент Южного федерального университета, г. Ростов-на-Дону.
E-mail: **vl.deundyak@gmail.com**

**ЗУБОВ Анатолий Юрьевич** — кандидат физико-математических наук, доцент, ООО «Центр сертификационных исследований», г. Москва.
E-mail: **zubovanatoly@yandex.ru**

**КОНДРАТЬЕВ Вадим Андреевич** — студент Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **wadim.condratjev@yandex.ru**

**КОСОЛАПОВ Юрий Владимирович** — кандидат технических наук, доцент Южного федерального университета, г. Ростов-на-Дону. E-mail: **itaim@mail.ru**

**ЛЕБЕДЕВ Роман Константинович** — аспирант Новосибирского государственного университета, г. Новосибирск. E-mail: **n0n3m4@gmail.com**

**ПАНКРАТОВА Ирина Анатольевна** — кандидат физико-математических наук, доцент, заведующая лабораторией компьютерной криптографии Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **pank@mail.tsu.ru**

**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**

**СМЫШЛЯЕВ Станислав Витальевич** — кандидат физико-математических наук, заместитель генерального директора ООО «КРИПТО-ПРО», г. Москва.
E-mail: **svs@cryptopro.ru**

**ХУНГ Ле Хуан** — Ханойский университет природных ресурсов и окружающей среды, г. Ханой. E-mail: **lxhung@hunre.edu.vnl**

Журнал «Прикладная дискретная математика» включен в перечень ВАК рецензируемых российских журналов, в которых должны быть опубликованы основные результаты диссертаций, представляемых на соискание учёной степени кандидата и доктора наук, в базы данных Web of Science (Emerging Sources Citation Index — ESCI и Russian Science Citation Index — RSCI), Scopus и MathSciNet, а также в перечень журналов, рекомендованных ФУМО ВО ИБ в качестве учебной литературы по специальности «Компьютерная безопасность».

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 в журнал.

**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*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*