МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2021 № 51

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

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

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

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

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

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Подписано к печати 15.03.2021. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 15. Тираж 300 экз.
Заказ № 4611. Цена свободная. Дата выхода в свет 19.03.2021.

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

СОДЕРЖАНИЕ

CONTENTS

ПАМЯТИ УЧИТЕЛЯ

К величайшему сожалению, 16 декабря 2020 г. ушёл из жизни Геннадий Петрович Агибалов — основатель и бессменный главный редактор журнала «Прикладная дискретная математика».

Геннадий Петрович Агибалов родился 17 мая 1939 г. в селе Казанцево Шушенского района Красноярского края. В 1961 г. окончил радиофизический факультет Томского государственного университета (ТГУ) и до конца жизни работал на педагогических и научных должностях в ТГУ, заведовал кафедрами электронной вычислительной техники и автоматики, математической логики и проектирования, лабораторией синтеза дискретных автоматов.

В 1968 г. он защитил кандидатскую диссертацию на тему криптографической защиты управляющей информации, в 1993 г. — докторскую, посвящённую теории дискретных автоматов на полурешётках.

С 1972 г. Г. П. Агибалов возглавлял в ТГУ ведущую научную школу прикладной дискретной математики, известную в мире созданными в ней русским языком программирования и автоматизированными системами логического проектирования, достижениями в криптографии, теории автоматов, компьютерной безопасности. На базе этих достижений он создал в ТГУ кафедру защиты информации и криптографии (в 1999 г.) и лабораторию компьютерной криптографии (в 2018 г.).

В 2002 г. Г. П. Агибалов организовал Всероссийскую конференцию «Сибирская научная школа-семинар с международным участием "Компьютерная безопасность и криптография" — SIBECRYPT», которая с тех пор с неизменным успехом каждый год проходит в разных городах и весях Сибири, собирая до 60 участников из многих городов Российской Федерации, а также Белоруссии и Украины.

В 2008 г. Г. П. Агибалов основал ежеквартальный журнал «Прикладная дискретная математика». До конца жизни он был его главным редактором, через его руки проходили все статьи, которые он внимательно прочитывал, правил (а зачастую и полностью переписывал) английские аннотации. Под руководством Геннадия Петровича журнал вошёл в перечень ВАК, международные базы цитирования Scopus, WoS, MathSciNet и Zentralblatt MATH.

Образовательная программа ТГУ по специальности «Компьютерная безопасность», разработанная Г. П. Агибаловым, трижды (2011–2013 гг.) по оценкам независимого органа — журнала «Аккредитация в образовании» — входила в число лучших образовательных программ инновационной России.

В 2015 г. Государственная Дума Российской Федерации присвоила кафедре защиты информации и криптографии звание «Образовательный центр года РФ в области информационной безопасности»; её студенты Олег Брославский (2015 г.) и Артём Ивачёв (2016 г.) получили звание «Студент года РФ в области информационной безопасности».

Студенческая команда SiBears кафедры — трёхкратный чемпион России и многократный призёр международных соревнований CTF в области информационной безопасности.

Г. П. Агибалов награждён нагрудными знаками «Почётный работник высшего профессионального образования Российской Федерации» (2003 г.) и «Почётный работник науки и техники Российской Федерации» (2009 г.)

По результатам научной деятельности в области прикладной дискретной математики и криптографии Г. П. Агибалов опубликовал более 100 статей в рецензируемых журналах и сборниках, 5 монографий и 4 учебных пособия. Под его руководством успешно защищены 12 кандидатских и 3 докторских диссертации.

Наиболее крупным научным достижением Г. П. Агибалова является создание им в 1980-х — начале 1990-х гг. основ теории дискретных автоматов на полурешётках, открывших новое научное направление на стыке алгебры (полурешёток), дискретной математики (дискретных функций и автоматов) и математической кибернетики (управляющих систем), в рамках которого, помимо решения некоторых чисто алгебраических задач, относящихся к кодированию полурешёток и полурешёточно упорядоченным алгебрам, ему впервые удалось формализовать такие понятия, относящиеся к дискретным автоматам, как динамическое поведение (вызываемое асинхронным изменением входных сигналов), его физическая реализуемость, адекватная модель и её точность, и все традиционные задачи логического проектирования, включая задачи эквивалентных преобразований, минимизации, декомпозиции, кодирования, моделирования, анализа, синтеза, функциональной полноты и др., решить в постановке, отражающей динамику поведения автомата, возможность её физической реализации на

современной электронной базе и адекватность моделирования с любой наперед заданной точностью. Докторская диссертация Г. П. Агибалова на эту тему признана в ВАК РФ л у ч ш е й за 1993 г. по специальности 05.13.01.

Последние 20 лет научной деятельности Г. П. Агибалова были целиком посвящены непосредственно криптографии и её применениям в компьютерной безопасности. В ряду результатов исследований, проведённых им в этот период, можно отметить: алгоритмы криптоанализа некоторых симметричных и асимметричных шифров (2000 г.); метод линеаризационного множества в решении систем уравнений над конечным полем и линеаризационная атака на генераторы ключевого потока (ГКП) со сложностью меньше, чем у корреляционной атаки (2003 г.); алгоритмы криптоанализа сжимающего и самосжимающего ГКП методом решения их систем логических уравнений (2004 г.); вероятностные схемы симметричного поточного шифрования над конечным полем со свойством совершенности (2005 г.); элементы теории дифференциального криптоанализа итеративных блочных шифров с аддитивным раундовым ключом и алгоритм такого криптоанализа методом решения системы уравнений последнего раунда (2008 г.); элементы теории статистических аналогов дискретных функций и их применение в криптоанализе итеративных блочных шифров с эффектом более высоким, чем у линейного криптоанализа M. Matsui (2010 г.); описание криптосистем с функциональными ключами на базе векторных булевых функций и конечных автоматов (2014 г.); условия обратимости с конечной задержкой конечных автоматов для различных типов обратимости (2017 г.).

Геннадий Петрович был очень увлечённым человеком, всегда полон идей — будь то новое понятие, необходимое для решения научной проблемы, или ещё одно спортивное состязание в честь Дня кафедры (в которых, к слову, сам Геннадий Петрович неизменно участвовал), или неожиданное (порой экзотическое) место проведения школы-семинара SIBECRYPT ... Он заражал своим энтузиазмом всех вокруг, не боялся трудностей (коих было немало), никогда не делал выбор в пользу чего-то, что было бы выгодно ему лично, но «неправильно» в целом или вредило бы его Делу.

Геннадий Петрович Агибалов навсегда останется в сердцах и душах коллег и учеников, и мы все, бесконечно скорбя о невосполнимой потере, будем всё же осознавать своё счастье от того, что нам довелось встретиться с таким Человеком, учиться у него и работать рядом с ним.

О своём отношении к жизни, Родине, работе, людям лучше всего сказал сам Геннадий Петрович в своём стихотворении.

От имени коллег и учеников
И. А. Панкратова

МЫ

(штрихи к автопортрету)

Все поколенья в жизни проходили
Стихии, войны, мор иль гнет властей,
И только мы сторонкой проскочили
От этих самых гибельных страстей.

Нам повезло, мы начали с Победы,
Которую отцы нам принесли,
И 50 спокойных лет безбедно
Учились и трудились, как могли.

Трудились не за деньги, не за славу
И не искали сверхпостов себе;
Желали одного — чтоб за державу
Не стыдно было нам в своей судьбе.

Ценили честь превыше, чем карьеру,
И поступали, совесть как вела,
Свободными в достаточную меру
Мы оставались в мыслях и делах.

Не верили ни в Бога и ни в Чёрта,
Но свято чтили славный красный флаг,
И никогда под членов в «Волгах» чёрных
Не прогибались ради личных благ.

Мы оставляли близких без вниманья,
Нам было некогда согреть родных.
Любили мать с отцом на расстояньи,
О, как теперь нам не хватает их!

Друзей нередко всуе обижали,
Но их обиды вмиг прощали им,
Учителей посмертно вспоминали,
Но забывали позвонить живым.

Политика нас слабо занимала,
Прохвосты ж находились всякий раз,
И вот однажды всей страны не стало
И вместе с нею отменили нас.

Теперь цивилизованные страны,
Спасённые Россией от врагов,
Как те монголы, требуют с нас дани,
А мы не в силах вразумить их вновь.

Есть дети, внуки и работы много,
Но на душе великая тоска:
Бессмысленная жизнь, и у порога
Уже стоит надгробная доска.

Одна лишь просьба к поколеньям новым:
Когда наш луч сверкнёт из вечной тьмы,
Вы нас хорошим помяните словом —
Не самые плохие были мы.

Г. П. Агибалов
23.02.2001

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 511.23

АЛГОРИТМ ВЫЧИСЛЕНИЯ ИДЕАЛА ШТИКЕЛЬБЕРГЕРА ДЛЯ МУЛЬТИКВАДРАТИЧНЫХ ПОЛЕЙ[1]

Е. А. Киршанова, Е. С. Малыгина, С. А. Новоселов, Д. О. Олефиренко

Балтийский федеральный университет им. И. Канта, г. Калининград, Россия

Представлен алгоритм вычисления идеала Штикельбергера для мультиквадратичного поля $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$, $i \in \{1, \ldots, n\}$, или некоторый $d_j \equiv \pm 2 \bmod 8$, $j \in \{1, \ldots, n\}$, все d_i — целые, попарно взаимно простые и свободные от квадратов. В основу работы положена статья Р. Кучеры (J. Number Theory, 1996, no. 56). Мы предлагаем алгоритм вычисления идеала Штикельбергера, работающий за время $\mathcal{O}(\lg \Delta_K \cdot 2^n \cdot \mathrm{poly}(n))$, где Δ_K — дискриминант поля K. В качестве приложения показана взаимосвязь идеала Штикельбергера с числом классов мультиквадратичного поля.

Ключевые слова: *мультиквадратичные поля, элемент Штикельбергера, идеал Штикельбергера, группа классов мультиквадратичного поля.*

DOI 10.17223/20710410/51/1

AN ALGORITHM FOR COMPUTING THE STICKELBERGER IDEAL FOR MULTIQUADRATIC NUMBER FIELDS

E. A. Kirshanova, E. S. Malygina, S. A. Novoselov, D. O. Olefirenko

Immanuel Kant Baltic Federal University, Kaliningrad, Russia

E-mail: {ekirshanova, emalygina, snovoselov, dolefirenko}@kantiana.ru

We present an algorithm for computing the Stickelberger ideal for multiquadratic fields $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, where the integers $d_i \equiv 1 \bmod 4$ for $i \in \{1, \ldots, n\}$ or $d_j \equiv 2 \bmod 8$ for one $j \in \{1, \ldots, n\}$; all d_i's are pairwise co-prime and square-free. Our result is based on the paper of Kučera [J. Number Theory, no. 56, 1996]. The algorithm we present works in time $\mathcal{O}(\lg \Delta_K \cdot 2^n \cdot \mathrm{poly}(n))$, where Δ_K is the discriminant of K. As an interesting application, we show a connection between Stickelberger ideal and the class number of a multiquadratic field.

Keywords: *multiquadratic number field, Stickelberger element, Stickelberger ideal, class group of multiquadratic field.*

[1]Работа Киршановой Е. А. выполнена при частичной финансовой поддержке конкурса «Молодая математика России» 2020 и Программы мобильности 5-100.

Введение

Идеал Штикельбергера I числового поля K — это идеал групповой алгебры $\mathbb{Z}[G_K]$, где $G_K = \mathrm{Gal}(K/\mathbb{Q})$ — группа Галуа поля K. Ключевое свойство идеала Штикельбергера, известное как теорема Штикельбергера [1] (современное изложение см. в [2, § 6.2]), заключается в том, что элементы этого идеала аннигилируют группу классов K, то есть для любого $\sigma \in I$ и любого дробного идеала J кольца целых поля K идеал J^σ является главным.

Получение идеала Штикельбергера в явном виде, то есть вычисление его образующих, является важной алгоритмической задачей в алгебраической теории чисел [3] и, с недавних пор, в криптоанализе [4].

Так, например, явный вид идеала Штикельбергера для кругового поля $K = \mathbb{Q}(\zeta_r)$ ($r > 0, r \in \mathbb{Z}$), описанный в [2], позволил получить алгоритм [4] нахождения короткого вектора в идеалах кольца целых кругового поля; в современном криптоанализе нахождение короткого вектора в решётках является основополагающей задачей.

В данной работе рассматриваются другие поля, а именно мультиквадратичные поля $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$ свободны от квадратов и попарно взаимно просты. Предлагается алгоритм вычисления идеала Штикельбергера поля K. Такой алгоритм интересен, в первую очередь, с точки зрения вычислений группы классов поля K (см. п. 4). В криптографии эта задача возникает в конструкциях проверяемых функций задержки (VDF) [5, 6] и в гомоморфном шифровании [7].

Алгоритм, описанный в п. 3, имеет сложность $\mathcal{O}(\lg \Delta_K \cdot 2^n \cdot \mathrm{poly}(n))$, где Δ_K — дискриминант K. Таким образом, он является полиномиальным от степени расширения $[K : \mathbb{Q}] = 2^n$ и имеет логарифмическую зависимость от дискриминанта поля. В основе работы лежат результаты [8], где представлены основные ингредиенты алгоритма.

Результаты работы:

— Предложен алгоритм вычисления образующих идеала Штикельбергера для мультиквадратичных полей вида $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$ свободны от квадратов и попарно взаимно просты. Этот алгоритм, во-первых, систематизирует идеи, описанные в [8]. Во-вторых, расширена область действия алгоритма на поля с образующими $d_i \equiv \pm 2 \bmod 8$.

— Разработана эффективная реализация данного алгоритма[2].

В п. 1 приведены определения, связанные с идеалом Штикельбергера, и описано, как с помощью гауссовых сумм мультиквадратичное поле вкладывается в круговое. Пункты 2 и 3 посвящены алгоритму вычисления образующих идеала Штикельбергера и анализу его сложности. В п. 4 изучается связь идеала Штикельбергера с числом классов поля K.

Предварительные результаты данной работы были представлены на конференции SIBECRYPT'20 [9].

1. Предварительные сведения

1.1. Определение идеала Штикельбергера

Рассмотрим мультиквадратичные поля $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_1, \ldots, d_n \in \in \mathbb{Z}$ попарно взаимно просты и свободны от квадратов. Через Δ_K обозначим дискриминант K. Для $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, заданного взаимно простыми и свободными

[2] `https://gitlab.com/DenisOl/stickelberger-ideal`

от квадратов образующими d_i, известно [10, Satz 2.1], что $\Delta_K = (2^a \prod\limits_i d_i)^{2^{n-1}}$, где $a \in \{0, 2, 3\}$.

Рассмотрим башню числовых полей $\mathbb{Q} \subseteq K \subseteq L$ и обозначим их группы Галуа $G_L = \mathrm{Gal}(L/\mathbb{Q})$ и $G_K = \mathrm{Gal}(K/\mathbb{Q})$. Группу, конечно-порождённую элементами из G_L над $\mathbb{Q}$ (соответственно элементами G_K над $\mathbb{Q}$), будем обозначать $\mathbb{Q}[G_L] = \{\sum a_i \cdot \sigma_i : a_i \in \mathbb{Q}, \sigma_i \in G_L\}$ $\big(\mathbb{Q}[G_K] = \{\sum a_j \cdot \sigma_j : a_j \in \mathbb{Q}, \sigma_j \in G_K\}\big)$. Важными понятиями при вычислении элементов Штикельбергера являются отображения res и cor. Определим их, согласно [8, с. 157], для расширения числовых полей L/K:

$$\mathsf{res}_{L/K} : \mathbb{Q}[G_L] \to \mathbb{Q}[G_K], \quad \mathsf{res}_{L/K}\Big(\sum_{\sigma \in G_L} a_\sigma \sigma\Big) = \sum_{\sigma \in G_L} a_\sigma (\sigma|_K),$$

$$\mathsf{cor}_{L/K} : \mathbb{Q}[G_K] \to \mathbb{Q}[G_L], \quad \mathsf{cor}_{L/K}\Big(\sum_{\sigma \in G_K} a_\sigma \sigma\Big) = \sum_{\sigma \in G_L} a_{\sigma|_K} \sigma,$$

где $\sigma|_K$ — сужение автоморфизма $\sigma \in G_L$ на поле K; a_σ, $a_{\sigma|_K}$ — коэффициенты, соответствующие автоморфизмам σ, $\sigma|_K$.

Обозначим: $\langle\cdot\rangle$, $0 \leqslant \langle\cdot\rangle < 1$, — дробная часть числа; (a, b) — наибольший общий делитель двух элементов $a, b \in \mathbb{Z}$; $\left(\dfrac{a}{b}\right)$ — символ Кронекера — Якоби для a, b (для нечётного $b > 1$ — символ Якоби, для простого нечётного b — символ Лежандра); $\#A$ — мощность множества A; $\varphi(n)$ — функция Эйлера.

Дадим классические определения элемента и идеала Штикельбергера [11, с. 189].

Определение 1. Для любого положительного целого r, любого $\alpha \in \mathbb{Z}$ и кругового поля $\mathbb{Q}(\zeta_r)$ определим

$$\theta_r(\alpha) = \sum_{(a,r)=1} \left\langle -\tfrac{\alpha a}{r} \right\rangle \sigma_a^{-1} \in \mathbb{Q}[G_{\mathbb{Q}(\zeta_r)}],$$

где $0 < a \leqslant r$ и $\sigma_a \in G_{\mathbb{Q}(\zeta_r)} = \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q})$.

Определение 2. Для любого положительного целого r и произвольного $\alpha \in \mathbb{Z}$ элементом Штикельбергера $\theta_r'(\alpha)$ называется элемент вида

$$\theta_r'(\alpha) = \left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_r)/K\cap\mathbb{Q}(\zeta_r)}\right)(\theta_r(\alpha)) \in \mathbb{Q}[G_K],$$

где K и $\mathbb{Q}(\zeta_r)$ — абелево числовое поле и круговое поле соответственно.

Определение 3. Идеалом Штикельбергера поля K называется идеал вида $I = I' \cap \mathbb{Z}[G_K]$, где

$$I' = \{\theta_r'(\alpha) : \alpha, r \in \mathbb{Z}, r \geqslant 1\}.$$

1.2. Квадратичные гауссовы суммы

Дадим определение квадратичных гауссовых сумм, а также покажем, как они связаны с автоморфизмами круговых полей, поскольку эта взаимосвязь поможет вычислить действие отображения res и, как следствие, элемент Штикельбергера соответствующего числового поля.

Определение 4. Пусть $m, k \in \mathbb{Z}$, $k > 0$. Квадратичная гауссова сумма определяется как $g(m, k) = \sum\limits_{b=0}^{k-1} e^{2\pi i m b^2/k}$.

Следующая теорема позволяет выражать квадратные корни, которые можно рассматривать как элементы мультиквадратичного поля, через квадратичные гауссовы суммы.

Теорема 1 [12, 1.5.2, с. 26]. Пусть $(m,k)=1$, $k>0$ и k нечётное. Тогда

$$g(m,k)=\left(\frac{m}{k}\right)g(1,k)=\begin{cases}\left(\frac{m}{k}\right)\sqrt{k}, & k\equiv 1 \bmod 4,\\ \left(\frac{m}{k}\right)i\sqrt{k}, & k\equiv 3 \bmod 4.\end{cases}$$

Заметим, что если $-k\equiv 1 \bmod 4$, то $k\equiv 3 \bmod 4$. Отсюда следует, что в этом случае $\sqrt{-k}=g(1,k)$ по теореме 1.

Рассмотрим действие автоморфизмов кругового поля $\mathbb{Q}(\zeta_k)$ на корни $\sqrt{k}$, которое необходимо для вычисления отображения res. Всякий автоморфизм поля $\mathbb{Q}(\zeta_k)$ имеет вид $\sigma_a(e^{2\pi i/k})=e^{2\pi ia/k}$. В случае, когда $k=p$ — нечётное простое и $p\nmid a$, согласно [12, (1.5.3), с. 26], имеем

$$\sigma_a(g(1,p))=\left(\frac{a}{p}\right)g(1,p)=\begin{cases}\left(\frac{a}{p}\right)\sqrt{p}, & p\equiv 1 \bmod 4,\\ \left(\frac{a}{p}\right)\sqrt{-p}, & -p\equiv 1 \bmod 4.\end{cases}$$

Отсюда следует, что $\sigma_a(\sqrt{p})=\left(\frac{a}{p}\right)\sqrt{p}$, если $p\equiv 1 \bmod 4$, и $\sigma_a(\sqrt{-p})=\left(\frac{a}{p}\right)\sqrt{-p}$, если $p\equiv 3 \bmod 4$.

Если k не является простым, для нахождения действия автоморфизма σ_a можно применить китайскую теорему об остатках [12, с. 43]. Пусть $k=k_1\cdot\ldots\cdot k_s$, где k_i — различные простые числа, $M_i=k/k_i$. Тогда

$$g(a,k)=g(aM_1,k_1)\cdot\ldots\cdot g(aM_s,k_s).$$

По теореме 1 имеем

$$g(a,k)=\left(\frac{aM_1}{k_1}\right)\cdot\ldots\cdot\left(\frac{aM_s}{k_s}\right)g(1,k_1)\cdot\ldots\cdot g(1,k_s).$$

Таким образом, так как $\sigma_a(g(1,k))=g(a,k)$, нахождение действия автоморфизма σ_a на $g(1,k)$ в случае, когда k — не простое свободное от квадратов (хотя утверждение верно и в случае, если k_i взаимно просты), сводится к определению его действия на все $g(1,k_1),\ldots,g(1,k_s)$. Отсюда для $(a,k)=1$ следует: $\sigma_a(\sqrt{k})=\left(\frac{a}{k}\right)\sqrt{k}$, если $k\equiv 1 \bmod 4$, и $\sigma_a(\sqrt{-k})=\left(\frac{a}{k}\right)\sqrt{-k}$, если $k\equiv 3 \bmod 4$.

Рассмотрим поле $K=\mathbb{Q}(\sqrt{d})$, где $d\equiv 2 \bmod 8$ и $d=2p$, p — простое. Тогда $g(1,d)$ имеет следующий вид:

$$g(1,d)=g(1,8)\,g(1,p)=\sqrt{2}\cdot\sqrt{p}=\sqrt{2p}=\sqrt{d}.$$

1.3. Вложения числовых полей в круговые

По теореме Кронекера — Вебера [2, Chapter 14] любое абелево расширение K поля $\mathbb{Q}$ вкладывается в $\mathbb{Q}(\zeta_f)$ для некоторого $f\in\mathbb{Z}$, $f>1$. Кондуктором K называется минимальное такое f [13, с. 75]. Тогда если f — кондуктор абелева числового поля K, то для положительного целого r справедливо: $K\cap\mathbb{Q}(\zeta_r)=K\cap\mathbb{Q}(\zeta_f)\cap\mathbb{Q}(\zeta_r)=K\cap\mathbb{Q}(\zeta_{(f,r)})$.

Мультиквадратичное поле является абелевым расширением поля рациональных чисел. Рассмотрим, как определяется кондуктор для мультиквадратичного поля K в соответствии с [14, с. 159], [8, с. 140] и [15, с. 112]. Обозначим

$$d'_i = \begin{cases} |d_i|, & d_i \equiv 1 \bmod 4, \\ 4|d_i|, & d_i \equiv 2, 3 \bmod 4. \end{cases} \tag{1}$$

Тогда $\mathbb{Q}(\sqrt{d_1}) \subset \mathbb{Q}(\zeta_{d'_1})$, $\mathbb{Q}(\sqrt{d_2}) \subset \mathbb{Q}(\zeta_{d'_2})$, $\ldots$, $\mathbb{Q}(\sqrt{d_n}) \subset \mathbb{Q}(\zeta_{d'_n})$. Таким образом, $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \subset \mathbb{Q}(\zeta_{d'_1}) \ldots \mathbb{Q}(\zeta_{d'_n}) = \mathbb{Q}(\zeta_{d'_1 \cdot \ldots \cdot d'_n}) = \mathbb{Q}(\zeta_{\text{НОК}(d'_1, \ldots, d'_n)})$ и соответственно кондуктор K равен $\text{НОК}(d'_1, \ldots, d'_n)$.

Рассмотрим теперь отображение $\mathsf{res}_{\mathbb{Q}(\zeta_r)/K \cap \mathbb{Q}(\zeta_r)}$. Возникают два вопроса: каким образом происходит сужение автоморфизмов кругового поля $\mathbb{Q}(\zeta_r)$ на поле $K \cap \mathbb{Q}(\zeta_r)$ и что есть пересечение $K \cap \mathbb{Q}(\zeta_r)$?

Ответим на первый вопрос, определив сужение автоморфизмов кругового поля $\mathbb{Q}(\zeta_r)$ сначала на круговые подполя, а затем с помощью гауссовых сумм — на мультиквадратичные числовые поля. Рассмотрим случай, когда $r = p \cdot q$, где $p, q > 0$ — взаимно простые. Произвольным образом выберем a, взаимно простое с p и q. Тогда автоморфизм $\sigma_a : \zeta_{pq} \mapsto \zeta_{pq}^a$ поля $\mathbb{Q}(\zeta_{pq})$ можно связать с действием автоморфизмов полей $\mathbb{Q}(\zeta_p)$ и $\mathbb{Q}(\zeta_q)$ на элементы $\sqrt{\pm p}$ и $\sqrt{\pm q}$ следующим образом:

$$\sigma_a(g(1, pq)) = g(a, pq) = \left(\frac{aq}{p}\right) g(1, p) \left(\frac{ap}{q}\right) g(1, q) = \begin{cases} \sigma_{aq}(\sqrt{p})\, \sigma_{ap}(\sqrt{q}), & p, q \equiv 1, 1 \bmod 4, \\ \sigma_{aq}(\sqrt{p})\, \sigma_{ap}(\sqrt{-q}), & p, q \equiv 1, 3 \bmod 4, \\ \sigma_{aq}(\sqrt{-p})\, \sigma_{ap}(\sqrt{q}), & p, q \equiv 3, 1 \bmod 4, \\ \sigma_{aq}(\sqrt{-p})\, \sigma_{ap}(\sqrt{-q}), & p, q \equiv 3, 3 \bmod 4. \end{cases}$$

Здесь индекс aq в случае $\sigma_{aq}(\sqrt{\pm p})$ рассматривается по модулю p, а индекс ap в случае $\sigma_{ap}(\sqrt{\pm q})$ — по модулю q.

Ответ на второй вопрос даёт следующая лемма.

Лемма 1. Пусть $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$ — мультиквадратичное поле, такое, что $d_1, \ldots, d_n$ попарно взаимно просты и свободны от квадратов, $(i_1, \ldots, i_\ell) \in \{1, \ldots, n\}^\ell$ — набор из ℓ индексов и $d'_{i_1}, \ldots, d'_{i_\ell}$ определены по формуле (1). Тогда

$$K \cap \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}}) = \mathbb{Q}(\sqrt{d_{i_1}}, \sqrt{d_{i_2}}, \ldots, \sqrt{d_{i_\ell}}).$$

Доказательство. Заметим, что $K_0 = \mathbb{Q}(\sqrt{d_{i_1}}, \sqrt{d_{i_2}}, \ldots, \sqrt{d_{i_\ell}})$ — подполе K, имеющее кондуктор $\text{НОК}(d'_{i_1}, \ldots, d'_{i_\ell})$. Поэтому оно содержится в $\mathbb{Q}(\zeta_{\text{НОК}(d'_{i_1}, \ldots, d'_{i_\ell})}) = \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}})$, т. е. имеем включение $K_0 \subset K \cap \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}})$.

Обратное включение докажем от противного.

Предположим, что найдётся $\alpha \in K \cap \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}})$, такое, что $\alpha \notin K_0$. Любое $\alpha \in K$ имеет вид $\alpha = \sum\limits_{j_1, \ldots, j_n} a_{j_1, \ldots, j_n} \sqrt{d_1}^{j_1} \cdot \ldots \cdot \sqrt{d_n}^{j_n}$. Из условия $\alpha \notin K_0$ следует, что в сумме найдётся слагаемое $a_{j_1, \ldots, j_n} \sqrt{d_1}^{j_1} \cdot \ldots \cdot \sqrt{d_n}^{j_n}$, содержащее $\sqrt{d_k}$, $k \notin \{i_1, \ldots, i_\ell\}$, $j_k \equiv 1 \bmod 2$ и $a_{j_1, \ldots, j_k, \ldots, j_n} \neq 0$. При этом $\sqrt{d_k} \in \mathbb{Q}(\zeta_{d'_k})$. Так как все $d_1, \ldots, d_n$ попарно взаимно просты, имеем $d'_k \nmid d'_{i_1} \cdot \ldots \cdot d'_{i_n}$, поэтому $\mathbb{Q}(\zeta_{d'_k}) \not\subset \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}})$. Отсюда следует, что $\alpha \notin \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}})$. Получили противоречие, значит, $K \cap \mathbb{Q}(\zeta_{d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}}) \subset K_0$. ■

Лемма 2. Пусть $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$ — мультиквадратичное поле, такое, что $d_1, \ldots, d_n$ попарно взаимно просты и свободны от квадратов, $f = d'_1 \cdot \ldots \cdot d'_n$ —

кондуктор поля K, где d'_i заданы в (1), и r — целое положительное число, такое, что $r|f$. Тогда r можно представить в виде $r = t \cdot d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}$, где $\{i_1, \ldots, i_\ell\}$ — множество всех индексов i, таких, что $d'_i \mid r$ и t — положительное целое. Кроме того,

$$K \cap \mathbb{Q}(\zeta_r) = \mathbb{Q}(\sqrt{d_{i_1}}, \ldots, \sqrt{d_{i_\ell}}).$$

Доказательство. Так как $d_1, \ldots, d_n$ взаимно просты, то $f = \text{НОК}(d'_1, \ldots, d'_n) = d'_1 \cdot \ldots \cdot d'_n$. Отсюда следует представление r.

Пусть $K_0 = \mathbb{Q}(\sqrt{d_{j_1}}, \ldots, \sqrt{d_{j_{n-\ell}}})$, где $j_1, \ldots, j_{n-\ell} \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_\ell\}$, и $K_1 = \mathbb{Q}(\sqrt{d_{i_1}}, \ldots, \sqrt{d_{i_\ell}})$. Тогда $K = K_0K_1 = K_0(\sqrt{d_{i_1}}, \ldots, \sqrt{d_{i_\ell}}) = K_1(\sqrt{d_{j_1}}, \ldots, \sqrt{d_{j_{n-\ell}}})$ — композит полей K_0 и K_1.

Покажем, что $K_0 \cap \mathbb{Q}(\zeta_r) = \mathbb{Q}$. Если $i \notin \{i_1, \ldots, i_\ell\}$, то $\sqrt{d_i} \notin \mathbb{Q}(\zeta_r)$, так как кондуктор d'_i поля $\mathbb{Q}(\sqrt{d_i})$ не делит r и соответственно $\mathbb{Q}(\sqrt{d_i}) \subseteq \mathbb{Q}(\zeta_{d'_i}) \not\subseteq \mathbb{Q}(\zeta_r)$. Кроме того, кондукторы полей, образованных произведениями элементов $\sqrt{d_i}$, также не делят r, поэтому все такие произведения не лежат в $\mathbb{Q}(\zeta_r)$. Так как всевозможные произведения различных $\sqrt{d_i}$ порождают поле K_0 как $\mathbb{Q}$-векторное пространство [27, Th. 2.1], получаем $K_0 \cap \mathbb{Q}(\zeta_r) = \mathbb{Q}$.

Аналогично можно показать, что $\mathbb{Q}(\sqrt{d_i}) \subseteq \mathbb{Q}(\zeta_r)$ для $i \in \{i_1, \ldots, i_\ell\}$ и, как следствие, $K_1 \subseteq K \cap \mathbb{Q}(\zeta_r)$.

Таким образом, пересечение $K \cap \mathbb{Q}(\zeta_r)$ содержит поле K_1 с кондуктором $d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}$ и не содержит поле K_0. Так как кондукторы всех подполей $\mathbb{Q}(\zeta_r)$ должны делить r, остаётся рассмотреть только поля с кондуктором — делителем t.

Рассмотрим мультиквадратичные поля с кондуктором t_0 — делителем t. Отметим, что число t делит кондуктор $d'_{j_1} \cdot \ldots \cdot d'_{j_{n-\ell}}$ поля K_0, причём $d'_{j_k} \nmid t$. Все мультиквадратичные поля с кондуктором $t_0 \mid t$ содержатся в $\mathbb{Q}(\zeta_t) \subseteq \mathbb{Q}(\zeta_r)$. Из условия $K_0 \cap \mathbb{Q}(\zeta_r) = \mathbb{Q}$ следует, что $K_0 \cap \mathbb{Q}(\zeta_t) = \mathbb{Q}$. Из того, что $(t, d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}) = 1$, имеем $K_1 \cap \mathbb{Q}(\zeta_t) = \mathbb{Q}$.

Так как K — композит полей K_0 и K_1, получаем $K \cap \mathbb{Q}(\zeta_r) = K_1$. ■

1.4. Расширенное определение идеала Штикельбергера

Прежде чем приступить к детальному описанию вычисления элементов Штикельбергера, дадим альтернативные определения элементу и идеалу Штикельбергера, которые, во-первых, упростят вычисления, а во-вторых, позволят доказать их корректность.

Определение 5 [8]. Идеалом Штикельбергера поля K с кондуктором f называется идеал вида $I = I' \cap \mathbb{Z}[G_K]$, где

$$I' = \{\theta'_r(\alpha) : r|f, \alpha \in \mathbb{Z}, \alpha < 0\} \cup \left\{\frac{1}{2}N_K\right\}.$$

Пусть $a \in \mathbb{Z}$, $a \geqslant 1$ и $(a, fr) = 1$, где f — кондуктор числового поля K. Как и прежде, обозначим через $\sigma_a \in G_{\mathbb{Q}(\zeta_{fr})}$ автоморфизм, ставящий в соответствие корню из единицы его a-ю степень. Тогда по определению

$$\theta'_r(a\alpha) = \left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_r)/K\cap\mathbb{Q}(\zeta_r)}\right)(\theta_r(a\alpha)).$$

С другой стороны, исходя из определений отображений res и σ_a, можно записать $\theta_r(a\alpha)$ следующим образом:

$$\theta_r(a\alpha) = \mathsf{res}_{\mathbb{Q}(\zeta_{rf})/\mathbb{Q}(\zeta_r)}\sigma_a(\theta_r(\alpha)).$$

Окончательно имеем

$$\begin{aligned}\theta_r^{'}(a\alpha) &= \left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_r)/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_{rf})/\mathbb{Q}(\zeta_r)}\sigma_a\right)(\theta_r(\alpha)) = \\ &= \left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_{rf})/K\cap\mathbb{Q}(\zeta_r)}\sigma_a\right)(\theta_r(\alpha)) = \\ &= \left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_{rf})/K\cap\mathbb{Q}(\zeta_r)}\right)(\sigma_a)\left(\mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)} \circ \mathsf{res}_{\mathbb{Q}(\zeta_{rf})/K\cap\mathbb{Q}(\zeta_r)}\right)(\theta_r(\alpha)) = \\ &= \sigma\theta_r^{'}(\alpha),\end{aligned}$$

где σ — автоморфизм поля K согласно определениям res и cor. Полагая $\alpha = -1$, получаем, что элемент Штикельбергера имеет вид $\theta_r^{'}(-a) = \sigma\theta_r^{'}(-1)$, где $\sigma \in G_K$. Соответственно можно переписать определение идеала Штикельбергера:

Определение 6. Идеалом Штикельбергера числового поля K с кондуктором f называется идеал вида $I = I' \cap \mathbb{Z}[G_K]$, где

$$I' = \{\sigma \cdot \theta_r^{'}(-1) :\ r|f, \sigma \in G_K\} \cup \left\{\frac{1}{2}N_K\right\}.$$

В случае мультиквадратичного поля K и некоторых дополнительных ограничений определение можно упростить, как показано в лемме 3.

Лемма 3. Пусть $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ — мультиквадратичное поле, такое, что d_i свободны от квадратов и попарно взаимно просты. Тогда идеал Штикельбергера поля K имеет вид $I = I' \cap \mathbb{Z}[G_K]$, где

$$I' = \{\sigma \cdot \theta_r^{'}(-1) : r = d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}, (i_1, \ldots, i_\ell) \subseteq \{1, \ldots n\}, \sigma \in G_K\} \cup \left\{\frac{1}{2}N_K\right\}.$$

Доказательство. Согласно определению 6, идеал I' состоит из элементов Штикельбергера, соответствующих всем делителям r кондуктора f. Пусть $r = td_{i_1} \cdot \ldots \cdot d_{i_\ell}$, как в лемме 2, тогда достаточно показать, что $\theta'_r(-1) = c\theta'_{r/t}(-1)$ для некоторого $c \in \mathbb{Q}$. В конце доказательства получим $c = \varphi(t)$. Имеем

$$\begin{aligned}\theta'_r(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_r)}\mathsf{res}_{\mathbb{Q}(\zeta_r)/K\cap\mathbb{Q}(\zeta_r)}(\theta_r(-1)), \\ \theta'_{r/t}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{r/t})}\mathsf{res}_{\mathbb{Q}(\zeta_{r/t})/K\cap\mathbb{Q}(\zeta_{r/t})}(\theta_{r/t}(-1)).\end{aligned}$$

Применяя лемму 2, получаем

$$\begin{aligned}\theta'_r(-1) &= \mathsf{cor}_{K/\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})}\mathsf{res}_{\mathbb{Q}(\zeta_r)/\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})}(\theta_r(-1)), \\ \theta'_{r/t}(-1) &= \mathsf{cor}_{K/\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})}\mathsf{res}_{\mathbb{Q}(\zeta_{r/t})/\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})}(\theta_{r/t}(-1)).\end{aligned}$$

Рассмотрим элемент

$$\theta_r(-1) = \sum_{(a,r)=1} \left\langle \frac{a}{r} \right\rangle \sigma_a^{-1} \in \mathbb{Q}[G_{\mathbb{Q}(\zeta_r)}].$$

Группа Галуа $G_{\mathbb{Q}(\zeta_r)}$ изоморфна $(\mathbb{Z}/r\mathbb{Z})^\times$, где $a \bmod r$ соответствует автоморфизму $\sigma_a : \zeta_r \mapsto \zeta_r^a$ [2, Th. 2.5]. Так как $(t, r/t) = 1$, по китайской теореме об остатках имеем изморфизм $(\mathbb{Z}/r\mathbb{Z})^\times \simeq (\mathbb{Z}/t\mathbb{Z})^\times \oplus \left(\mathbb{Z}\Big/\frac{r}{t}\mathbb{Z}\right)^\times$. Поэтому

$$G_{\mathbb{Q}(\zeta_r)} \simeq \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_{r/t})) \oplus \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_t)),$$

и любой автоморфизм $\sigma \in G_{\mathbb{Q}(\zeta_r)}$ можно единственным образом представить в виде $\sigma = \rho \circ \tau$, где $\rho \in \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_{r/t}))$ и $\tau \in \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_t))$.

В явном виде изоморфизм записывается следующим образом. По китайской теореме об остатках имеем $a = b'\dfrac{r}{t}\left(\left(\dfrac{r}{t}\right)^{-1} \bmod t\right) + c't\left(t^{-1} \bmod \dfrac{r}{t}\right)$ для некоторых b', c', таких, что $(b', t) = 1$, $b' \in (\mathbb{Z}/t\mathbb{Z})^\times$ и $(c', r/t) = 1$, $c' \in \left(\mathbb{Z}\Big/\dfrac{r}{t}\mathbb{Z}\right)^\times$. Тогда

$$\sigma_a : \zeta_r \mapsto \zeta_r^a = \zeta_t^{b'((r/t)^{-1} \bmod t)}\zeta_{r/t}^{c'(t^{-1} \bmod (r/t))},$$

где мы используем равенства $\zeta_r^{r/t} = \zeta_t$ и $\zeta_r^t = \zeta_{r/t}$. Заметим, что поля $\mathbb{Q}\left(\zeta_t^{((r/t)^{-1} \bmod t)}\right)$ и $\mathbb{Q}\left(\zeta_{r/t}^{(t^{-1} \bmod (r/t))}\right)$ изоморфны полям $\mathbb{Q}(\zeta_t)$ и $\mathbb{Q}(\zeta_{r/t})$, где изоморфизмы задаются отображениями $\zeta_t \mapsto \zeta_t^{r/t}$ и $\zeta_{r/t} \mapsto \zeta_{r/t}^t$. Поэтому для каждой пары b', c' всегда существуют единственные $b \in (\mathbb{Z}/t\mathbb{Z})^\times$, $c \in \left(\mathbb{Z}\Big/\dfrac{r}{t}\mathbb{Z}\right)^\times$, такие, что $b' = br/t \bmod r$, $(b, t) = 1$, и $c' = ct \bmod r$, $(c, r/t) = 1$. Тогда можно записать σ_a в виде

$$\begin{aligned}\sigma_a : \zeta_r \mapsto \zeta_r^a &= \zeta_r^{b(r/t)^2((r/t)^{-1} \bmod t)+ct^2(t^{-1} \bmod (r/t))} = \\ &= \zeta_t^{br/t((r/t)^{-1} \bmod t)}\zeta_{r/t}^{ct(t^{-1} \bmod (r/t))} = \zeta_t^b\zeta_{r/t}^c = \rho_b \circ \tau_c,\end{aligned}$$

где $\rho_b \in \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_{r/t}))$ и $\tau_c \in \mathrm{Gal}(\mathbb{Q}(\zeta_r)/\mathbb{Q}(\zeta_t))$ задаются следующим образом:

$$\begin{aligned}\rho_b : \zeta_r \mapsto \zeta_r^{b(r/t)^2((r/t)^{-1} \bmod t)+t^2(t^{-1} \bmod (r/t))} &= \zeta_r^{b(r/t)^2((r/t)^{-1} \bmod t)}\zeta_r^{t^2(t^{-1} \bmod (r/t)} = \zeta_t^b\zeta_{r/t}, \\ \tau_c : \zeta_r \mapsto \zeta_r^{(r/t)^2((r/t)^{-1} \bmod t)+ct^2(t^{-1} \bmod (r/t)} &= \zeta_r^{(r/t)^2((r/t)^{-1} \bmod t)}\zeta_r^{ct^2(t^{-1} \bmod (r/t))} = \zeta_t\zeta_{r/t}^c.\end{aligned}$$

Теперь, получив автоморфизмы в явном виде, выразим через них элемент $\theta_r(-1)$:

$$\begin{aligned}\theta_r(-1) &= \sum_{(a,r)=1}\left\langle\frac{a}{r}\right\rangle\sigma_a^{-1} = \\ &= \sum_{(c,r/t)=1}\left(\sum_{(b,t)=1}\left\langle\frac{b(r/t)^2\big((r/t)^{-1} \bmod t\big) + ct^2\big(t^{-1} \bmod (r/t)\big)}{r}\right\rangle\rho_b^{-1}\right)\tau_c^{-1}.\end{aligned}$$

Так как ограничение автоморфизма ρ_b^{-1} на поле $K_1 = \mathbb{Q}(\sqrt{d_{i_1}}, \ldots, \sqrt{d_{i_\ell}})$ равно id, получаем

$$\mathsf{res}_{\mathbb{Q}(\zeta_r)/K_1}(\theta_r(-1)) = \sum_{(c,r/t)=1}\left(\sum_{(b,t)=1}\left\langle\frac{b(r/t)^2\big((r/t)^{-1} \bmod t\big) + ct^2\big(t^{-1} \bmod (r/t)\big)}{r}\right\rangle\right)\tau_c^{-1}.$$

Поскольку мы перешли в подгруппу группы Галуа $G_{\mathbb{Q}(\zeta_r)}$, образованную автоморфизмом τ_c, значение множителя перед τ_c можно рассматривать по модулю r/t, тогда

$$\begin{aligned}\mathsf{res}_{\mathbb{Q}(\zeta_r)/K_1}(\theta_r(-1)) &= \sum_{(c,r/t)=1}\left(\sum_{(b,t)=1}\left\langle\frac{ct^2(t^{-1} \bmod (r/t))}{r}\right\rangle\right)\tau_c^{-1} = \sum_{(c,r/t)=1}\left(\sum_{(b,t)=1}\frac{ct}{r}\right)\tau_c^{-1} = \\ &= \varphi(t)\sum_{(c,r/t)=1}(ct/r)\tau_c^{-1} = \mathsf{res}_{\mathbb{Q}(\zeta_r)/K_1}(\theta_{r/t}(-1)).\end{aligned}$$

В итоге получаем $\theta'_r(-1) = \varphi(t)\theta'_{r/t}(-1)$. ■

2. Метод вычисления элемента Штикельбергера

1) Рассмотрим $K = \mathbb{Q}(\sqrt{d})$, где $d \equiv 1 \bmod 4$, $d > 0$, d свободно от квадратов. Вычислим элемент Штикельбергера для действительного квадратичного поля. Отметим, что предвычисления проводятся в круговом поле $\mathbb{Q}(\zeta_f)$, где $f = d$ — кондуктор поля K, согласно (1). Тогда

$$\theta'_d(-1) = \mathrm{cor}_{K/K\cap\mathbb{Q}(\zeta_d)}\left(\mathrm{res}_{\mathbb{Q}(\zeta_d)/K\cap\mathbb{Q}(\zeta_d)}(\theta_d(-1))\right),$$

где $\theta_d(-1) = \sum\limits_{(a,d)=1}\left\langle\dfrac{a}{d}\right\rangle\sigma_a^{-1}$ и $\sigma_a^{-1} \in \mathrm{Gal}(\mathbb{Q}(\zeta_d))$. Запишем более подробно $\theta_d(-1)$:

$$\theta_d(-1) = \sum_{(a,d)=1}\left\langle\frac{a}{d}\right\rangle\sigma_a^{-1} = \frac{1}{d}\sigma_1^{-1} + \frac{2}{d}\sigma_2^{-1} + \ldots + \frac{d-2}{d}\sigma_{d-2}^{-1} + \frac{d-1}{d}\sigma_{d-1}^{-1} =$$
$$= \frac{1}{d}\sigma_1 + \frac{2}{d}\sigma_{(1/2) \bmod d} + \ldots + \frac{d-2}{d}\sigma_{(1/(d-2)) \bmod d} + \frac{d-1}{d}\sigma_{(1/(d-1)) \bmod d} = \ldots,$$

или, переобозначив, получаем

$$\ldots = \frac{1}{d}\sigma_1 + \frac{2}{d}\sigma_b + \ldots + \frac{d-2}{d}\sigma_{-b \bmod d} + \frac{d-1}{d}\sigma_{-1 \bmod d}. \tag{2}$$

Важно отметить, что при дробях с противоположными по модулю d числителями фигурируют автоморфизмы с противоположными по модулю d индексами. Напомним, что, вычисляя отображение res, мы, по сути, вычисляем символ Кронекера — Якоби $\left(\dfrac{i}{d}\right)$, сопоставляя автоморфизму кругового поля σ_i автоморфизм квадратичного поля id в случае $\left(\dfrac{i}{d}\right) = 1$, или автоморфизм $\sigma : \sqrt{d} \mapsto -\sqrt{d}$, если $\left(\dfrac{i}{d}\right) = -1$. Рассмотрим пары $\dfrac{1}{d}\sigma_1$ и $\dfrac{d-1}{d}\sigma_{-1 \bmod d}$, $\dfrac{2}{d}\sigma_b$ и $\dfrac{d-2}{d}\sigma_{-b \bmod d}$ и т. д. из разложения (2):

$$\left(\frac{1}{d}\right) = 1 \text{ и } \left(\frac{-1}{d}\right) = 1, \text{ поскольку } d \equiv 1 \bmod 4.$$

Это означает, что σ_1 и $\sigma_{-1 \bmod d}$ соответствуют одному и тому же автоморфизму поля K и $\dfrac{1}{d} + \dfrac{d-1}{d} = 1$. Аналогично,

$$\left(\frac{b}{d}\right) \equiv b^{(d-1)/2} \bmod d \text{ и } \left(\frac{-b}{d}\right) = \left(\frac{-1}{d}\right)\left(\frac{b}{d}\right) \equiv b^{(d-1)/2} \bmod d, \text{ поскольку } d \equiv 1 \bmod 4.$$

Это снова означает, что σ_b и $\sigma_{-b \bmod d}$ соответствуют одному и тому же автоморфизму поля K и $\dfrac{2}{d} + \dfrac{d-2}{d} = 1$. То же и для оставшихся пар. Тогда

$$\theta_d(-1) = \frac{1}{2}\sum_{a \bmod d}\sigma_a,$$

согласно определению I' идеала Штикельбергера числового поля K и по определению нормы получаем $\theta_d(-1) = \dfrac{1}{2}N_K$.

Отметим, что

$$\sum_{\substack{i=1\\ i\in\mathbb{Z}_d^\times}}^{d-1}\chi(i)=0,$$

где $\chi(i)=\left(\dfrac{i}{d}\right)$ — характер Дирихле по модулю d, определяемый символом Якоби $\left(\dfrac{i}{d}\right)$. Поскольку число характеров по модулю d равно $\varphi(d)$ и значение $\varphi(d)$ чётно, число символов Якоби, равных -1, совпадает с числом символов Якоби, равных 1, и соответственно равно $\varphi(d)/2$. В итоге получаем

$$\theta'_d(-1)=\frac{\varphi(d)}{4}(id+\sigma)=\frac{\varphi(d)}{4}N_{K/\mathbb{Q}}.$$

Для составного $d=p_1\cdot p_2$, где p_1, p_2 — простые числа, в силу мультипликативности функции Эйлера имеем $4|(p_1-1)(p_2-1)=\varphi(p_1)\varphi(p_2)=\varphi(d)$. Это условие распространяется и на большее число множителей d. Тогда $\varphi(d)/4\in\mathbb{Z}$.

2) Рассмотрим теперь $K=\mathbb{Q}(\sqrt{-d})$, где $d>0$ и $-d\equiv 1 \bmod 4$. Запишем элемент Штикельбергера

$$\theta_d(-1)=\sum_{(a,d)=1}\left\langle\frac{a}{d}\right\rangle\sigma_a^{-1}=\sum_{a\in(\mathbb{Z}_d)^\times}\frac{a}{d}\sigma_a^{-1}=u\cdot id+v\cdot\sigma.$$

Докажем, что $u,v\in\mathbb{Z}$. Согласно определению отображения res,

$$\sigma_a^{-1}|_K=id\Leftrightarrow\left(\frac{a}{d}\right)=1,\quad \sigma_a^{-1}|_K=\sigma\Leftrightarrow\left(\frac{a}{d}\right)=-1.$$

Тогда

$$u=\sum_{\substack{a=1\\ \left(\frac{a}{d}\right)=1}}^{d}\frac{a}{d},\quad v=\sum_{\substack{a=1\\ \left(\frac{a}{d}\right)=-1}}^{d}\frac{a}{d}. \tag{3}$$

Как уже было отмечено, для элементов из $(\mathbb{Z}_d)^\times$ число символов Якоби, равных -1, равно числу символов Якоби, равных 1. Имеем

$$\sum_{\left(\frac{b}{d}\right)=1}b=\sum_{\substack{k=1\\ \left(\frac{\alpha}{d}\right)=1}}^{\varphi(d)/2}\alpha^k=\alpha^2+\alpha^3+\ldots+\alpha^{\varphi(d)/2+1}=$$
$$=\alpha^2(1+\alpha+\ldots+\alpha^{\varphi(d)/2-1})=\alpha^2\sum_{\left(\frac{b}{d}\right)=1}b.$$

Заметим, что такой $\alpha\in\mathbb{Z}_d^\times$ всегда существует, так как для $d=p_1\cdot p_2\cdot\ldots\cdot p_s$ справедливо $\mathbb{Z}_d^\times\cong\mathbb{Z}_{p_1}^\times\oplus\mathbb{Z}_{p_2}^\times\oplus\ldots\oplus\mathbb{Z}_{p_s}^\times$ — циклическая в силу взаимной попарной простоты $p_1,p_2,\ldots,p_s$. Следовательно, поскольку $\alpha^2\neq 1$, то

$$\sum_{\left(\frac{b}{d}\right)=1}b\equiv 0 \bmod d.$$

Соответственно $u=\sum\limits_{\left(\frac{a}{d}\right)=1}\dfrac{a}{d}\in\mathbb{Z}$. Отсюда следует, что

$$d(u+v)=\sum_{\substack{a=1\\ (a,d)=1}}^{d}a=\frac{\varphi(d)}{2}d;$$

учитывая, что $\varphi(d)/2$ — целое в силу разложения $d = p_1 \cdot \ldots \cdot p_n$, где p_i — различные простые, получаем

$$v = \varphi(d)/2 - u \in \mathbb{Z}.$$

Отметим также, что применение отображения cor не изменяет целостности коэффициентов при автоморфизмах.

Теперь рассмотрим случай, когда d простое. Покажем, что тогда вычисление значений в (3) можно значительно ускорить.

При $d \equiv 1 \bmod 4$ значения u и v легко получить, не вычисляя суммы уравнений (3). Запишем $d = 4k + 1$, $k \in \mathbb{Z}$. Если $a \in \mathbb{Z}_d^\times$ — квадратичный вычет, то $(-a)$ также является квадратичным вычетом в $\mathbb{Z}_d^\times$, и существует всего $(d-1)/2$ квадратичных вычетов. Получаем $(d-1)/4$ пар вычетов, сумма элементов каждый пары равна d. Отсюда сумма всех квадратичных вычетов равна $d(d-1)/4 = d \cdot k$, а значит, $u = k$. Следовательно, $v = \varphi(d)/2 - u = \varphi(d)/2 - k$.

Очевидно, эти рассуждения не переносятся на случай $d = 4k + 3$, $k \in \mathbb{Z}$. Однако и здесь можно ускорить вычисления по формулам (3), используя результат [16]: если $d \equiv 3 \bmod 8$, то $v = dv'$, где v' — сумма всех квадратичных невычетов по модулю d, меньших $d/2$, а если $d \equiv 7 \bmod 8$, то $v = \dfrac{1}{3}\left(dv' + \binom{d}{2}\right)$. Таким образом, вычисления v ускоряются в 2 раза.

3) Рассмотрим $K = \mathbb{Q}(\sqrt{d})$, где $d \equiv \pm 2 \bmod 8$ и d может быть как положительным, так и отрицательным, но свободным от квадратов. В зависимости от знака получаем вид элемента Штикельбергера, соответствующий одному из предыдущих случаев. Очевидно, что $|d|$ имеет вид $|2p|$, где p нечётное. Кроме того, $\sqrt{2}$ соответствует ζ_8, а $\sqrt{p}$ соответствует ζ_p. Согласно теории круговых полей, композит $\mathbb{Q}(\zeta_8)\mathbb{Q}(\zeta_p)$ есть $\mathbb{Q}(\zeta_{\text{НОК}(8,p)}) = \mathbb{Q}(\zeta_{8p}) = \mathbb{Q}(\zeta_{4d})$, откуда кондуктор поля равен $4|d| = 8|p|$. Тогда

$$\theta_{4|d|}(-1) = \frac{1}{4|d|} \sum_{k=1}^{4|d|-1} k \cdot \sigma_k^{-1}(\zeta_{4|d|}),$$

где $(k, 4|d|) = 1$. При этом

$$\sigma_k^{-1}(\zeta_{4|d|}) = \sigma_{k(|d|/2) \bmod 8}(\zeta_8)\sigma_{k\cdot 8 \bmod (|d|/2)}\left(\zeta_{|d|/2}\right).$$

Если $\left(\dfrac{8}{k(|d|/2) \bmod 8}\right)\left(\dfrac{k \cdot 8 \bmod (|d|/2)}{|d|/2}\right) = 1$, то действие автоморфизма σ_k^{-1} кругового поля $\mathbb{Q}(\zeta_{4|d|})$ на $\zeta_{4|d|}$ соответствует действию автоморфизма id числового поля K на элемент $\sqrt{d}$. Если $\left(\dfrac{8}{k(|d|/2) \bmod 8}\right)\left(\dfrac{k \cdot 8 \bmod (|d|/2)}{|d|/2}\right) = -1$, то действие автоморфизма σ_k^{-1} кругового поля $\mathbb{Q}(\zeta_{4|d|})$ на $\zeta_{4|d|}$ соответствует действию автоморфизма σ числового поля K на элемент $\sqrt{d}$.

Обобщение на мультиквадратичный случай аналогично предыдущим случаям, поскольку $d_i \equiv \pm 2 \bmod 8$ в записи $\mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$ фигурирует в одной позиции.

Замечание 1. Рассмотрим поле $K = \mathbb{Q}(\sqrt{2})$. Из рассуждений в п. 3 следует, что кондуктор данного поля равен 8. Тогда

$$\theta_8(-1) = \frac{1}{8}\sum_{k=1}^{7} k\,\sigma_k^{-1}(\zeta_8) = \frac{1}{8}\left(\sigma_1^{-1}(\zeta_8) + 3\,\sigma_3^{-1}(\zeta_8) + 5\,\sigma_5^{-1}(\zeta_8) + 7\,\sigma_7^{-1}(\zeta_8)\right) =$$

$$= \frac{1}{8}\left(\sigma_1(\zeta_8) + 3\,\sigma_3(\zeta_8) + 5\,\sigma_5(\zeta_8) + 7\,\sigma_7(\zeta_8)\right).$$

Вычислим символы Кронекера — Якоби. Поскольку $\left(\frac{8}{1}\right) = 1$, действие автоморфизма σ_1^{-1} на ζ_8 соответствует действию автоморфизма id на элемент $\sqrt{2}$; $\left(\frac{8}{3}\right) = -1$, поэтому действие автоморфизма σ_3^{-1} на ζ_8 соответствует действию автоморфизма σ на элемент $\sqrt{2}$. Аналогично рассуждаем для $\left(\frac{8}{5}\right) = -1$, $\left(\frac{8}{7}\right) = 1$. Таким образом, получаем

$$\theta_8(-1) = \frac{1}{8}\left(id + 3\,\sigma + 5\,\sigma + 7\,id\right) = id + \sigma.$$

4) Обобщим рассмотренные случаи на произвольное биквадратичное поле. Пусть $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2})$, где $d_1 \equiv 2 \bmod 8$ и $d_1 < 0$, $d_2 \equiv 1 \bmod 4$ и $d_2 > 0$, d_1, d_2 свободны от квадратов. В соответствии с п. 1 и 3 кондуктор числового поля K равен $4|d_1|d_2$. Тогда

$$\theta'_{4|d_1|d_2}(-1) = \mathrm{cor}_{K/K\cap\mathbb{Q}(\zeta_{4|d_1|d_2})}\left(\mathrm{res}_{\mathbb{Q}(\zeta_{4|d_1|d_2})/K\cap\mathbb{Q}(\zeta_{4|d_1|d_2})}(\theta_{4|d_1|d_2}(-1))\right),$$

где

$$\theta_{4|d_1|d_2}(-1) = \frac{1}{4|d_1|d_2}\sum_{k=1}^{4|d_1|d_2-1} k\,\sigma_k^{-1}(\zeta_{4|d_1|d_2}), \quad (k, 4|d_1|d_2) = 1.$$

Рассмотрим отдельно $\sigma_k^{-1}(\zeta_{4|d_1|d_2})$:

$$\sigma_k^{-1}(\zeta_{4|d_1|d_2}) = \sigma^{-1}_{k\cdot d_2 \bmod 4|d_1|}(\zeta_{4|d_1|})\sigma^{-1}_{k\cdot 4|d_1| \bmod d_2}(\zeta_{d_2}) =$$

$$= \sigma_{(k\cdot d_2 \bmod 4|d_1|)(|d_1|/2) \bmod 8}(\zeta_8)\,\sigma_{(k\cdot d_2 \bmod 4|d_1|)\,8 \bmod (|d_1|/2))}(\zeta_{|d_1|/2})\,\sigma_{(1/(k\cdot 4|d_1| \bmod d_2)) \bmod d_2}(\zeta_{d_2}).$$

Далее действуем в соответствии с п. 1 и 3. Рассмотрим

$$\sigma_{(k\cdot d_2 \bmod 4|d_1|)(|d_1|/2) \bmod 8}(\zeta_8)\,\sigma_{(k\cdot d_2 \bmod 4|d_1|)\,8 \bmod (|d_1|/2)}(\zeta_{|d_1|/2}).$$

Если $\left(\frac{8}{(k\cdot d_2 \bmod 4|d_1|)(|d_1|/2) \bmod 8}\right)\left(\frac{(k\cdot d_2 \bmod 4|d_1|)\,8 \bmod (|d_1|/2))}{|d_1|/2}\right) = 1$, то действие автоморфизма $\sigma^{-1}_{k\cdot d_2 \bmod 4|d_1|}$ на $\zeta_{4|d_1|}$ соответствует действию автоморфизма id_1 на элемент $\sqrt{d_1}$.

Если $\left(\frac{8}{(k\cdot d_2 \bmod 4|d_1|)(|d_1|/2) \bmod 8}\right)\left(\frac{(k\cdot d_2 \bmod 4|d_1|)\,8 \bmod (|d_1|/2))}{|d_1|/2}\right) = -1$, то действие автоморфизма $\sigma^{-1}_{k\cdot d_2 \bmod 4|d_1|}$ на $\zeta_{4|d_1|}$ соответствует действию автоморфизма σ_1 на $\sqrt{d_1}$.

Рассмотрим $\sigma_{(1/(k\cdot 4|d_1| \bmod d_2)) \bmod d_2}(\zeta_{d_2})$. Вычислим символ Кронекера — Якоби $\left(\frac{(1/(k\cdot 4|d_1| \bmod d_2)) \bmod d_2}{d_2}\right)$. Если он равен 1, то действие автоморфизма $\sigma^{-1}_{k\cdot 4|d_1| \bmod d_2}$ на ζ_{d_2} соответствует действию автоморфизма id_2 на элемент $\sqrt{d_2}$. Если он равен -1, то действие автоморфизма $\sigma^{-1}_{k\cdot 4|d_1| \bmod d_2}$ на ζ_{d_2} соответствует действию автоморфизма σ_2 на элемент $\sqrt{d_2}$. В итоге получим комбинации произведений автоморфизмов $id_1, id_2, \sigma_1, \sigma_2$, где каждая соответствует одному из автоморфизмов τ_i поля K.

Аналогичным образом вычисляются $\theta_f(-1)$ для мультиквадратичных полей.

3. Алгоритм

Рассмотрим подробно алгоритм вычисления идеала Штикельбергера для мультиквадратичных полей в соответствии с описанной теорией. Заметим, что, согласно лемме 3, при взаимно простых d_i для нахождения идеала Штикельбергера достаточно ограничиться делителями кондуктора, составленными из произведений d'_i.

3.1. Вычисление действий отображений res и cor

I. Вычисление res**.** Рассмотрим алгоритм для вычисления $\mathsf{res}_{\mathbb{Q}(\zeta_f)/K\cap\mathbb{Q}(\zeta_f)}(\theta_f(-1))$, где $K=\mathbb{Q}(\sqrt{d_1},\ldots,\sqrt{d_n})$ и f — кондуктор K. Значения $\mathsf{res}_{\mathbb{Q}(\zeta_r)/K\cap\mathbb{Q}(\zeta_r)}(\theta_r(-1))$ для $r\mid f$ можно найти по тому же алгоритму, положив $f=r$, $n=\ell$ и $d_1=d_{i_1}$, …, $d_\ell=d_{i_\ell}$ для $\ell, i_1,\ldots,i_\ell$, как в лемме 2.

Если одно из $d_i\equiv\pm2 \bmod 8$, то алгоритм вычисляет

$$\mathrm{res}_{\mathbb{Q}(\zeta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n})/K\cap\mathbb{Q}(\zeta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n})}(\theta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n}(-1)),$$

где

$$\theta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n}(-1)=\frac{1}{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n}\sum_{k=1}^{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n-1}k\cdot\sigma_k^{-1}(\zeta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n})$$

и $(k,4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n)=1$.

В противном случае алгоритм вычисляет

$$\mathsf{res}_{\mathbb{Q}(\zeta_{d'_1\cdot\ldots\cdot d'_n})/K\cap\mathbb{Q}(\zeta_{d'_1\cdot\ldots\cdot d'_n})}\theta_{d'_1\cdot\ldots\cdot d'_n}(-1),$$

где

$$\theta_{d'_1\cdot\ldots\cdot d'_n}(-1)=\sum_{(a,d'_1\cdot\ldots\cdot d'_n)=1}\left\langle\frac{a}{d'_1\cdot\ldots\cdot d'_n}\right\rangle\sigma_a^{-1}.$$

Результатами вычисления $\theta_{4d'_1\cdot\ldots|d_i|\cdot\ldots\cdot d'_n}(-1)$ или $\theta_{d'_1\cdot\ldots\cdot d'_n}(-1)$ будут комбинации автоморфизмов, которые обозначим так:

$$\begin{gathered}id_1\cdot id_2\cdot\ldots\cdot id_n=id:\sqrt{d_1}+\ldots+\sqrt{d_n}\ \to\ \sqrt{d_1}+\sqrt{d_2}+\ldots+\sqrt{d_n},\\ \ldots\\ \sigma_1\cdot\ldots\cdot\sigma_{n-1}\cdot\sigma_n=\tau_m:\sqrt{d_1}+\ldots+\sqrt{d_{n-1}}+\sqrt{d_n}\ \to\ -\sqrt{d_1}-\ldots-\sqrt{d_n}.\end{gathered}\tag{4}$$

Псевдокод процедуры вычисления res представлен в алгоритме 1.

II. Вычисление cor**.** Автоморфизмы, полученные после вычисления res, являются автоморфизмами поля $K\cap\mathbb{Q}(\zeta_{d'_1\cdot\ldots\cdot d'_n})$ (или $K\cap\mathbb{Q}(\zeta_{d'_{i_1}\cdot\ldots\cdot d'_{i_\ell}})$ для $r\mid f$ и $\ell, i_1,\ldots i_\ell$, как в лемме 2). Вычисление действия отображения cor представляет собой переход от этих автоморфизмов к автоморфизмам поля K. Обозначим автоморфизмы поля K следующим образом: сопоставим действие автоморфизма ρ_i с бинарным вектором из $\mathbb{Z}_2^n$, причём если j-я координата вектора (считая слева направо) есть 1, то $\rho_i:\sqrt{d_j}\to-\sqrt{d_j}$ (например, $\rho_1:\sqrt{d_1}+\ldots+\sqrt{d_n}\to\sqrt{d_1}+\ldots-\sqrt{d_n}$).

Если $K\cap\mathbb{Q}(\zeta_{d'_{i_1}\cdot\ldots\cdot d'_{i_\ell}})=K$, то отображение cor действует тождественно (автоморфизмы τ_i совпадают с ρ_i). Отметим, что такой случай возникает при вычислении $\theta'_{d'_1\cdot\ldots\cdot d'_n}(-1)$. А как быть, если мы вычисляем, например, элемент Штикельбергера вида $\theta'_{d'_{i_1}\cdot\ldots\cdot d'_{i_\ell}}(-1)$, где $\ell<n$? Положим $K\cap\mathbb{Q}(\zeta_{d'_{i_1}\cdot\ldots\cdot d'_{i_\ell}})=\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})$. Результатом действия отображения res в этом случае будет

$$a_1\cdot id_\ell+a_2\cdot\tau_1+\ldots+a_{2^\ell-1}\cdot\tau_{2^\ell-2}+a_{2^\ell}\cdot\tau_{2^\ell-1},$$

где id_ℓ, τ_i — автоморфизмы поля $\mathbb{Q}(\sqrt{d_{i_1}},\ldots,\sqrt{d_{i_\ell}})$, $i=1,\ldots,2^\ell-1$. Нумерация автоморфизмов τ_i аналогична нумерации автоморфизмов ρ_i.

Алгоритм 1. Вычисление $\mathrm{res}_{\mathbb{Q}(\zeta_f)/K\cap\mathbb{Q}(\zeta_f)}(\theta_f(-1))$

Вход: $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$ для всех $i \in \{1, \ldots, n\}$, или $d_j \equiv \pm 2 \bmod 8$ для некоторого $j \in \{1, \ldots, n\}$; d_i свободны от квадратов и взаимно просты.

Выход: $\mathsf{res}(\theta_f(-1))$.

$f := \prod_{j=1}^{n} d'_j$. // f — кондуктор K

Для $a \in \mathbb{Z}_f^\times$:

  $\sigma_a := 1$.

  Для $j = 1, \ldots, n$:

    Если $d_j = 2 \bmod 8$, **то**

      $i_1 := a\,(|d_j|/2) \bmod 8$;

      $i_2 := a \cdot 8 \bmod (|d_j|/2)$.

      Если $\left(\frac{8}{i_1}\right)\left(\frac{i_2}{|d_j|/2}\right) = 1$, **то**

        $\sigma_a^{-1} := \sigma_a^{-1} \cdot id_j$, // id_j — тождественный в $\mathbb{Q}(\sqrt{d_j})$

      иначе

        $\sigma_a^{-1} := \sigma_a^{-1} \cdot \sigma_j$, // σ_j — сопряжение в $\mathbb{Q}(\sqrt{d_j})$

    иначе

      $t := \frac{a \cdot d'_1 \cdot \ldots \cdot d'_n}{d'_j} \bmod d'_j$, $i := \frac{1}{t} \bmod d'_j$.

      Если $\left(\frac{i}{d'_j}\right) = 1$, **то**

        $\sigma_a^{-1} := \sigma_a^{-1} \cdot id_j$, // id_j — тождественный в $\mathbb{Q}(\sqrt{d_j})$

      иначе

        $\sigma_a^{-1} := \sigma_a^{-1} \cdot \sigma_j$. // σ_j — сопряжение в $\mathbb{Q}(\sqrt{d_j})$

  $\sigma_a^{-1} := \frac{a}{f} \cdot \sigma_a^{-1}$.

$\theta := \sum_{a \in \mathbb{Z}_f^*} \sigma_a^{-1}$.

res := (заменить получившиеся комбинации автоморфизмов в каждом слагаемом θ на соответствующие τ_i согласно формулам (4)).

Вернуть res.

Далее переходим от перечисленных автоморфизмов поля $\mathbb{Q}(\sqrt{d_{i_1}}, \ldots, \sqrt{d_{i_\ell}})$ к автоморфизмам ρ_i поля K. Если ρ_i относительно элемента $\sqrt{d_{i_1}} + \ldots + \sqrt{d_{i_\ell}}$ действует как id_ℓ, то все такие автоморфизмы ρ_i участвуют в записи элемента Штикельбергера с коэффициентом a_1. Аналогично, если ρ_i относительно $\sqrt{d_{i_1}} + \ldots + \sqrt{d_{i_\ell}}$ действует как τ_1, то все такие автоморфизмы ρ_i участвуют в записи элемента Штикельбергера с коэффициентом a_2. Применяя аналогичный подход для всех остальных случаев, получаем, что в общем случае элемент Штикельбергера примет следующий вид:

$$\theta'_r(-1) = c_0 \cdot id + c_1 \cdot \rho_1 + \ldots + c_{m-1} \cdot \rho_{m-1} + c_m \cdot \rho_m,$$

где $c_i \in \mathbb{Z}$ для $i = 0, \ldots, m$ и $m = 2^n - 1$.

Общее количество элементов Штикельбергера в поле K равно $2^n - 1$. Количество различных комбинаций зависит от количества разных коэффициентов в элементе

Штикельбергера. Запишем все различные комбинации в множество I' (алгоритм 2) мощности $\#I'$.

Алгоритм 2. Вычисление идеала Штикельбергера

Вход: $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$ для всех $i \in \{1, \ldots, n\}$, или некоторый $d_j \equiv \pm 2 \bmod 8$, $j \in \{1, \ldots, n\}$; d_i свободны от квадратов и взаимно просты.

Выход: Выход: $I = I' \cap \mathbb{Z}[\mathrm{Gal}(K/\mathbb{Q})]$.

1: $A :=$ массив, индексирующий подполя K вида $\mathbb{Q}(\{\sqrt{d_i}\}_{i \in J})$, для всех непустых подмножеств $J \subseteq \{1, \ldots, n\}$.
2: **Для** $i = 1, \ldots, 2^n - 1$:
3: $\quad r := d'_{i_1} \cdot \ldots \cdot d'_{i_\ell}$, где $i_1, \ldots, i_\ell = A[i]$.
4: $\quad \mathsf{res}_i := \mathsf{res}_{\mathbb{Q}(\zeta_r)/\mathbb{Q}(\sqrt{d_{i_1}}, \sqrt{d_{i_2}}, \ldots, \sqrt{d_{i_\ell}})} \theta_r(-1)$; // Алгоритм 1
5: $\quad \gamma_i \leftarrow \mathsf{cor}_{K/\mathbb{Q}(\sqrt{d_{i_1}}, \sqrt{d_{i_2}}, \ldots, \sqrt{d_{i_\ell}})} \mathsf{res}_i$.
6: $I' := \varnothing$.
7: **Для** $i = 1, \ldots, 2^n - 1$:
8: $\quad$ **Для** $j = 0, \ldots, 2^n - 1$:
9: $\quad\quad t := \rho_j \cdot \gamma_i$.
10: $\quad\quad$ **Если** $t \notin I'$, **то**
11: $\quad\quad\quad I' := I' \cup \{t\}$.
12: **Вернуть** I.

3.2. Сложность алгоритма

Теорема 2. Получив на вход поле $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, где $d_i \equiv 1 \bmod 4$ для всех $i \in \{1, \ldots, n\}$, или некоторый $d_j \equiv \pm 2 \bmod 8$, $j \in \{1, \ldots, n\}$, d_i свободны от квадратов и взаимно просты, алгоритм 2 возвращает образующие идеала Штикельбергера поля K за время

$$T = \mathcal{O}(\lg \Delta_K \cdot 2^n \cdot \mathrm{poly}(n)).$$

В частности, если d_i — первые n простых чисел, сложность алгоритма 2 равна

$$T = \mathcal{O}(e^{n \log(n)} \cdot 2^{2n} \cdot n^4 \cdot \mathrm{poly}\log(n)).$$

Доказательство. Поскольку алгоритм 1 является частью алгоритма 2, рассмотрим сначала его вычислительную сложность. На вход алгоритма 1 подаётся ℓ-квадратичное поле $\mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_\ell})$ ($d_1 = d_{i_1}, \ldots, d_\ell = d_{i_\ell}$ во входных данных алгоритма) и на шаге 1 вычисляется произведение всех d'_j, $j \leqslant \ell$. Для взаимно простых d'_i имеем $\prod\limits_i d'_i = \mathcal{O}\left(\lg \Delta_K / 2^n\right)$. В случае, когда d'_i — первые из ℓ простых чисел (ℓ-е простое оценивается $\approx \ell \log(\ell)$), получаем $f = \prod d'_i = e^{\ell \log(\ell)}$. В доказательстве дальше рассматриваем именно этот случай.

Рассмотрим цикл на шаге 2. Он повторяется $\varphi(f)$ раз, и это значит, что его оценка $\mathcal{O}(f) = \mathcal{O}(e^{\ell \log(\ell)})$. В теле цикла выполняются шаги 4–10 либо 11–17. Рассмотрим сначала шаги 4–10. Оценка шага 5 не влияет на максимальную сложность. На шаге 6 производятся вычисления по модулю d_j. В худшем случае наибольшая образующая сравнима с 2 mod 8, тогда $d = \max\limits_j d'_j$. Таким образом, сложность шага 6 равна $\mathcal{O}(\log^3(d))$. Аналогично для шагов 7 и 9, где также производятся вычисления по модулю d'_j. Далее выполняется либо шаг 7, либо шаг 9, а оценки шагов 8 и 10 не влияют на максимальную сложность. Таким образом, сложность одной итерации внутреннего цикла для

случая $d_j \equiv 2 \bmod 8$ равна $\mathcal{O}(\log^3(d))$. Аналогично рассуждаем для шагов 11–17. Шаги 12, 13, 14 и 16 представляют собой вычисления по модулю d'_j. Оценка каждого из них равна $\mathcal{O}(\log^3(d))$. В цикле выполняется либо шаг 14, либо шаг 16, а шаги 15 и 17 имеют сложность $\mathcal{O}(1)$. Таким образом, сложность одной итерации внутреннего цикла для всех остальных случаев равна $\mathcal{O}(\log^3(d))$. Во внешнем цикле также выполняется шаг 18, чья сложность равна $\mathcal{O}(\log^3(f)) = \mathcal{O}(\log^3(e^{\ell \log(\ell)})) = \mathcal{O}(\ell^3 \cdot \log^3(\ell))$. Оценка шага 19 не влияет на максимальную сложность, а замена на шаге 20 имеет сложность $\mathcal{O}(2^\ell)$, поскольку мы имеем 2^ℓ автоморфизмов. Обобщая, получаем, что общая сложность алгоритма 1 равна

$$\mathcal{O}(e^{\ell \log(\ell)} \cdot \ell^4 \cdot \log^3(d) \cdot \log^3(\ell) + 2^\ell).$$

Поскольку функция $e^{\ell \log(\ell)}$ возрастает быстрее, чем 2^ℓ, итоговая сложность алгоритма 1 имеет вид

$$\mathcal{O}(e^{\ell \log(\ell)} \cdot \ell^4 \cdot \log^3(d) \cdot \log^3(\ell)).$$

Вернёмся к алгоритму 2. Ему на вход мы подаём $\mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$. Рассмотрим цикл на шаге 2. Он повторяется $2^n - 1$ раз. На шаге 4 в теле цикла вычисляется результат отображения res, а значит, используется алгоритм 1. Поскольку на входе n-квадратичное поле и в худшем случае в цикл попадёт именно оно, то сложность шага 4 примет вид $\mathcal{O}(e^{n \log(n)} \cdot n^4 \cdot \log^3(d) \cdot \log^3(n))$. На шаге 5 осуществляется переход к автоморфизмам поля K, сложность равна $\mathcal{O}(2^n)$ (по количеству автоморфизмов). Таким образом, общая сложность этого цикла равна $\mathcal{O}(e^{n \log(n)} \cdot n^4 \cdot \log^3(d) \cdot \log^3(n) \cdot 2^{2n})$. Циклы на шагах 7 и 8 повторяются $2^n - 1$ и 2^n раз соответственно, но осуществляемые в теле внутреннего цикла операции имеют сложность $\mathcal{O}(1)$. Таким образом, общая сложность этих циклов равна $\mathcal{O}(2^{2n})$.

Окончательно общая сложность алгоритма 2 равна

$$\mathcal{O}(e^{n \log(n)} \cdot n^4 \cdot \log^3(d) \cdot \log^3(n) \cdot 2^{2n} + 2^{2n}).$$

Упростив, получим $\mathcal{O}(e^{n \log(n)} \cdot n^4 \cdot \log^3(d) \cdot \log^3(n) \cdot 2^{2n})$. ■

3.3. П р и м е р

Проиллюстрируем вычисление элемента и идеала Штикельбергера в случае триквадратичного поля $K = \mathbb{Q}(\sqrt{-7}, \sqrt{10}, \sqrt{13})$. Имеем: $-7 \equiv 1 \bmod 4$, $10 \equiv 2 \bmod 8$, $13 \equiv 1 \bmod 4$. Выпишем все подполя исходного поля K:

$$\mathbb{Q}(\sqrt{-7}), \mathbb{Q}(\sqrt{10}), \mathbb{Q}(\sqrt{13}), \mathbb{Q}(\sqrt{-7}, \sqrt{10}), \mathbb{Q}(\sqrt{-7}, \sqrt{13}), \mathbb{Q}(\sqrt{10}, \sqrt{13}), \mathbb{Q}(\sqrt{-7}, \sqrt{10}, \sqrt{13}).$$

Вычислим элементы Штикельбергера, соответствующие каждому из подполей:

$$\begin{aligned}
\theta'_7(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_7)}\left(\mathsf{res}_{\mathbb{Q}(\zeta_7)/K\cap\mathbb{Q}(\zeta_7)}(\theta_7(-1))\right) = \\
&= id + \rho_1 + \rho_2 + \rho_3 + 2\rho_4 + 2\rho_5 + 2\rho_6 + 2\rho_7, \\
\theta'_{40}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{40})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{40})/K\cap\mathbb{Q}(\zeta_{40})}(\theta_{40}(-1))\right) = \\
&= 4id + 4\rho_1 + 4\rho_2 + 4\rho_3 + 4\rho_4 + 4\rho_5 + 4\rho_6 + 4\rho_7, \\
\theta'_{13}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{13})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{13})/K\cap\mathbb{Q}(\zeta_{13})}(\theta_{13}(-1))\right) = \\
&= 3id + 3\rho_1 + 3\rho_2 + 3\rho_3 + 3\rho_4 + 3\rho_5 + 3\rho_6 + 3\rho_7,
\end{aligned}$$

$$\begin{aligned}
\theta'_{91}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{91})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{91})/K\cap\mathbb{Q}(\zeta_{91})}(\theta_{91}(-1))\right) = \\
&= 9id + 10\rho_1 + 9\rho_2 + 10\rho_3 + 9\rho_4 + 8\rho_5 + 9\rho_6 + 8\rho_7, \\
\theta'_{280}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{280})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{280})/K\cap\mathbb{Q}(\zeta_{280})}(\theta_{280}(-1))\right) = \\
&= 11id + 11\rho_1 + 13\rho_2 + 13\rho_3 + 13\rho_4 + 13\rho_5 + 11\rho_6 + 11\rho_7, \\
\theta'_{520}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{520})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{520})/K\cap\mathbb{Q}(\zeta_{520})}(\theta_{520}(-1))\right) = \\
&= 24id + 24\rho_1 + 24\rho_2 + 24\rho_3 + 24\rho_4 + 24\rho_5 + 24\rho_6 + 24\rho_7, \\
\theta'_{3640}(-1) &= \mathsf{cor}_{K/K\cap\mathbb{Q}(\zeta_{3640})}\left(\mathsf{res}_{\mathbb{Q}(\zeta_{3640})/K\cap\mathbb{Q}(\zeta_{3640})}(\theta_{3640}(-1))\right) = \\
&= 71id + 75\rho_1 + 73\rho_2 + 69\rho_3 + 73\rho_4 + 69\rho_5 + 71\rho_6 + 75\rho_7,
\end{aligned}$$

где ρ_i — автоморфизмы поля K. Идеал Штикельбергера в соответствии с определением 6 примет следующий вид (полагаем, что столбцы матрицы соответствуют автоморфизмам G_K, а строки — образующим идеала):

$$I = \begin{pmatrix}
1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\
2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \\
4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
11 & 11 & 13 & 13 & 13 & 13 & 11 & 11 \\
13 & 13 & 11 & 11 & 11 & 11 & 13 & 13 \\
9 & 10 & 9 & 10 & 9 & 8 & 9 & 8 \\
10 & 9 & 10 & 9 & 8 & 9 & 8 & 9 \\
9 & 8 & 9 & 8 & 9 & 10 & 9 & 10 \\
8 & 9 & 8 & 9 & 10 & 9 & 10 & 9 \\
24 & 24 & 24 & 24 & 24 & 24 & 24 & 24 \\
71 & 75 & 73 & 69 & 73 & 69 & 71 & 75 \\
75 & 71 & 69 & 73 & 69 & 73 & 75 & 71 \\
73 & 69 & 71 & 75 & 71 & 75 & 73 & 69 \\
69 & 73 & 75 & 71 & 75 & 71 & 69 & 73
\end{pmatrix}.$$

4. Вычисление группы классов мультиквадратичных полей

В заключение покажем, как идеал Штикельбергера связан с числом группы классов мультиквадратичного поля. Через Cl_K будем обозначать группу классов числового поля K с кольцом целых O_K, то есть фактор-группу $\mathcal{I}/\mathcal{P}$, где $\mathcal{I}$ — мультипликативная группа дробных идеалов O_K; $\mathcal{P}$ — группа главных идеалов O_K. Группа классов Cl_K конечна, её размер называется числом классов и обозначается h_K. Вычисление Cl_K или её размера h_K — одна из фундаментальных задач алгоритмической теории чисел [17]. Из теоремы Брауера — Зигеля [18, 19] известно, что асимптотически $h_K \sim \frac{1}{2}\sqrt{|\Delta_K|}$, где Δ_K — дискриминант K.

Наиболее быстрые из известных алгоритмов вычисления Cl_K для произвольного поля K основаны на методе исчисления индексов [20, 21] и работают за время, субэкспоненциальное от Δ_K. Для некоторых полей существуют специальные алгоритмы, в частности, для мультиквадратичных предложен метод [22], значительно ускоряющий вычисление Cl_K.

Задача вычисления Cl_K и h_K более интересна в случае мнимых полей. Для действительных полей гипотеза Коэна — Ленстры [23] утверждает, что бóльшая часть действительных квадратичных полей является областью главных идеалов (то есть Cl_K тривиальна). В [24] гитопеза расширена на поля бóльших степеней. Кроме того, для

действительных полей идеал Штикельбергера тривиален [2, с. 94], поэтому сосредоточимся на мнимых полях.

Мнимое квадратичное поле. Рассмотрим $K = \mathbb{Q}(\sqrt{d})$, $d < 0$, d свободно от квадратов. Для произвольного d алгоритм Хафнера — МакКёрли [20] вычисляет Cl_K за время $e^{\mathcal{O}\left(\sqrt{\ln|d| \cdot \ln\ln|d|}\right)}$.

П. Шмид в [25, док-во теоремы 2] показал, что $h_K = 2v - \varphi(d)/2$, где v — сумма квадратичных невычетов в $(\mathbb{Z}_d)^\times$. Таким образом, зная $\theta_d(-1)$, мы знаем h_K. Для произвольного d вычисление значений u, v по формулам (3) займёт $\mathcal{O}(d)$ времени, что уступает в асимптотике алгоритму Хафнера — МакКёрли. Однако, как показано в п. 2, для простого d, такого, что $|d| \equiv 1 \bmod 4$, значения u, v, а значит, и h_K можно вычислить за время $\operatorname{poly}\log(d)$.

Совсем недавно мнимые квадратичные поля были предложены для эффективных конструкций так называемых *проверяемых функций задержки* (verifiable delay functions) [5, 6], которые, например, активно использует блокчейн Chia[3]. Для обеспечения должного уровня безопасности при генерации большого d алгоритм должен проверять, выполняется ли $|d| \equiv 1 \bmod 4$. Для $|d| \equiv 3 \bmod 4$ остаётся открытым вопрос, для каких значений d алгоритм Хафнера — МакКёрли работает быстрее на практике, чем непосредственное вычисление u, v по формулам (3). Отметим, что вычисления сумм в (3) тривиально распараллеливаются, что даёт значительное практическое преимущество этому наивному методу относительно алгоритма Хафнера — МакКёрли.

Мнимые биквадратичное и мультиквадартичные поля. Положим теперь $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{d_2})$, $d_i < 0$, d_i свободны от квадратов и взаимно просты. Поле K содержит три квадратичных подполя: два мнимых $k_1 = \mathbb{Q}(\sqrt{d_1})$, $k_2 = \mathbb{Q}(\sqrt{d_2})$ и одно действительное $k_3 = \mathbb{Q}(\sqrt{d_1 d_2})$. Т. Кубота в [26, Satz 5] доказал, что $h_K = \dfrac{q(K/\mathbb{Q})}{2} h_{k_1} h_{k_2} h_{k_3}$, где $q(K/\mathbb{Q}) = [U_K : U_{k_1} U_{k_2} U_{k_3}]$ — индекс единиц поля K (то есть степень расширения U_K — группы единиц поля K — над группой $U_{k_1} U_{k_2} U_{k_3}$, порождённой образующими групп единиц подполей k_i). Этот индекс для мультиквадратичных полей можно получить за время $\operatorname{poly}\log(\Delta_K)$ с помощью алгоритма Бауха и др. [27]. Исходя из рассуждений выше, можно посчитать число классов мнимых подполей h_{k_1}, h_{k_2}, получив тем самым h_K с точностью до множителя h_{k_3}. Вычислению h_{k_3} — числа группы классов действительных квадратичных полей — посвящены работы [28, 29]. В продолжение эвристики Коэна — Ленстры в [30] показано, что h_{k_3} с большой вероятностью не делится на малые простые, а в [31, 32] приведены случаи, когда $h_{k_3} \in \{1, 2\}$. Отметим, что идеал Штикельбергера для действительных квадратичных полей (см. случай 3 в п. 2) не даёт информации о числе группы классов.

Формула Куботы обобщается до мультиквадратичного случая. Из [33, 34] известно, что для $K = \mathbb{Q}(\sqrt{d_1}, \ldots \sqrt{d_n})$ справедлива формула $h_K = \dfrac{1}{2^\nu} q(K/\mathbb{Q}) \prod\limits_i h(k_i)$, где $\nu = (n-1)(2^{n-2} - 1) + 2^{n-1} - 1$, а произведение по k_i пробегает все 2^{n-1} квадратичных подполей K. В работе [35] классифицированы всевозможные значения $q(K/\mathbb{Q})$. Таким образом, при вычислении h_K для мультиквадратичного поля ключевое значение играет алгоритм, вычисляющий группу классов квадратичного поля, а эта задача алгоритмически эквивалентна вычислению идеала Штикельбергера для мнимых полей. Классификация мультиквадратичных полей с числом классов 1 представлена в [36].

[3] `https://www.chia.net/`

Открытые вопросы. Было бы интересно расширить полученные результаты в следующих (прямо противоположных друг другу) направлениях: 1) ускорение алгоритма вычисления идеала Штикельбергера; 2) приложение мультиквадратичных полей к конструкциям функций с задержкой с эффективной верификацией как полей, число группы классов которых трудно вычислить на практике.

ЛИТЕРАТУРА

1. *Stickelberger L.* Über eine Verallgemeinerung der Kreistheilung // Math. Ann. 1890. V. 37. No. 3. P. 312–367.
2. *Washington L. C.* Introduction to Cyclotomic Fields. Springer, 1997.
3. *Denomme R.* A History of Stickelberger's Theorem. `https://core.ac.uk/download/pdf/159568254.pdf` — The Ohio State University, 2009.
4. *Cramer R., Ducas L., and Wesolowski B.* Short Stickelberger class relations and application to ideal-SVP // EUROCRYPT 2017. LNCS. 2017. V. 10210. P. 324–348.
5. *Wesolowski B.* Efficient verifiable delay functions // EUROCRYPT 2019. LNCS. 2019. V. 11478. P. 379–407.
6. *Pietrzak K.* Simple verifiable delay functions // Innovations in Theoretical Computer Science Conference, ITCS, 2019. P. 1–60.
7. *Pedrouzo-Ulloa A., Troncoso-Pastoriza J. R., Gama N., et al.* Revisiting Multivariate Ring Learning with Errors and its Applications on Lattice-based Cryptography. IACR Cryptol. ePrint Arch. 2019/1109.
8. *Kučera R.* On the Stickelberger ideal and circular units of a compositum of quadratic fields // J. Number Theory. 1996. V. 56. No. 1. P. 139–166.
9. *Олефиренко Д. О., Киршанова Е. А., Малыгина Е. С., Новоселов С. А.* Алгоритм вычисления элемента Штикельбергера для мнимых мультиквадратичных полей // Прикладная дискретная математика. Приложение. 2020. № 13. С. 12–17.
10. *Schmal B.* Diskriminanten, $\mathbb{Z}$-anzheitsbasen und relative Ganzheitsbasen bei multiquadratischen Zahlkörpern // Archiv der Mathematik. 1989. V. 52. No. 3. P. 245–257.
11. *Sinnott W.* On the Stickelberger ideal and the circular units of an Abelian field // Inventiones Mathematicae. 1980. V. 62. P. 181–234.
12. *Berndt B. C., Evans R. J., and Williams K. S.* Gauss and Jacobi sums. N.Y.: Wiley, 1998.
13. *Lang S.* Cyclotomic Fields I and II. N.Y.: Springer, 1990.
14. *Milne J.* Class field theory (v4.03). 2020. `www.jmilne.org/math/`
15. *Weintraub S.* Galois Theory. Second Ed. Springer, 2009.
16. *Aebi C. and Cairns G.* Sums of quadratic residues and nonresidues. arXiv:1512.00896. 2015.
17. *Cohen H.* A Course in Computational Algebraic Number Theory. Springer Verlag, 1995.
18. *Brauer R.* On the zeta-function of algebraic number fields // Amer. J. Math. 1947. V. 69. No. 2. P. 243–250.
19. *Siegel C. L.* Über die Classenzahl quadratischer Zahlkörper // Acta Arithmetica. 1935. V. 1. No. 1. P. 83–86.
20. *Hafner J. L. and McCurley K. S.* A rigorous subexponential algorithm for computation of class groups // J. Amer. Math. Soc. 1989. V. 2. No 4. P. 837–850.
21. *Buchmann J.* A subexponential algorithm for the determination of class groups and regulators of algebraic number fields // Séminaire de Théorie des Nombres (Paris 1988/1989), Progr. Math. No. 91. Birkhäuser, Boston, 1990. P. 27–41.
22. *Biasse J.-F. and Van Vredendaal C.* Fast multiquadratic S-unit computation and application to the calculation of class groups // Proc. ANTS XIII. 2019. P. 103–118.

23. *Cohen H. and Lenstra H. W.* Heuristics on class groups of number fields // Number Theory Noordwijkerhout. Lecture Notes in Math. 1983. V. 1068. P. 33–62.
24. *Cohen H. and Martinet J.* Class groups of number fields: numerical heuristics // Math. Comp. 1987. V. 48. No. 177. P. 123–137.
25. *Schmid P.* The Stickelberger element of an imaginary quadratic field // Acta Arithmetica. 1999. V. 91. No. 2. P. 165–169.
26. *Kubota T.* Über den bizyklischen biquadratischen Zahlkörper // Nagoya Math. J. 1956. No. 10. P. 65–85.
27. *Bauch J., Bernstein D. J., de Valence H., et al.* Short generators without quantum computers: The case of multiquadratics // EUROCRYPT 2017. LNCS. 2017. V. 10210. P. 27–59.
28. *Bhand A. and Ram Murty M.* Class numbers of quadratic fields // Hardy-Ramanujan J. 2019. V. 42. P. 1–9.
29. *Sato H.* On class number formula for the real quadratic fields // Proc. Japan Acad. Ser. A. Math. Sci. 2004. V. 80. No. 7. P. 129–130.
30. *Ono K.* Indivisibility of class numbers of real quadratic fields // Compositio Mathematica. 1999. V. 119. P. 1–11.
31. *Mollin R. and Williams H.* On a determination of real quadratic fields of class number one and related continues fraction period length less than 25 // Proc. Japan Acad. Ser. A. Math. Sci. 1991. V. 67. P. 20–25.
32. *Mollin R. and Williams H.* On real quadratic fields of class number two // Mathemat. Comput. 1992. V. 59. No. 200. P. 625–632.
33. *Kuroda H.* Über die Klassenzahlen algebraischer Zahlkörper // Nagoya Math. J. 1950. V. 1. P. 1–10.
34. *Herglotz G.* Über einen Dirichletschen Satz // Mathematische Zeitschrift. 1922. V. 12. No. 1. P. 255–261.
35. *Benjamin E., Lemmermeyer F., and Snyder C.*. On the unit group of some multiquadratic number fields // Pacific J. Math. 2007. V. 230. P. 27–40.
36. *Feaver A.* Imaginary multiquadratic fields of class number 1 // J. Number Theory. 2017. V. 174. P. 93–117.

REFERENCES

1. *Stickelberger L.* Über eine Verallgemeinerung der Kreistheilung. Math. Ann., 1890, vol. 37, no. 3, pp. 312–367.
2. *Washington L. C.* Introduction to Cyclotomic Fields. Springer, 1997.
3. *Denomme R.* A History of Stickelberger's Theorem. `https://core.ac.uk/download/pdf/159568254.pdf` — The Ohio State University, 2009.
4. *Cramer R., Ducas L., and Wesolowski B.* Short Stickelberger class relations and application to ideal-SVP. EUROCRYPT 2017, LNCS, 2017, vol. 10210, pp. 324–348.
5. *Wesolowski B.* Efficient verifiable delay functions. EUROCRYPT 2019, LNCS, vol. 11478, pp. 379–407.
6. *Pietrzak K.* Simple verifiable delay functions. Innovations in Theoretical Computer Science Conference, ITCS, 2019, pp. 1–60.
7. *Pedrouzo-Ulloa A., Troncoso-Pastoriza J. R., Gama N., et al.* Revisiting Multivariate Ring Learning with Errors and its Applications on Lattice-based Cryptography. IACR Cryptol. ePrint Arch. 2019/1109.
8. *Kučera R.* On the Stickelberger ideal and circular units of a compositum of quadratic fields. J. Number Theory, 1996, vol. 56, no. 1, pp. 139–166.

9. *Olefirenko D. O., Kirshanova E. A., Malygina E. S., and Novoselov S. A.* Algoritm vychisleniya elementa Shtikel'bergera dlya mnimykh mul'tikvadratichnykh poley [An algorithm for computing the Stickelberger elements for imaginary multiquadratic fields]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2020, no. 13, pp. 12–17. (in Russian)
10. *Schmal B.* Diskriminanten, $\mathbb{Z}$-anzheitsbasen und relative Ganzheitsbasen bei multiquadratischen Zahlkörpern. Archiv der Mathematik, 1989, vol. 52, no. 3, pp. 245–257.
11. *Sinnott W.* On the Stickelberger ideal and the circular units of an Abelian field. Inventiones Mathematicae, 1980, vol. 62, pp. 181–234.
12. *Berndt B. C., Evans R. J., and Williams K. S.* Gauss and Jacobi sums. N.Y., Wiley, 1998.
13. *Lang S.* Cyclotomic Fields I and II. N.Y., Springer, 1990.
14. *Milne J.* Class field theory (v4.03). 2020. `www.jmilne.org/math/`
15. *Weintraub S.* Galois Theory. Second Edition. Springer, 2009.
16. *Aebi C. and Cairns G.* Sums of quadratic residues and nonresidues. arXiv:1512.00896. 2015.
17. *Cohen H.* A Course in Computational Algebraic Number Theory. Springer Verlag, 1995.
18. *Brauer R.* On the zeta-function of algebraic number fields. Amer. J. Math., 1947, vol. 69, no. 2, pp. 243–250.
19. *Siegel C. L.* Über die Classenzahl quadratischer Zahlkörper. Acta Arithmetica, 1935, vol. 1, no. 1, pp. 83–86.
20. *Hafner J. L. and McCurley K. S.* A rigorous subexponential algorithm for computation of class groups. J. Amer. Math. Soc., 1989, vol. 2, no 4, pp. 837–850.
21. *Buchmann J.* A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. Séminaire de Théorie des Nombres (Paris 1988/1989), Progr. Math., no. 91, Birkhäuser, Boston, 1990, pp. 27–41.
22. *Biasse J.-F. and Van Vredendaal C.* Fast multiquadratic S-unit computation and application to the calculation of class groups. Proc. ANTS XIII, 2019, pp. 103–118.
23. *Cohen H. and Lenstra H. W.* Heuristics on class groups of number fields. Number Theory Noordwijkerhout. Lecture Notes in Math., 1983, vol. 1068, pp. 33–62.
24. *Cohen H. and Martinet J.* Class groups of number fields: numerical heuristics. Math. Comp., 1987, vol. 48, no. 177, pp. 123–137.
25. *Schmid P.* The Stickelberger element of an imaginary quadratic field. Acta Arithmetica, 1999, vol. 91, no. 2, pp. 165–169.
26. *Kubota T.* Über den bizyklischen biquadratischen Zahlkörper. Nagoya Math. J., 1956, no. 10, pp. 65–85.
27. *Bauch J., Bernstein D. J., de Valence H., et al.* Short generators without quantum computers: The case of multiquadratics. EUROCRYPT 2017, LNCS, 2017, vol. 10210, pp. 27–59.
28. *Bhand A. and Ram Murty M.* Class numbers of quadratic fields. Hardy-Ramanujan J., 2019, vol. 42, pp. 1–9.
29. *Sato H.* On class number formula for the real quadratic fields. Proc. Japan Acad. Ser. A. Math. Sci., 2004, vol. 80, no. 7, pp. 129–130.
30. *Ono K.* Indivisibility of class numbers of real quadratic fields. Compositio Mathematica, 1999, vol. 119, pp. 1–11.
31. *Mollin R. and Williams H.* On a determination of real quadratic fields of class number one and related continues fraction period length less than 25. Proc. Japan Acad. Ser. A. Math. Sci., 1991, vol. 67, pp. 20–25.
32. *Mollin R. and Williams H.* On real quadratic fields of class number two. Mathemat. Comput., 1992, vol. 59, no. 200, pp. 625–632.

33. *Kuroda H.* Über die Klassenzahlen algebraischer Zahlkörper. Nagoya Math. J., 1950, vol. 1, pp. 1–10.
34. *Herglotz G.* Über einen Dirichletschen Satz. Mathematische Zeitschrift, 1922, vol. 12, no. 1, pp. 255–261.
35. *Benjamin E., Lemmermeyer F., and Snyder C.* On the unit group of some multiquadratic number fields. Pacific J. Math., 2007, vol. 230, pp. 27–40.
36. *Feaver A.* Imaginary multiquadratic fields of class number 1. J Number Theory, 2017, vol. 174, pp. 93–117.

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 003.26 + 004.056

ОСНОВНЫЕ ЭТАПЫ РАЗВИТИЯ КРИПТОГРАФИЧЕСКИХ ПРОТОКОЛОВ SSL/TLS И IPsec

И. В. Мартыненков

Астраханский государственный технический университет, г. Астрахань, Россия

Рассматриваются основные этапы развития криптографических протоколов от SSL 2.0 (Secure Socket Layer) до TLS 1.3 (Transport Layer Security), обеспечивающих защиту данных транспортного уровня модели OSI. Приводится краткое описание модификации протокола RuTLS, построенного на базе TLS 1.3, и их основные отличия. Развитие IPsec, предоставляющего криптографическую защиту коммуникаций на сетевом уровне модели OSI, рассмотрено на примерах развития трёх наиболее часто применяемых протоколов, на основе которых он строится. В их число входят IKE (Internet Key Exchange), AH (Authentication Header), ESP (Encapsulation Security Payload).

Ключевые слова: *криптографические протоколы, SSL, TLS, IPsec.*

DOI 10.17223/20710410/51/2

THE MAIN STAGES OF DEVELOPMENT OF THE CRYPTOGRAPHIC PROTOCOLS SSL/TLS AND IPsec

I. V. Martynenkov

Astrakhan State Technical University, Astrakhan, Russia

E-mail: mivpost@yandex.ru

The paper discusses the main stages of development of cryptographic protocols from SSL 2.0 (Secure Socket Layer) to TLS 1.3 (Transport Layer Security), which ensure the protection of transport layer data in the OSI model. A brief description of the modification of the RuTLS protocol based on TLS 1.3 and their main differences is given. The development of IPsec, which provides cryptographic protection of communications at the network level of the OSI model, is considered using examples of the development of the three most commonly used protocols. These include IKE (Internet Key Exchange), AH (Authentication Header), and ESP (Encapsulation Security Payload). For the SSL/TLS and IPsec specifications, the basic handshake protocols and the main stages of their development are considered. The described handshakes include primary cryptographic information exchange cycles in the form of identifiers of interaction participants, one-time numbers, lists of supported cryptographic combinations. Authentication of participants based on certificates, shared symmetric keys, data exchange for establishing a shared Diffie — Hellman secret, development of key material for secret keys of communication sessions, message authentication, and other cryptographic parameters are presented. For different versions of SSL/TLS and IPsec,

the logical structures of application data cryptographic protection functions are described.

Keywords: *cryptographic protocols, SSL, TLS, IPsec.*

Введение

Целью работы является описание основных этапов развития криптографических протоколов защиты транспортного и сетевого уровней модели OSI, представленных семейством спецификаций SSL/TLS, а также IPsec. Рассматриваются разные этапы взаимодействия участников информационного обмена, необходимые для установления безопасного канала связи. К ним относятся, например, обмен первичной криптографической информацией в виде идентификаторов участников взаимодействия, одноразовых номеров, списков поддерживаемых комбинаций криптографических алгоритмов; аутентификация участников на основе сертификатов, общих симметричных ключей; обмен данными для установления разделяемого секрета Диффи — Хеллмана; выработка ключевого материала для секретных ключей сеансов связи, аутентификации сообщений и другие криптографические параметры. Дополнительно приводятся основные меры повышения безопасности протокола RuTLS, разработанного на основе TLS 1.3.

В связи с непрерывным анализом и исследованиями криптографических протоколов в рамках предотвращения возникновения выявленных угроз безопасности с течением времени способы взаимодействия участников информационного обмена развивались и претерпевали изменения. Развитие криптографических протоколов в направлении увеличения безопасности защищённых соединений рассмотрено согласно хронологическому порядку выпуска соответствующих спецификаций от SSL 2.0 до TLS 1.3, а также спецификаций, составляющих основу функционала протокола IPsec, а именно IKE, AH/ESP. При этом приводятся только наиболее важные изменения относительно предыдущих релизов криптографических протоколов.

1. Развитие криптографических протоколов семейства SSL/TLS

1.1. Протокол SSL 2.0

Протокол SSL 2.0 [1] разработан компанией «Netscape» и в 1995 г. опубликован в качестве исторической справки как первый и небезопасный [2] представитель семейства одноимённых протоколов.

Рукопожатия SSL 2.0 включают три типовых сценария: установка нового безопасного соединения, возобновление соединения и аутентификация клиента C (Client). Сообщения квитирования установки нового соединения состоят из шести циклов. Клиент отправляет случайное число r_c (Random, размер r от 16 до 32 байт), генерируемое для каждого соединения в рамках аутентификации и борьбы с атаками воспроизведения (Replay Attack), а также список криптонаборов клиента cs_c (Cipher Suites). Сервер S (Server) отвечает случайным числом идентификатора соединения cid_s (Connection Identifier, размер cid от 16 до 32 байт), генерируемым для целей, аналогичных r_c, а также сертификатом crt_s (Certificate) открытого ключа сервера (e_s) типа X.509 с подписью согласно PKCS#1 и выбранным криптонабором cs_s.

Клиент генерирует главный секрет mk (Master Key) и передаёт его в зашифрованном виде с использованием открытого ключа сервера e_s. Клиент и сервер применяют согласованный mk для генерации k_1, k_2 и iv (сеансовые ключи записи клиента — чтения сервера, чтения клиента — записи сервера, которые совпадают с ключами аутен-

тификации сообщений, и вектор инициализации для блочных шифров) за счёт представленного ниже преобразования, константа зависит от согласованного криптонабора (символ «$\|$» означает конкатенацию данных):

$$(k_1 \parallel k_2 \parallel iv) = \mathrm{MD5}(mk \parallel \mathrm{const} \parallel r \parallel cid).$$

Затем на согласованных криптонаборах и (выработанных) ключах происходит последовательный обмен идентификаторами cid_s, r_c и выработанным сервером новым идентификатором сеанса sid_s (Session Identifier, 16 байт), который содержит копию mk. Полученный sid_s может храниться в кэш-памяти сервера и клиента для целей будущего ускоренного восстановления сеанса:

$$\begin{aligned}
&1) \quad C \to S: \quad r_c, cs_c;\\
&2) \quad C \leftarrow S: \quad cid_s, crt_s, cs_s;\\
&3) \quad C \to S: \quad E_{e_s}(mk);\\
&4) \quad C \to S: \quad E_{k_1}(cid_s);\\
&5) \quad C \leftarrow S: \quad E_{k_2}(r_c);\\
&6) \quad C \leftarrow S: \quad E_{k_1}(sid_s).
\end{aligned}$$

Для попытки возобновления сессии циклы 1, 2 и 3 заменяются двумя другими циклами, в которых флаг sid_{hc} указывает на возможность возобновления сессии при наличии $sid_s = sid_c$ в кэш-памяти сервера, сообщения передаются в открытом виде. Применение cid и sid делают сеансовые ключи зависимыми от восстанавливаемого и текущего сеанса:

$$\begin{aligned}
&1') \quad C \to S: \quad r_c, sid_c, cs_c;\\
&2') \quad C \leftarrow S: \quad cid_s, sid_{hc}.
\end{aligned}$$

Аутентификация клиента происходит за счёт дополнительного выбора сервером типа аутентификации aut_s (Authentication), типа сертификата клиента crt_t (Certificate Type) и данных rd_c (Response Data) как функции от aut_s между циклами 5 и 6, сообщения зашифрованы с использованием согласованных криптонаборов и ключей:

$$\begin{aligned}
&5'') \quad C \leftarrow S: \quad E_{k_2}(aut_s, r'_s);\\
&6'') \quad C \to S: \quad E_{k_1}(crt_t, crt_c, rd_c).
\end{aligned}$$

После выработки общих параметров безопасности происходит обмен данными приложения $data$, которые защищены в порядке преобразования MAC-and-Encrypt (M&E). Сообщения строятся с использованием симметричных ключей шифрования k, ключей аутентификации сообщений k, дополнения pad до кратности размера блока блочного шифра, а также 32-разрядного порядкового номера SN (Sequence Number) сообщения (инкрементный счетчик с зацикливанием):

$$data_{\mathrm{app}} = E_k(data \parallel pad) \parallel \mathrm{MD5}(k \parallel data \parallel pad \parallel SN).$$

Для SSL 2.0 определены криптонаборы из комбинаций RC2, IDEA, DES, режима CBC, MD5, RSA и сертификатов X.509. Экспортные реализации передают часть ключа в открытом виде. Например, для экспортного RC4 88 бит ключа передаются открыто и только 40 бит в зашифрованном виде.

SSL 2.0 поддерживает оповещения о следующих ошибках: отсутствие выбранного шифронабора/ключа, отсутствие сертификата клиента при его аутентификации, некорректный сертификат клиента с ошибочной подписью или идентификаторами, неподдерживаемый тип сертификата.

1.2. Протокол SSL 3.0

Протокол SSL 3.0 [3] также неустойчив ко многим атакам, однако по сравнению с SSL 2.0 претерпел значительные изменения и стал основоположником будущих версий TLS. Версия SSL 3.0 устанавливается в значение 0x0300. Протокол квитирования включает пять основных циклов. Клиент отправляет открытый запрос на сервер с самой новой версией поддерживаемого протокола v_c, случайным числом r_c (32 байта: 4 байта время/дата в UNIX-формате и 28 байт генератора случайных чисел, $r_c \neq r_s$), $sid_c = 0$ для новой сессии, список поддерживаемых криптонаборов cs_c и методов сжатия $comp_c$ (Compression). Сервер отвечает своими параметрами и избранными cs_s и $comp_s$ из списков клиента.

При установлении соединения могут использоваться опциональные сообщения (отмечены индексом «*»). К ним относятся сертификат сервера crt_s, типы сертификатов и информация о центрах сертификации (crt_{req}) неанонимного сервера, сообщения ke_s (Key Exchange), содержащие параметры prm_s (Parameters) для обмена ключевой информацией: модуль ($\bmod p$), открытая фиксированная или временная экспонента (e_s) RSA; модуль ($\bmod p$), генератор (g) и открытое значение (g^x) Диффи — Хеллмана (DH); случайное число (r_f) Fortezza.

Открытые параметры подписываются RSA или DSS. Подписываемое RSA сообщение кодируется в формате блока типа 1 PKCS#1 [5], шифрование RSA выполняется над сообщением, отформатированным в виде блока типа 2 PKCS#1:

$$sig_{\mathrm{RSA}} = E_{\mathrm{RSA}}(\mathrm{MD5}(r_c \parallel r_s \parallel prm_s \parallel \mathrm{SHA1}(r_c \parallel r_s \parallel prm_s))),$$
$$sig_{\mathrm{DSS}} = E_{\mathrm{DSS}}(\mathrm{SHA1}(r_c \parallel r_s \parallel prm_s)).$$

Клиент отвечает зашифрованным предварительным главным секретом $prems$ (Pre-Master Secret). Для DH $prems$ является согласованным ключом DH. При аутентификации и согласовании ключей на основе RSA 48 байт $prems = (v_c \parallel rnd_c)$, где v_c (Version) — 2 байта, rnd_c (Random Data) — 46 случайных байт. Сообщение зашифровывается на открытом ключе сервера e_s из сертификата сервера crt_s или временном ключе e_t из обмена сервера ke_s. Для защиты $prems$ в случае алгоритма Fortezza также передаётся открытый ключ алгоритма обмена ключами (KEA) и его подпись на секретном ключе DSS клиента, 128 случайных байт для расчёта KEA, симметричные ключи записи сервера и клиента, зашифрованные с использованием токена ключа шифрования (TEK), векторы инициализации для шифрования сервера, клиента и TEK:

$$ke_c = E_{\mathrm{RSA}}(prems).$$

Далее формируется 48 байт главного секрета ms (Master Secret) и блок ключевого материала $keyb$ (Key Block), который последовательно нарезается на ключи необходимого размера в следующем порядке: ключи записи MAC-кодов клиента/сервера, ключи записи симметричного шифрования клиента/сервера, векторы инициализации клиента/сервера. Ключи экспортных реализаций строятся хешированием ключей и случайных значений из этапа генерации $keyb$. Для Fortezza главный секрет ms используется только для вычисления MAC-кодов:

$$ms = \mathrm{MD5}(prems \parallel \mathrm{SHA}(\text{«}A\text{»} \parallel prems \parallel r_c \parallel r_s)) \parallel \mathrm{MD5}(prems \parallel$$
$$\parallel \mathrm{SHA}(\text{«}BB\text{»} \parallel prems \parallel r_c \parallel r_s)) \parallel \mathrm{MD5}(prems \parallel \mathrm{SHA}(\text{«}CCC\text{»} \parallel prems \parallel r_c \parallel r_s));$$

$$keyb = (k_{c_w}^{\mathrm{MAC}} \parallel k_{s_w}^{\mathrm{MAC}} \parallel k_{c_w} \parallel k_{s_w} \parallel iv_c \parallel iv_s) =$$
$$= \mathrm{MD5}(ms \parallel \mathrm{SHA}(\text{«}A\text{»} \parallel ms \parallel r_s \parallel r_c)) \parallel \mathrm{MD5}(ms \parallel \mathrm{SHA}(\text{«}BB\text{»} \parallel ms \parallel r_s \parallel r_c)) \parallel$$
$$\parallel \mathrm{MD5}(ms \parallel \mathrm{SHA}(\text{«}CCC\text{»} \parallel ms \parallel r_s \parallel r_c)) \parallel \ldots;$$

$$k^e_{c_w} = \mathrm{MD5}(k_{c_w} \parallel r_c \parallel r_s), \quad k^e_{s_w} = \mathrm{MD5}(k_{s_w} \parallel r_s \parallel r_c),$$
$$iv^e_c = \mathrm{MD5}(r_c \parallel r_s), \quad iv^e_s = \mathrm{MD5}(r_s \parallel r_c).$$

Отправляется также финальное сообщение, которое является первым защищённым сообщением на согласованных секретах и криптонаборах cs. Используется согласованный хеш H (Hash, MD5 или SHA), const = 0x434C4E54/0x53525652 (клиент/сервер), $hsmsg$ (Handshake Message) — все сообщения квитирования, исключая текущие fin (Finished) и cs, pad_1 (Padding, 48 значений 0x36 для MD5 или 40 значений для SHA) и pad_2 (48 значений 0x5C для MD5 или 40 значений для SHA):

$$fin_c = H(ms \parallel pad_2 \parallel H(hsmsg \parallel \mathrm{const} \parallel ms \parallel pad_1)).$$

Верификация сертификата (crt_{verif}) содержит главный секрет ms и все сообщения квитирования, исключая crt_{verif} и cs. Данные подписываются алгоритмом RSA или DSS на секретном ключе сертификата клиента, что необходимо для явной проверки сертификата клиента с возможностью подписи:

$$sig_{\mathrm{RSA/DSS}} = E_{\mathrm{RSA\ /\ DSS}}(H(ms \parallel pad_2 \parallel H(hsmsg \parallel ms \parallel pad_1))).$$

В заключительном цикле сервер формирует fin_s, которое включает все предыдущие сообщения квитирования, причём $fin_c \neq fin_s$, так как, как минимум, fin_c не содержит fin_s:

$$\begin{aligned}
&1) \quad C \to S: \quad v_c, r_c, sid_c, cs_c, comp_c;\\
&2) \quad C \leftarrow S: \quad v_s, r_s, sid_s, cs_s, comp_s;\\
&3) \quad C \leftarrow S: \quad crt^*_s, ke^*_s, crt^*_{req};\\
&4) \quad C \to S: \quad ke_c, cs_c, fin_c, crt^*_c, crt^*_{\mathrm{verif}};\\
&5) \quad C \leftarrow S: \quad cs_s, fin_s.
\end{aligned}$$

Для восстановления сессии циклы 1–5 заменяются тремя обменами, причем сервер S ищет запрашиваемый sid_c (32 байта) в кэш-памяти. Это позволяет создавать несколько независимых соединений без полного квитирования на уже согласованных криптографических параметрах, экономя дорогостоящие операции асимметричной криптографии. Аутентификация sid_c выполняется в финальных сообщениях fin:

$$\begin{aligned}
&1') \quad C \to S: \quad sid_c;\\
&2') \quad C \leftarrow S: \quad sid_s, E_{k_{s_w}}(cs_c), fin_s;\\
&3') \quad C \to S: \quad E_{k_{c_w}}(cs_c), fin_c.
\end{aligned}$$

Данные приложения защищены в порядке преобразования MAC-then-Encrypt (MtE). Используется функция сжатия $comp$ и её идентификатор ($comp_t$), 64-разрядный порядковый номер сообщения SN (по два счётчика на приём/передачу для S и C, обнуляемых при изменении шифронаборов), а также размер сжатых данных $\mathrm{comp}(data)_{\mathrm{len}}$. Для поточных шифров используется структура функции, приведенная ниже, в которой ключевая гамма распространяется на следующие блоки данных непрерывно. Для блочных шифров вводится дополнение (pad) и его длина (pad_{len}) в конец блока открытого текста. В режиме CBC вектор инициализации первого блока вырабатывается протоколом квитирования, для последующих блоков iv равен последнему блоку предыдущего шифртекста:

$$\begin{aligned}
data_{\mathrm{app}} = E_{k_{c/s_w}}(comp(data) \parallel H(k^{\mathrm{MAC}}_{c/s_w} \parallel pad_2 \parallel H(k^{\mathrm{MAC}}_{c/s_w} \parallel pad_1 \parallel SN \parallel\\
\parallel comp_t \parallel comp(data)_{\mathrm{len}} \parallel comp(data)) \parallel (pad \parallel pad_{\mathrm{len}})^*)).
\end{aligned} \tag{1}$$

Криптонаборы SSL 3.0 содержат алгоритмы шифрования RC2-40, IDEA, DES-40, DES, 3DES-EDE и Fortezza в режиме CBC, RC4-40/128. Обмен ключами и аутентификация реализуются на основе алгоритма RSA, эфемерного DHE, анонимного DH и статического DH с подписями DSS и RSA и их экспортными реализациями, а также алгоритмом Fortezza-KEA. Целостность обеспечивается функциями MD5 и SHA.

Для SSL 3.0 определено 12 типов оповещений об ошибках с двумя уровнями опасности: предупреждение и фатальная ошибка. Проверки затрагивают корректность MAC-кодов, сообщений рукопожатия, сертификатов и других криптографических параметров сервера и клиента.

1.3. Протокол TLS 1.0

По сравнению с SSL 3.0 протокол TLS 1.0 [4] добавляет опциональные данные расширений ext (Extensions) в первом цикле, в том числе для совместимости с будущими версиями TLS. Они включаются в дайджесты рукопожатия. Расширения состоят из типа расширения и данных расширения ($ext = ext_t \parallel ext_{data}$) и могут включать имя сервера, длину максимального фрагмента, сетевой адрес сертификата клиента, доверенные ключи корневого центра сертификации, усечённый HMAC или статус запроса. Остальные циклы полного квитирования и восстановления сессий архитектурно изменений не претерпели и внешне соответствуют SSL 3.0, однако изменены способы вычисления их сообщений. Версия TLS 1.0 устанавливается в значение 0x0301:

$$1)\quad C \to S:\ \ v_c, r_c, sid_c, cs_c, comp_c, ext.$$

Взамен обычного хеширования TLS 1.0 определяет новую функцию вычисления случайных данных PRF (Pseudo-Random Function), которая применяет HMAC на основе MD5 и SHA1. На вход PRF поступает секрет (Secret) $s = (s_1 \parallel s_2)$, состоящий из частей одинаковой длины. Для нечётных длин s последний байт s_1 совпадает с первым байтом s_2. Используются также начальное заполнение $seed$ и метка-идентификатор L (Label), дополнительно вычисляются параметры вида $a_0 = seed$, $a_i = \text{HMAC-H}(s_j, a_{i-1})$, $i = 1, 2, \ldots$, причём для HMAC-MD5 и HMAC-SHA1 количество итераций независимо и может отличаться, части суммируются по модулю 2, а избыточные данные отбрасываются:

$$\begin{gathered}\text{PRF}(s, L, seed) = \\ = \text{HMAC-MD5}(s_1, a_1 \parallel L \parallel seed) \parallel \ldots \parallel \text{HMAC-MD5}(s_1, a_i \parallel L \parallel seed) \oplus \\ \oplus\ \text{HMAC-SHA1}(s_2, a_1 \parallel L \parallel seed) \parallel \ldots \parallel \text{HMAC-SHA1}(s_2, a_i \parallel L \parallel seed).\end{gathered} \tag{2}$$

Протокол TLS 1.0 поддерживает сертификаты X.509 [5] и исключает алгоритм Fortezza в сообщениях обмена ключевой информацией, в остальном ke_c и ke_s соответствуют SSL 3.0. Криптографические преобразования для получения 48 байт главного секрета ms, блока ключевого материала сеансовых ключей $keyb$ и векторов инициализации iv, включая экспортируемые данные, в TLS 1.0 применяют PRF (2) с константными строками на входе. Однако для вычисления $prems$ явно задаётся использование версии протокола клиента v_c из первого цикла:

$$\begin{gathered}ms = \text{PRF}(prems, \text{«master secret»}, r_c \parallel r_s), \\ keyb = (k^{\text{MAC}}_{c_w} \parallel k^{\text{MAC}}_{s_w} \parallel k_{c_w} \parallel k_{s_w} \parallel iv_c \parallel iv_s) = \text{PRF}(ms, \text{«key expansion»}, r_s \parallel r_c), \\ k^e_{c_w} = \text{PRF}(k_{c_w}, \text{«client write key»}, r_c \parallel r_s), \\ k^e_{s_w} = \text{PRF}(k_{s_w}, \text{«server write key»}, r_c \parallel r_s), \\ ivblock = (iv^e_c \parallel iv^e_s) = \text{PRF}(\text{«»}, \text{«IV block»}, r_c \parallel r_s).\end{gathered} \tag{3}$$

Финальные сообщения в TLS 1.0 теперь применяют PRF (2) с константной строкой, идентифицирующей отправителя. Исключая текущее, они включают все сообщения протокола квитирования и составляют 12 байт:

$$fin_{c/s} = \mathrm{PRF}(ms, \text{«client/server finished»}, \mathrm{MD5}(hsmsg) \parallel \mathrm{SHA1}(hsmsg)). \qquad (4)$$

Для явной проверки сертификата клиента SSL 3.0 применял двойное хеширование. TLS 1.0 упрощает вычисления crt_{verif}, в которых RSA или DSS подписывают данные вида $H(hsmsg)$, $hsmsg$ — также все сообщения квитирования, исключая crt_{verif} и cs.

Протокол TLS 1.0 исключает поддержку Fortezza для согласования общего секрета и симметричного шифрования, однако в остальном криптонаборы соответствуют SSL 3.0. Порядок криптографических преобразований данных приложения в TLS 1.0 соответствует (1) протокола SSL 3.0, однако в них функции хешировании MD5 и SHA1 заменены на HMAC-MD5 и HMAC-SHA1, а также исключены дополнения. TLS 1.0 предписывает поддержку обязательного криптографического набора в виде TLS-DHE-DSS-WITH-3DES-EDE-CBC-SHA.

Оповещения об ошибках TLS 1.0 расширены включением проверок корректности расшифровки, достоверности подписей, финальных сообщений, сообщений обмена ключами. Добавлен контроль переполнения длин сообщений, наличия доверенного корневого центра сертификации, контроль допустимых значений полей сообщений и другие. Всего 23 типа сообщений об ошибках.

1.4. Протокол TLS 1.1

Протокол TLS 1.1 [6] оставляет многие функции и способы вычисления сообщений аналогичными TLS 1.0. К ним относятся функция PRF, сообщения согласования общих секретов ke, финальные сообщения fin, сообщения верификации сертификата клиента crt_{verif}, порядок вычисления ms и ключевого материала $keyb$. Версия TLS 1.1 устанавливается в значение 0x0302. Тем не менее параметры TLS 1.1 теперь задокументированы в реестре IANA. Для алгоритма RSA введена поддержка новой кодировки сообщений, направленной на защиту от атак Блейхенбахера [7], включены сертификаты X.509v3 [8].

Порядок криптографических преобразований в TLS 1.1 при шифровании данных приложения остается MtE. Функция поточного шифрования совпадает с функцией TLS 1.0. Однако, в отличие от предыдущих версий протокола, в целях предотвращения атак, описанных в [9], TLS 1.1 вводит явный вектор инициализации iv в единственном поддерживаемом режиме блочного шифрования CBC. После генерации вектор инициализации передается в зашифрованном виде как первый блок шифртекста:

$$iv = r \oplus mask,$$
$$data_{\mathrm{app}} = E_{k_w}(iv \parallel comp(data) \parallel \text{HMAC-H}(\ldots)).$$

Значение $mask$ соответствует 0 или предыдущему CBC-блоку шифртекста: если r получен из криптографически нестойкого PRNG, то $mask$ равно CBC-блоку предыдущего шифртекста; если r получен из криптографически стойкого PRNG, то $mask = 0$. Один непредсказуемый iv используется для одной пары «открытый текст — MAC-код». В остальном функция безопасности протокола записи соответствует TLS 1.0.

TLS 1.1 исключил использование экспортных шифронаборов и добавил поддержку AES-128/256, однако оставил единственный режим работы блочных шифров CBC. Для TLS 1.1 обязателен криптонабор TLS-RSA-WITH-3DES-EDE-CBC-SHA. Приняты

меры от атак на паддинг в режиме CBC, поэтому изменён порядок обработки ошибок, согласно которому в случае ошибочного заполнения используется оповещение о некорректном MAC-коде сообщения взамен предупреждения некорректного расшифрования. В остальном оповещения об ошибках совпадают с TLS 1.0.

1.5. Протокол TLS 1.2

Первый цикл TLS 1.2 [10], как и в предыдущих версиях, содержит расширения ext, которые теперь объединены в одну спецификацию [11]. Зачастую они содержат информацию о поддерживаемых парах «хеш-функция — алгоритм подписи». Версия TLS 1.2 имеет значение 0x0303.

В TLS 1.2 для RSA-подписи используется схема RSASSA-PKCS1-v1.5 [7] с одним хешированием вместо пары MD5/SHA1 в ранних версиях. Подписываемое сообщение содержит информацию о применяемом дайджесте в DER-кодировке. Более ранние версии TLS не включали информацию о дайджесте при кодировании сообщений. Шифрование RSA выполняется с помощью схемы RSAES-PKCS1-v1.5, также определённой в [7]. Применяемые алгоритмы могут задаваться в расширении ext.

В TLS 1.2 структурно функция PRF упрощена и использует только одну функцию HMAC. Секрет s не разделяется на части, поэтому суммирование отдельных промежуточных результатов исключается. Теперь PRF задаётся явным образом криптонаборами в фазе согласования, хотя по умолчанию определена функция HMAC-SHA256. Входные параметры (s, L, $seed$ и a_i) аналогичны PRF из TLS ранних версий:

$$\mathrm{PRF}(s, L, seed) = \text{HMAC-H}(s, a_1 \parallel L \parallel seed) \parallel \ldots \parallel \text{HMAC-H}(s, a_i \parallel L \parallel seed). \qquad (5)$$

Функции поточного и блочного шифрования соответствуют TLS 1.1, однако алгоритмы вычисления MAC-кодов сообщений заменяются более стойкими.

В TLS 1.2 добавлена поддержка аутентифицированного шифрования с ассоциированными данными (AEAD, Authenticated Encryption with Associated Data), представленная режимами CCM (Counter with Cipher Block Chaining — Message Authentication Code) и GCM (Galois/Counter Mode) [12, 13]. На вход AEAD-алгоритма поступают ключ записи k_w, одноразовый номер $nonce$, данные приложения $data$ с необходимым дополнением, а также ассоциированные данные ad (Associated Data), состоящие из номера сообщения SN, типа $comp_{\text{type}}$ и версии $comp_{\text{ver}}$ функции компрессии и длины сжатых данных $comp(data)_{\text{len}}$. Как правило, построение одноразовых номеров $nonce$ включает явную и неявную части [14]. Ключ записи MAC-кода в AEAD-режиме не применяется. Логически AEAD-функцию можно представить в следующем виде:

$$ad = (SN \parallel comp_{\text{type}} \parallel comp_{\text{ver}} \parallel comp(data)_{\text{len}}),$$
$$data_{\text{AEAD}} = \text{AEAD}(k_w, nonce, data, ad).$$

Ключевой блок TLS 1.2 вырабатывается с использованием новой функции PRF (5), входные параметры аналогичны предыдущим версиям протокола. В случае AEAD-шифрования явная часть $nonce$ принимает значение iv_c или iv_s из ключевого блока $keyb$. Структурно функция вычисления $keyb$ соответствует функции (3) для TLS 1.0 и TLS 1.1.

Если явно не указан другой тип сертификата сервера, например PGP [15], то он имеет тип X.509v3 [8]. Публичные ключи сертификатов должны соответствовать выбранному алгоритму обмена ключами. В TLS 1.2 сертификат открытого ключа одного алгоритма подписи может подписываться с использованием другого алгоритма подписи, например ключ RSA может подписываться ключом DSA.

Протокол TLS 1.2 начинает поддерживать криптографию на эллиптических кривых [16]. Для таких алгоритмов обмена ключами должно указываться явно, отправляется ли сообщение ke или нет. Если отправляется, то оно содержит требуемые для обмена $prems$ публичные параметры. Сообщения ke структурно аналогичны предыдущим версиям TLS, однако поддерживают новые алгоритмы, представленные криптонаборами TLS 1.2.

Запрос на сертификат клиента, помимо поддерживаемых типов сертификатов и информации о центрах сертификации, теперь включает список пар «хеш-функция — алгоритм подписи», которые сервер в состоянии верифицировать. Если у клиента отсутствуют необходимые сертификаты, то он отправляет пустой список сертификатов, что отличается от ранних TLS.

Состав и длина предварительного секрета аналогичен ранним версиям TLS, т. е. $prems = (v_c \parallel rnd_c)$. Однако теперь, если $prems$ не проходит проверку, то он рандомизируется и для стороннего наблюдателя дальнейшие вычисления происходят, как если бы $prems$ был корректен. Это необходимо для предотвращения атак на отформатированные согласно PKCS#1 сообщения, направленных на выяснение того, является ли предполагаемый противником секрет $prems$ верным [17, 18]. На основании [18] этих уязвимостей можно избежать путём обработки неверно отформатированных сообщений и несоответствий номеров версий неотличимым от правильно отформатированных RSA-сообщений способом за счёт инициализации $prems$ новыми значениями.

Для TLS 1.1 и выше вводятся правила обработки сообщений $prems$ с более жёстким контролем версий. Если после расшифрования RSA-блока заполнение и длина корректны и v_c равна TLS 1.1 и выше, то $prems = (v_c \parallel rnd_c)$. Для версий v_c ниже или равных TLS 1.0 и отключенных проверках v_c, $prems$ принимает значение расшифрованного RSA-блока. При некорректных заполнениях или длинах расшифрованных RSA-блоков формируется новый $prems = (v_c \parallel R)$ или $prems = R$, где R — 46 или 48 случайных байт сервера. Если v_c не соответствует версии из первого цикла квитирования, то R может быть 48 байт.

Эта техника получила название «Ослепление RSA». В любом случае, сервер TLS 1.2 не должен генерировать предупреждение, если обработка RSA-зашифрованного сообщения $prems$ завершается ошибкой или номер версии оказался неожиданным. Вместо этого он должен продолжать рукопожатие на основе случайно сгенерированного секрета $prems$ как в случае, если бы ошибка отсутствовала. Ошибка в $prems$ будет установлена клиентом, когда он создаст недопустимый главный секрет ms.

Сообщения верификации сертификата клиента crt_{verif} для TLS 1.2 также рассчитываются на основе всех предыдущих сообщений рукопожатия с использованием подписей RSA или DSA, однако теперь для RSA могут применяться любые согласованные алгоритмы хеширования, не противоречащие сертификату. Для DSA определена только функция SHA1, хотя ожидаются изменения [19], которые позволят применять как другие хеши, так и связывать отдельные хеши с разными длинами ключей DSA.

Расчёт финальных сообщений отличается от (4). Двойное хеширование заменено на однократное, соответствующее применяемому в HMAC для PRF (5). Иными словами, хеш-функция H в сообщении fin теперь задаётся криптонаборами явно, в то время как в ранних версиях TLS были жёстко предписаны MD5/SHA1. Длина fin устанавливается согласно шифронабору, но если она не задана явно, составляет 12 байт. Функция расчёта fin имеет следующий вид:

$$fin_{c/s} = \text{PRF}(ms, \text{«client/server finished»}, H(hsmsg)).$$

Протокол TLS 1.2 по сравнению с другими версиями исключает из применения алгоритмы IDEA и DES, включает поддержку SHA256 и принципиально новых схем шифрования AEAD [14], представленных режимами CCM и GCM [12, 13]. Кроме того, добавлен функционал криптографии на эллиптических кривых [16], в котором, например, ECDSA теперь может использоваться с дайджестами, отличными от SHA1. Обязательным шифронабором для TLS 1.2 является TLS-RSA-AES-128-CBC-SHA. В TLS 1.2 добавлено критическое предупреждение для случая, в котором клиент получает ответ от сервера с расширением, отсутствовавшим в запросе клиента.

1.6. Протокол TLS 1.3

Механизм согласования версии TLS 1.2 устарел в пользу списка версий в расширении. В TLS 1.3 [20] расширение s_v (Supported Versions) используется клиентом для указания списка поддерживаемых версий, а сервером — для указания используемой версии протокола. Если оно присутствует в начальном цикле, то сервер не использует версию клиента v_c для согласования. Сервер, который согласовывает версии до TLS 1.3, не отправляет такое расширение. При согласовании TLS 1.3 отправляется $v_s = 0\text{x}0303$ и $s_v = 0\text{x}0304$.

Полное рукопожатие сокращено до трёх циклов. Опционально, первый цикл может включать параметры эллиптической кривой или конечного поля для обмена ключами $keys$ (Key Share), алгоритмы подписи sa (Signature Algorithms), предварительно распределённый ключ PSK (Pre-Shared Key) и режим обмена общими ключами $pskkem$ (PSK Key Exchange Modes) в виде (EC) DHE, PSK или PSK с (EC) DHE. Второй цикл может использовать защищённые данные приложения $data_{\text{app}}$ и обязательно включает зашифрованные расширения ext_{enc} (Encrypted Extensions):

1) $C \to S:\ \ v_c, r_c, sid_c, cs_c, keys^*, sa^*, pskkem^*, PSK^*$;
2) $C \leftarrow S:\ \ v_s, r_s, sid_s, cs_s, keys^*, PSK^*, ext_{\text{enc}}, crt^*_{\text{req}}, crt^*_s, crt^*_{\text{verif}_s}, fin_s, data^*_{\text{app}}$;
3) $C \to S:\ \ crt^*_c, crt^*_{\text{verif}_c}, fin_c$.

Сервер может инициировать перезапрос параметров (EC) DHE для выработки нового общего ключа. Тогда протокол принимает вид

1) $C \leftarrow S:\ \ keys$;
2) $C \to S:\ \ v_c, r_c, sid_c, cs_c, keys$;
3) $C \leftarrow S:\ \ v_s, r_s, sid_s, cs_s, keys, ext_{\text{enc}}, crt^*_{\text{req}}, crt^*_s, crt^*_{\text{verif}_s}, fin_s, data^*_{\text{app}}$;
4) $C \to S:\ \ crt^*_c, crt^*_{\text{verif}_c}, fin_c$.

В процессе квитирования TLS 1.3 выполняет аутентификацию за счёт конкатенации сообщений m_i, $i = 1, 2, \ldots$, на основе хеширования H_T (Transcript-Hash). В перечень m_i в порядке их легитимного поступления входят: $ClientHello$, $HelloRetryRequest$, новый $ClientHello$, $ServerHello$, ext_{enc}, crt_{req_s}, crt_s, crt_{verif_s}, fin_s, ed_{end} (End Early Data), crt_c, crt_{verif_c}, fin_c. В единственном случае — при перезапросе $ClientHello$ — m_1 заменяется специальным синтетическим сообщением. Это сделано для возможности сервера перезапустить квитирование без сохранения состояния, фиксируя только хеш $ClientHello$ в расширении cookie:

$$H_T(m_1, m_2, \ldots, m_i) = H(m_1 \parallel m_2 \parallel \ldots \parallel m_i). \tag{6}$$

Данные cookie также могут использоваться сервером для требования подтверждения доставки сообщений от клиента как защиты от DoS-атак. Для повторного квитирования TLS 1.3 предусматривает отправку фиксированного значения от SHA256

во втором цикле квитирования. Для защиты от атак на понижение версии 8 младших байт r_s устанавливаются в одно из двух специальных значений, определённых отдельно для попыток согласования TLS 1.2 или TLS 1.1 и более ранних версий.

В TLS 1.3 все сообщения рукопожатия после приветствия сервера второго цикла, а также расширения ext_{enc} зашифрованы. Для верификации сертификата crt_{verif} теперь подписываются 64 байта 0x20 для предотвращения атак с выбранным значением r_c на подписи ранних TLS, строка-константа отправителя сообщения, разделительный байт 0x00 и дайджест от функции (6):

$$sig_{s/c} = (\text{0x20 } \ldots \text{ 0x20} \,\|\, \text{«TLS 1.3, server/client CertificateVerify»} \,\| \\ \| \text{ 0x00} \,\|\, H_T(hsmsg, crt_{s/c})).$$

Выработка всех ключей основана на базовом примитиве HKDF. Вычисление финишного сообщения TLS 1.3 производится за счёт HMAC и HKDF [21] на основе базового ключа k_b, константы, длины хеша H_{len}, сообщений протокола рукопожатия, а также сертификата crt и его верификации crt_{verif} (при их наличии), как определено в следующих функциях:

$$k_f = \text{HKDF}(k_b, \text{«finished»}, \text{«»}, H_{\text{len}}),$$
$$fin = \text{HMAC-H}(k_f, H_T(hsmsg, crt, crt_{\text{verif}})).$$

Добавлена поддержка передачи сообщений рукопожатия после выполнения главного рукопожатия, зашифрованных на ключах и алгоритмах трафика приложения. Так, сообщения *newst* (New Session Ticket) создают уникальную связь между значениями билетов и ключами PSK, полученными из возобновлённого главного секрета ms_{res}. Шифронабор восстанавливаемой сессии должен включать HKDF [21] исходной сессии. Ключи PSK могут использоваться для будущих рукопожатий и открытия нескольких параллельных соединений. Состав билета *newst*: 32-разрядное время жизни *time* в секундах (но не более 7 суток), 32-разрядное случайное значение *age* для скрытия истинного возраста билета (суммируется клиентом с *time* $(\text{mod } 2^{32})$ и включается в PSK), одноразовый номер *nonce*, идентификатор *ticket* для ключа PSK и расширение (только данные 0-RTT). Для каждого билета ставится в соответствие уникальный ключ PSK, вычисленный применением функции с дополнительной меткой HKDF_L из [21]:

$$newst = (time \,\|\, age \,\|\, nonce \,\|\, ticket \,\|\, ext),$$
$$PSK = \text{HKDF}_L(ms_{\text{res}}, \text{«resumption»}, nonce, H_{\text{len}}).$$

Вводится опция повышения производительности под названием «нулевое время возобновления приёма-передачи», или 0-RTT (Zero Round Trip Time Resumption), предназначенная для передачи защищённых ранних данных до окончания квитирования. В этом случае клиент использует ранние данные *ed* (Early Data) и расширение для *PSK*, необходимое при согласовании идентификатора предварительно установленного общего ключа, который будет использоваться с данным рукопожатием как указатель на ключ.

Расширение *PSK* состоит из метки ключа, возраста ключа, списка идентификаторов ключей клиента и списка взаимосвязанных пар «HMAC — ключ *PSK*» (Binders), где HMAC связывает *PSK* с конкретным рукопожатием. Каждый *PSK* связан с одним алгоритмом хеширования. Для *PSK*, установленного через билетный механизм,

выбирается HKDF соединения. Для *PSK*, установленного внешним способом, алгоритм хеширования устанавливается одновременно с *PSK* или по умолчанию SHA256. Версия TLS 1.3 фиксирует KDF для связки с *PSK*, однако TLS 1.2 применяет PRF. Протокол квитирования для создания общего ключа *PSK* принимает следующий вид:

1) $C \to S:\ \ v_c, r_c, sid_c, cs_c, keys;$
2) $C \leftarrow S:\ \ v_s, r_s, sid_s, cs_s, keys, ext_{\text{enc}}, crt^*_{\text{req}}, crt^*_s, crt^*_{\text{verif}_s}, fin_s, data^*_{\text{app}};$
3) $C \to S:\ \ crt^*_c, crt^*_{\text{verif}_c}, fin_c;$
4) $C \leftarrow S:\ \ newst.$

Для возобновления сессии на основе выработанного общего *PSK* выполняются три цикла вида

1) $C \to S:\ \ v_c, r_c, sid_c, cs_c, keys^*, PSK;$
2) $C \leftarrow S:\ \ v_s, r_s, sid_s, cs_s, PSK, keys^*, ext_{\text{enc}}, fin_s, data^*_{\text{app}};$
3) $C \to S:\ \ fin_c.$

Для передачи ранних данных до завершения квитирования выполняются следующие циклы:

1) $C \to S:\ \ v_c, r_c, sid_c, cs_c, ed, keys^*, pskkem, PSK, data^*_{\text{app}};$
2) $C \leftarrow S:\ \ v_s, r_s, sid_s, cs_s, PSK, keys^*, ext_{\text{enc}}, ed^*, fin_s, data^*_{\text{app}};$
3) $C \to S:\ \ ed_{\text{end}}, fin_c.$

Криптографические параметры данных 0-RTT поступают совместно с используемым ключом *PSK*. Если ключ *PSK* установлен с помощью билета новой сессии *newst*, то криптографические параметры поступают из согласованного соединения, которое установило *PSK*. Все данные сообщений первого цикла, исключая текущий список пар «HMAC — ключ *PSK*», обрабатываются аналогично финальному сообщению, но с ключом *PSK*. Сообщение fin_s является последним сообщением, зашифрованным с помощью ключа *PSK* для 0-RTT, после него ключ заменяется.

Примечательно, что в протоколе TLS 1.3 нет встроенной защиты от атак воспроизведения для 0-RTT, поэтому необходим дополнительный механизм ограничения количества повторов. Спецификация TLS 1.3 предлагает комбинацию из трёх контрмер. Первая — это одноразовые билеты или их использование в качестве ссылок на базы данных с *PSK*. Вторая — запись уникальных значений r_c и связующих HMAC для *PSK* в пределах заданного временного окна для выявления дубликатов. Третья — это контроль свежести $ClientHello$ на основе времени отправки сообщений. Сервер может включать в билет время его создания, суммированное с допустимым временем жизни. Клиент вычисляет возраст билета как разность между ticket-age-add из билета и obfuscated-ticket-age из *PSK*. В общем случае следует отправлять только такие данные 0-RTT, которые безопасны для воспроизведения.

Новая концепция шифронаборов TLS 1.3 отделяет механизмы аутентификации и обмена ключами от алгоритмов защиты записи (включая длину секретного ключа), одна хэш-функция используется для вывода ключа и MAC-кода сообщений. Удалено сжатие данных. Для защиты данных используются исключительно AEAD-режимы. Ассоциированные данные состоят из типа записи $content_{\text{type}}$ (шифронабор, рукопожатие, данные приложения и т. д.), устаревшей версии записи (0x0303 соответствует

TLS 1.2) и длины блока зашифрованного текста. Данные приложения *data* содержат тип контента, версию протокола, длину и фрагмент данных. В блок открытого текста *pt* входят данные *data*, тип контента и заполнение нулевыми байтами zeros. Для расшифрования в функцию AEAD вместо *pt* подаются $data_{\text{AEAD}}$. Одноразовое значение *nonce* (не менее 8 байт) строится на основе 64-разрядного порядкового номера SN, дополненного нулевыми байтами в старших разрядах до длины вектора инициализации, которое суммируется (mod 2) с ключом записи k_w. В отличие от TLS 1.2, *nonce* для TLS 1.3 не разделяется на части и полностью вычисляется явно:

$$ad = (content_{\text{type}} \parallel v \parallel ct_{\text{len}}),$$
$$data = (content_{\text{type}} \parallel v \parallel \text{len} \parallel \text{fragment}),$$
$$pt = (data \parallel content_{\text{type}} \parallel \text{zeros}),$$
$$data_{\text{AEAD}} = \text{AEAD}(k_w, nonce, ad, pt).$$

Выработка ключей TLS 1.3 происходит за счёт функций расширения HKDF_{EXP}, экстрагирования HKDF_{EXT} и расширения с дополнительной меткой HKDF_L [21], в которых используются хеш-функции, согласованные криптонаборами. Входными источниками энтропии являются секреты *PSK* и общий секрет (EC) DHE. По сравнению с ранними версиями, TLS 1.3 заменяет PRF на HKDF и использует более сложный механизм расчёта для большего количества ключей. Также определена дополнительная функция выработки секрета DS (Derive Secret):

$$\text{HKDF}_L(s, L, context, \text{len}) = \text{HKDF}_{\text{EXP}}(s, \text{len} \parallel \text{«tls13»} \parallel L \parallel context, \text{len}),$$
$$\text{DS}(s, L, hsmsg) = \text{HKDF}_L(s, L, H_T(hsmsg), H_{\text{len}}).$$

Все основополагающие ключевые материалы TLS 1.3 взаимосвязаны криптографически и последовательно вытекают друг из друга путём вычисления в определённом порядке. В первую очередь вычисляется ранний секрет s_{early}, затем ключ функции HMAC для идентификации *PSK* k_{binder}, ключ клиента данных 0-RTT/экспортер главного секрета 0-RTT $k_{\text{early}}/ms^e_{\text{early}}$, промежуточное значение ds_1, главный секрет рукопожатия s_{HS}, секрет трафика рукопожатия клиента/сервера $s_{\text{HS}_{c/s}}$, промежуточное значение ds_2, главный секрет ms, секрет трафика приложения клиента/сервера $s_{\text{app}_{c/s}}$ и, наконец, экспортер главного секрета/возобновляемый главный секрет $ms_{\text{exp/res}}$. В спецификации зафиксированы значения const. Значение «0» обозначает строку из H_{len} байтов 0x00. Примечательно, что если *PSK* не используется, то ранний секрет s_{early} вычисляется из двух нулевых строк:

$$s_{\text{early}} = \text{HKDF}_{\text{EXT}}(\text{«0»}, PSK);$$
$$k_{\text{binder}} = \text{DS}(s_{\text{early}}, \text{const}_1, \text{«»});$$
$$k_{\text{early}}/ms^e_{\text{early}} = \text{DS}(s_{\text{early}}, \text{const}_{2/3}, ClientHello); \quad (7)$$
$$ds_1 = \text{DS}(s_{\text{early}}, \text{const}_4, \text{«»});$$
$$s_{\text{HS}} = \text{HKDF}_{\text{EXT}}(ds_1, \text{(EC) DHE}); \quad (8)$$
$$s_{\text{HS}_{c/s}} = \text{DS}(s_{\text{HS}}, \text{const}_{5/6}, ClientHello \ \ldots \ ServerHello); \quad (9)$$
$$ds_2 = \text{DS}(s_{\text{HS}}, \text{const}_7, \text{«»});$$
$$ms = \text{HKDF}_{\text{EXT}}(ds_2, \text{«0»});$$
$$s_{\text{app}_{c/s}} = \text{DS}(ms, \text{const}_{8/9}, ClientHello \ \ldots \ fin_s); \quad (10)$$
$$ms_{\text{exp/res}} = \text{DS}(ms, \text{const}_{10/11}, ClientHello \ \ldots \ fin_{s/c}). \quad (11)$$

Статические наборы шифров RSA и DH удалены из TLS 1.3. Режим обмена общими ключами определяется клиентом. Если он предлагает *PSK*, сервер отправляет новый билет сессии из предложенных режимов. При форматировании согласованный ключ DHE в (8) дополняется ведущими нулями до требуемого размера, что отличается от предыдущих версий TLS. Для кривых secp256r1, secp384r1 и secp521r1 вычисление выполняется согласно [22] по схеме ECKAS-DH1, выработанный секрет представлен X-координатой общей точки эллиптической кривой в виде строки октетов. Для кривых x25519 и x448 работа основана на [23], выход используется без дополнительной коррекции.

Итоговые ключи и векторы инициализации генерируются на основе переменных предыдущих вычислений, в которых для 0-RTT данных значение s соответствует (7), для квитирования (9), для данных приложения (10). Обновление секрета трафика после окончания квитирования инициируется соответствующим запросом (Key Update) и основывается на энтропии предыдущих секретов:

$$k_w = \mathrm{HKDF}_L(s, \text{«key»}, \text{«»}, k_{\text{len}}),$$
$$iv_w = \mathrm{HKDF}_L(s, \text{«iv»}, \text{«»}, iv_{\text{len}}),$$
$$s_{\text{app}(n+1)} = \mathrm{HKDF}_L(s_{\text{app}(n)}, \text{«traffic upd»}, \text{«»}, H_{\text{len}}).$$

Экспортируемый ключевой материал генерируется на основе секрета s, принятого из (7) или (11). Отсутствие и пустое значение *context* дают эквивалентные результаты, что отличает TLS 1.3 от предыдущих версий, в которых результаты не совпадали:

$$\mathrm{TLS}_{\text{EXP}}(L, context, k_{\text{len}}) = \mathrm{HKDF}_L(DS(s, L, \text{«»}), \text{«exporter»}, H(context), k_{\text{len}}).$$

Версия TLS 1.3 включает пять шифронаборов, первые три обязательны: TLS-AES-128-GCM-SHA256, TLS-AES-256-GCM-SHA384, TLS-CHACHA20-POLY1305-SHA256, TLS-AES-128-CCM-SHA256, TLS-AES-128-CCM-8-SHA256. К обязательным схемам цифровых подписей для сертификатов относятся RSA-PKCS1-SHA256, для сертификатов и их верификации RSA-PSS-RSAE-SHA256, а также ECDSA-secp256r1-SHA256. Обязательные эллиптические кривые для обмена ключами secp256r1, x25519 [23].

Зафиксирована поддержка дополнительных схем подписи RSASSA-PKCS1-v1.5, ECDSA на основе кривых secp384r1 и secp521r1, RSASSA-PSS (RSAE и PSS) совместно с SHA256, SHA384 и SHA512, а также EdDSA на основе эллиптических кривых Эдвардса ed25519 и ed448. Схемы RSA-PKCS1 и ECDSA совместно с SHA1 применяются исключительно в целях обратной совместимости. Протокол TLS 1.3 подписывает параметры используемых эллиптических кривых в ECDSA, а также взамен согласования делает выбор в пользу одной точки для одной кривой. Запрещается MD5, SHA224, DSA и сертификаты OpenPGP. Список оповещений о предупреждениях и ошибках расширен до 34 вариантов.

Важно отметить, что на базе прототипа TLS 1.3 ведётся разработка протокола RuTLS [24]. Модификация касается изменений логики взаимодействия участников протокола в различных режимах, используемых криптографических примитивов и шифронаборов, а также системы выработки ключей за счёт поддержки криптографических схем Лимонник-3 [25], Крокус [26] и Эхинацея-2(3) [27], разработанных в Российской Федерации.

В связи с нацеленностью RuTLS на повышение безопасности соединений исключается режим 0-RTT. Данные приложений передаются только после завершения процесса аутентификации, использование RawKeys исключено. Генерация секретных и открытых случайных значений использует только разные генераторы случайных чисел.

Новая ключевая система протокола позволяет формировать различные ключи шифрования/имитозащиты с использованием исключительно групп точек эллиптической кривой, согласование общей точки EC DH обязательно. Генерация ключей применяет идентификаторы участников протокола, извлекаемые из сертификатов согласно [27].

2. Развитие криптографического протокола IPsec: IKE, AH, ESP

2.1. Протокол IKEv1

Спецификация IKEv1 [28] представляет собой гибридный протокол, использующий функционал протоколов Oakley [29], SKEME [30] и ISAKMP [31], предназначенный для согласования ассоциации безопасности SA (Security Association) и выработки аутентифицированного ключевого материала в защищённом виде для последующего использования в AH/ESP.

Фаза 1 протокола IKEv1 предназначена для создания безопасного аутентифицированного канала связи ISAKMP SA, посредством которого происходит защищённый информационный обмен для фазы 2, отвечающей за согласование криптографических параметров различных сервисов, таких, как IPsec. Фаза 1 характеризуется криптографическими параметрами только для канала ISAKMP (IKEv1). Следующие атрибуты используются IKEv1 и согласовываются как часть ISAKMP SA: алгоритм шифрования, алгоритм хеширования для PRF/HMAC, метод проверки подлинности, группа DH.

Основной режим (Main Mode) включает два сообщения согласования политики обмена, два сообщения ценностей DH и *nonce* (n), а также два сообщения аутентификации параметров DH. Агрессивный режим (Aggressive Mode) включает два сообщения согласования политики обмена (аутентификация ответчика во втором сообщении), два сообщения ценностей DH (третье сообщение подтверждает подлинность инициатора и его участие в обмене) и вспомогательных данных DH, возведение в степень допустимо отложить до переговоров.

Main Mode и Aggressive Mode разрешают четыре метода аутентификации: цифровую подпись, две формы аутентификации с асимметричным шифрованием и PSK. Секретное значение k_s вычисляется отдельно для каждого метода аутентификации на основе *nonce* инициатора и респондента (n_I, n_R), общего секрета DH (g^{xy}), cookie (cookie_I, cookie_R) и PSK. Ключи k_s, определенные в порядке для случаев применения в подписи, асимметричном шифровании и PSK, представлены ниже. Функция H обозначает согласованную хеш-функцию:

$$\begin{gathered} k_s = \text{PRF}(n_I \parallel n_R, g^{xy}), \\ k_s = \text{PRF}(H(n_I \parallel n_R), \text{cookie}_I \parallel \text{cookie}_R), \\ k_s = \text{PRF}(PSK, n_I \parallel n_R). \end{gathered} \tag{12}$$

В результате работы Main Mode и Aggressive Mode вырабатываются три группы взаимосвязанного аутентифицированного ключевого материала: k_{s_d} используется для выработки ключей, например для IPsec, k_{s_a} применяется в ISAKMP SA для аутентификации сообщений, k_{s_e} — в ISAKMP SA для конфиденциальности сообщений. Сразу после выработки указанных ключей информационный обмен IKEv1 становится криптографически защищённым:

$$\begin{gathered} k_{s_d} = \text{PRF}(k_s, g^{xy} \parallel \text{cookie}_I \parallel \text{cookie}_R \parallel \text{0x00}), \\ k_{s_a} = \text{PRF}(k_s, k_{s_d} \parallel g^{xy} \parallel \text{cookie}_I \parallel \text{cookie}_R \parallel \text{0x01}), \\ k_{s_e} = \text{PRF}(k_s, k_{s_a} \parallel g^{xy} \parallel \text{cookie}_I \parallel \text{cookie}_R \parallel \text{0x02}). \end{gathered}$$

Аутентификация основывается на цифровых подписях, асимметричных алгоритмах и *PSK* за счёт значений H_I и H_R, полученных как дайджесты от всей полезной нагрузки, включая тип идентификатора, порт и протокол, но исключая общий заголовок. Применяется всё тело полезной нагрузки *SA*, исключая общий заголовок ISAKMP, а также полезная нагрузка для идентификации (id_I, id_R):

$$H_I = \mathrm{PRF}(k_s, g^{x_I} \parallel g^{x_R} \parallel \mathrm{cookie}_I \parallel \mathrm{cookie}_R \parallel SA_I \parallel id_I); \tag{13}$$

$$H_R = \mathrm{PRF}(k_s, g^{x_R} \parallel g^{x_I} \parallel \mathrm{cookie}_R \parallel \mathrm{cookie}_I \parallel SA_I \parallel id_R). \tag{14}$$

В фазе 1 аутентификация на основе подписей *sig* с использованием сертификатов открытого ключа *crt* происходит над значениями H_I (13) и H_R (14), при этом Main Mode включает шесть циклов. Если алгоритм подписи привязан к определённому алгоритму хэширования (например, DSS использует SHA с 160-битным выходом), то используется версия HMAC зафиксированного алгоритма хэширования. Согласованные PRF и дайджест используются для всех других псевдослучайных функций. Как и прежде, символ «*» сигнализирует об опциональном параметре, *hdr* обозначает заголовки ISAKMP согласно [31]:

$$\begin{aligned}
&1) \quad I \to R: \quad hdr_I, SA_I;\\
&2) \quad I \leftarrow R: \quad hdr_R, SA_R;\\
&3) \quad I \to R: \quad hdr_I, ke_I, n_I;\\
&4) \quad I \leftarrow R: \quad hdr_R, ke_R, n_R;\\
&5) \quad I \to R: \quad E_{k_{s_e}}(hdr_I, id_I, crt_I^*, sig_I(H_I));\\
&6) \quad I \leftarrow R: \quad E_{k_{s_e}}(hdr_R, id_R, crt_R^*, sig_R(H_R)).
\end{aligned}$$

Для аутентификации на основе шифрования Main Mode применяет одноразовые номера n и идентификаторы id, защищённые на открытом ключе получателя e. В этом случае циклы 3–6 заменяются другими циклами рукопожатия. Аутентификация на основе *PSK* заменяет циклы 5 и 6 на циклы, содержащие id_I (H_I) и id_R (H_R), в которых k_{s_e} является производным от *PSK* из (12):

$$\begin{aligned}
&3') \quad I \to R: \quad hdr_I, ke_I, H(crt_R), E_{e_R}(id_I), E_{e_R}(n_I);\\
&4') \quad I \leftarrow R: \quad hdr_R, ke_R, E_{e_I}(id_R), E_{e_I}(n_R);\\
&5') \quad I \to R: \quad E_{k_{se}}(hdr_I), H_I;\\
&6') \quad I \leftarrow R: \quad E_{k_{se}}(hdr_R), H_R.
\end{aligned}$$

При аутентификации на основе изменённого режима шифрования *nonce* зашифровывается с использованием открытого ключа получателя e. Однако обмен DH (ke), id и crt зашифровываются с использованием согласованного симметричного алгоритма из *SA* на симметричном ключе k, выведенном из *nonce*. Это решение экономит две дорогостоящих операции с открытым ключом на каждой стороне. Здесь циклы $3'$ и $4'$ заменяются:

$$\begin{aligned}
&3'') \quad I \to R: \quad hdr_I, H(crt_R), E_{e_R}(n_I), E_{k_I}(ke_I), E_{k_I}(id_I), E_{k_I}(crt_I);\\
&4'') \quad I \leftarrow R: \quad hdr_R, E_{e_I}(n_R), E_{k_R}(ke_R), E_{k_R}(id_R).
\end{aligned}$$

В Aggressive Mode аутентификация на основе подписей представлена тремя циклами, в которых заголовки hdr передаются в открытом виде:

$$\begin{aligned}
&1) \quad I \to R: \quad hdr_I, SA_I, ke_I, n_I, id_I;\\
&2) \quad I \leftarrow R: \quad hdr_R, SA_R, ke_R, n_R, id_R, crt_R^*, sig_R;\\
&3) \quad I \to R: \quad hdr_I, crt_I^*, sig_I.
\end{aligned}$$

Aggressive Mode с аутентификацией на основе шифрования с открытым ключом также состоит из трёх циклов:

$$\begin{aligned}
&1) \quad I \to R: \quad hdr_I, SA_I, H(crt_R), ke_I, E_{e_R}(id_I), E_{e_R}(n_I);\\
&2) \quad I \leftarrow R: \quad hdr_R, SA_R, ke_R, E_{e_I}(id_R), E_{e_I}(n_R), H_R;\\
&3) \quad I \to R: \quad hdr_I, H_I.
\end{aligned}$$

Aggressive Mode с аутентификацией на основе изменённого режима шифрования принимает следующий вид:

$$\begin{aligned}
&1) \quad I \to R: \quad hdr_I, SA_I, H(crt_R), E_{e_R}(n_I), E_{k_I}(ke_I), E_{k_I}(id_I), E_{k_I}(crt_I);\\
&2) \quad I \leftarrow R: \quad hdr_R, SA_R, E_{e_I}(n_R), E_{k_R}(ke_R), E_{k_R}(id_R), H_R;\\
&3) \quad I \to R: \quad hdr_I, H_I.
\end{aligned}$$

При аутентификации с помощью сертификатов и шифрования на открытом ключе отсутствуют доказательства факта информационного обмена между I и R, так как обе стороны могут полностью реконструировать сообщения.

В Aggressive Mode аутентификация на основе PSK представлена значениями H_I и H_R:

$$\begin{aligned}
&1) \quad I \to R: \quad hdr_I, SA_I, ke_I, n_I, id_I;\\
&2) \quad I \leftarrow R: \quad hdr_R, SA_R, ke_R, n_R, id_R, H_R;\\
&3) \quad I \to R: \quad hdr_I, H_I.
\end{aligned}$$

Симметричные ключи шифрования k являются эфемерными и вычисляются из величин n_{E_I} и n_{E_R} развертыванием до необходимой длины. В режиме CBC вектор инициализации iv первого блока равен нулю, для последующих блоков — последнему блоку шифртекста. Паддинг состоит из 0x00 и оканчивается значением количества дополненных байт:

$$\begin{gathered}
n_{E_{I/R}} = \mathrm{PRF}(n_{I/R}, \mathrm{cookie}_{I/R}),\\
k = (k_1 \parallel k_2 \parallel k_3 \parallel \ldots),\\
\text{где}\ \ k_1 = \mathrm{PRF}(n_{E_{I/R}}, 0\mathrm{x}00),\ k_2 = \mathrm{PRF}(n_{E_{I/R}}, k_1),\ k_3 = \mathrm{PRF}(n_{E_{I/R}}, k_2),\ \ldots
\end{gathered}$$

В фазе 2, или Quick Mode, согласуются дочерние SA и передаются одноразовые номера n для защиты от воспроизведения и создания нового ключевого материала, в параметре ke может согласовываться секрет DH. Допускается одновременное согласование нескольких экземпляров SA.

Quick Mode без ke только обновляет ключевой материал по данным фазы 1 и не предоставляет PFS (Perfect Forward Secrecy), в то время как DH обеспечивает PFS. Протокол ниже описывает Quick Mode, где id клиентов используются для направления

трафика в соответствующий туннель, а идентификатор сообщения m_{id} принимается из заголовка ISAKMP. В режиме CBC для первого сообщения Quick Mode вектор iv вырабатывается (согласованной) хеш-функцией от конкатенации последнего блока CBC фазы 1 и идентификатора сообщения m_{id} фазы 2. Последний блок CBC фазы 1 сохраняется в состоянии ISAKMP SA для синхронизации iv при их запросе в Quick Mode и информационном обмене:

$$\begin{aligned}
&1) \quad I \to R: \quad E_k(hdr_I), H_1, SA_I, n_I, ke_I^*, id_I^*, id_R^*;\\
&2) \quad I \leftarrow R: \quad E_k(hdr_R), H_2, SA_R, n_R, ke_R^*, id_I^*, id_R^*;\\
&3) \quad I \to R: \quad E_k(hdr_I), H_3.
\end{aligned}$$

Значения H_1, H_2 и H_3 вычисляются в следующем виде:

$$\begin{gathered}
H_1 = \mathrm{PRF}(k_{s_a}, m_{id_I} \parallel SA_I \parallel n_I \parallel ke_I^* \parallel id_I^* \parallel id_R^*),\\
H_2 = \mathrm{PRF}(k_{s_a}, m_{id_R} \parallel n_I \parallel SA_R \parallel n_R \parallel ke_R^* \parallel id_I^* \parallel id_R^*),\\
H_3 = \mathrm{PRF}(k_{s_a}, 0\mathrm{x}00 \parallel m_{id_I} \parallel n_I \parallel n_R).
\end{gathered}$$

Ключевой материал дочерней SA строится с применением индекса параметров безопасности SPI (Security Parameter Index). Если требуется свойство PFS, то ke (g^{xy}) принимается в расчёт:

$$k_m = \mathrm{PRF}(k_{s_d}, g^{xy*} \parallel Proto \parallel SPI \parallel n_I \parallel n_R).$$

При необходимости ключевой материал может расширяться:

$$k_m = (k_1 \parallel k_2 \parallel k_3 \parallel \ldots),$$

где

$$\begin{gathered}
k_1 = \mathrm{PRF}(k_{s_d}, g^{xy*} \parallel Proto \parallel SPI \parallel n_I \parallel n_R),\\
k_2 = \mathrm{PRF}(k_{s_d}, k_1 \parallel g^{xy*} \parallel Proto \parallel SPI \parallel n_I \parallel n_R),\\
k_3 = \mathrm{PRF}(k_{s_d}, k_2 \parallel g^{xy*} \parallel Proto \parallel SPI \parallel n_I \parallel n_R), \ \ldots
\end{gathered}$$

По умолчанию поддерживается две группы конечных полей и две группы эллиптических кривых. Доступен режим New Group Mode для согласования новых параметров DH, который запускается в конце фазы 1. Доступна передача только идентификатора группы или кривой. Циклы New Group Mode имеют следующий вид:

$$\begin{aligned}
&1) \quad I \to R: \quad E_k(hdr_I), H_1, SA_I;\\
&2) \quad I \leftarrow R: \quad E_k(hdr_R), H_2, SA_R.
\end{aligned}$$

Параметры H_1 и H_2 представляются в виде

$$\begin{aligned}
H_1 &= \mathrm{PRF}(k_{s_a}, m_{id_I} \parallel SA_I),\\
H_2 &= \mathrm{PRF}(k_{s_a}, m_{id_R} \parallel SA_R).
\end{aligned}$$

Реализации IKEv1 должны поддерживать DES-CBC с контролем слабых и полуслабых ключей [32], 3DES, MD5, SHA, Tiger, аутентификацию на основе *PSK* и DSA, подписи и аутентификацию на основе RSA. По умолчанию для DH определены 768- и 1024-разрядные группы (mod p), 155- и 185-разрядные группы эллиптических кривых. Также могут поддерживаться IDEA-CBC, Blowfish-CBC, RC5-R16-B64-CBC, CAST-CBC. Примечательно, что источник [28] не определяет структуру функции PRF.

2.2. Протокол IKEv2

Протокол IKEv2 [33] объединяет ранние документы [28, 31, 34] и поддерживает работу через NAT. В этом случае туннелируемые пакеты инкапсулируются в UDP для идентификации конечных узлов по номеру порта. IKEv2 заменяет восемь различных начальных обменов IKEv1 одним обменом с четырьмя сообщениями. Механизм аутентификации реализован на основе единственного поля $auth$ взамен рапределения частей данных аутентификации по всему обмену IKEv1. В рамках противодействия атакам на понижение, помимо явного указания версий в сообщениях, новая спецификация вводит дополнительный флаг оповещения о поддержке более высоких версий IKE.

В базовом обмене первые два цикла теперь носят название инициализации IKE (IKE INIT). Здесь hdr содержит SPI, номера версий и флаги. В SA указываются криптографические параметры инициатора для IKE SA, ke содержит значения DH. Одноразовые номера n должны быть не короче 128 бит и не короче половины длины ключа PRF. Ответчик выбирает криптонабор SA_{R_1} из списка инициатора SA_{I_1}. Запрос на сертификат crt_{req} включает список дайджестов SHA1 открытых ключей доверенных центров сертификации и представляет собой предложение сертификатов, а не требование их использования. На этом этапе каждая сторона может сгенерировать ключевое зерно $seed_k$ для выработки ключей шифрования величин ke и ключей аутентификации k_a для IKE SA, а также ключ k_d для последующего использования в AH и ESP. Все последующие сообщения, в том числе информационные, защищены согласованными алгоритмами и выработанными ключами.

Третий и четвёртый циклы отвечают за аутентификацию IKE (IKE AUTH). Идентичности сторон подтверждаются значениями id, доказывается знание секретов k_e и k_a, за целостность данных отвечает поле $auth$. Сертификаты crt и список корневых центров сертификации crt_{req} опциональны. При наличии сертификат используется для проверки целостности $auth$. Опция id_R позволяет инициатору указать ответчика переговоров, например когда один IP-адрес содержит несколько идентичностей. Согласование IPsec SA начинается с параметров SA_{I_2}.

Для обмена информацией из SPD (Security Policy Database) введены селекторы трафика ts (Traffic Selector), которые содержат параметры пакетов нового SA: диапазон IP-адресов, диапазон портов, идентификатор протокола IP.

Два заключительных цикла предлагают SA, одноразовые номера, опциональные значения DH и селекторы ts, которые связывают трафик с конкретной SA. Если, например, служба AH или ESP требует новые ключи, отличные от существующих в IKE SA, то в предпоследнем цикле указывается (опциональное) соответствующее информационное сообщение Inf. Также оно может обеспечивать динамическое выделение адресов для удалённого доступа через шлюз IPsec. Селекторы трафика опускаются, если поступил запрос на изменение ключа IKE SA. Все IKE SA идентифицируются по одному полю — SPI:

$$
\begin{aligned}
&1)\quad I \to R: \quad hdr_I, SA_{I_1}, ke_I, n_I;\\
&2)\quad I \leftarrow R: \quad hdr_R, SA_{R_1}, ke_R, n_R, crt^*_{\text{req}};\\
&3)\quad I \to R: \quad hdr_I, E_{k_e,k_a}(id_I, crt^*_I, crt^*_{\text{req}}, id^*_R, auth, SA_{I_2}, ts_I, ts_R);\\
&4)\quad I \leftarrow R: \quad hdr_R, E_{k_e,k_a}(id_R, crt^*_R, auth, SA_{R_2}, ts_I, ts_R);\\
&5)\quad I \to R: \quad hdr_I, E_{k_e,k_a}(Inf^*, SA_I, n_I, ke^*_I, ts^*_I, ts^*_R);\\
&6)\quad I \leftarrow R: \quad hdr_R, E_{k_e,k_a}(SA_R, n_R, ke^*_R, ts^*_I, ts^*_R).
\end{aligned}
$$

В рамках аутентификации инициатор подписывает данные первого цикла, объединённые с n_R и $\mathrm{PRF}(k_{P_I}, id_I)$, а ответчик — данные второго цикла, объединённые с n_I и $\mathrm{PRF}(k_{P_R}, id_R)$ (ключи k_{P_I} и k_{P_R} описаны ниже). Новые группы DH и cookie включаются в подпись (при их наличии). Инициатор и ответчик могут использовать общие ключи или открытые ключи из сертификатов независимо друг от друга в любой комбинации. Для общего ключа аутентификатор $auth$ вычисляется так:

$$auth = \mathrm{PRF}(\mathrm{PRF}(k_{P_{I/R}}, \text{«Key Pad for IKEv2»}), \text{цикл } 1/2).$$

Протокол IKEv2 вводит расширяемые методы аутентификации EAP (Extensible Authentication Protocol) [35]. В этом случае циклы 3–5 базового обмена претерпевают изменения. Протокол EAP обычно используется для аутентификации инициатора совместно с аутентификацией ответчика на основе подписи. Запрос на EAP объявляется исключением поля $auth$ из соответствующего сообщения квитирования. Если респондент желает использовать EAP, он включает соответствующие данные в ответный цикл. Параметры SA_{R_2}, ts_I, ts_R передаются после прохождения аутентификации. Если EAP создает общий ключ как побочный эффект аутентификации, то он используется (только) для генерации поля $auth$ в заключительных циклах, иначе применяются ключи k_{P_I} и k_{P_R}. EAP без выработки общего ключа небезопасны [36].

В EAP определены типы для идентификаторов, уведомлений, в том числе в случае отсутствия поддержки запрашиваемого метода аутентификации, MD5-вызовов, одноразовых паролей (OTP) и общей токен-карты (GTC). Успешная аутентификация EAP индуцируется как EAP*:

$$\begin{aligned}
&3') \quad I \to R: \quad hdr_I, E_{k_e,k_a}(id_I, crt^*_{\mathrm{req}}, id^*_R, SA_{I_2}, ts_I, ts_R);\\
&4') \quad I \leftarrow R: \quad hdr_R, E_{k_e,k_a}(id_R, crt^*_R, auth, \mathrm{EAP});\\
&5') \quad I \to R: \quad hdr_I, E_{k_e,k_a}(\mathrm{EAP});\\
&6') \quad I \leftarrow R: \quad hdr_R, E_{k_e,k_a}(\mathrm{EAP}^*);\\
&7') \quad I \to R: \quad hdr_I, E_{k_e,k_a}(auth);\\
&8') \quad I \leftarrow R: \quad hdr_R, E_{k_e,k_a}(auth, SA_{R_2}, ts_I, ts_R).
\end{aligned}$$

Информационные обмены ведутся в защищённом виде на основе установленных *SA* для IKE или AH/ESP. При закрытии IKE *SA* все дочерние *SA* для AH/ESP автоматически закрываются. При закрытии AH/ESP *SA* удаляются только они. Для удаления *SA* отправляется запрос с индексом *SPI*.

Введены таймеры повторной передачи, при которых инициатор запоминает запрос до получения ответа. Для увеличения пропускной способности определена опциональная поддержка параллельной обработки нескольких ожидающих сообщений не в хронологическом порядке следования, но в пределах выделенного окна. Ответчик запоминает ответ до получения нового запроса с номером, большим, чем в ответе, просуммированным с размером окна порядковых номеров. Для сопоставления запросов / ответов и определения повторных сообщений формат IKEv2 предлагает 32-разрядные идентификаторы как часть заголовков hdr, равные 0 для первых запросов каждого направления (по два счётчика на каждую сторону). Смена ключей IKE *SA* сбрасывает порядковый номер.

Компрессия [37] настраивается как часть IPsec *SA*, однако отдельно от согласования криптографических параметров. Ассоциации сжатия уничтожаются совместно с AH/ESP *SA*.

В рамках снижения эффективности DoS-атак ответчик может минимизировать использование CPU и исключить хранение SA до подтверждения IP-адреса отправителя. Для этого cookie специального вида включаются в обмен новой инициализации IKE. Однако долгое хранение cookie снижает потенциал защиты от DoS-атак:

$$\text{cookie} = (ver_s \parallel H(n_I \parallel \text{IP}_I \parallel SPI_I \parallel secret)).$$

Включение SPI_I гарантирует разные cookie для разных SA, n_I гарантирует невозможность создания cookie без знания первого цикла базового обмена, $secret$ известен только респонденту, а версия ver_s изменяется при обновлении секрета. Ответ от инициатора должен содержать аналогичную нагрузку. Для описанного случая между циклами 1 и 2 базового обмена добавляются следующие обмены:

$$\begin{aligned} 1'') \quad & I \leftarrow R: \quad hdr_R, \text{cookie}; \\ 2'') \quad & I \rightarrow R: \quad hdr_I, \text{cookie}, SA_{I_1}, ke_I, n_I. \end{aligned}$$

Респондент сравнивает отправленный и принятый cookie. Совпадение означает, что данные созданы после последнего изменения секрета, что гарантирует их актуальность. Кроме того, IP_I должен совпадать с исходным адресом, с которым соединялся респондент.

В IKEv1 время жизни SA согласовывалось, в IKEv2 каждая сторона ведёт собственную политику времени жизни и смены ключей SA. IKEv2 разрешает параллельные SA с одинаковыми селекторами трафика для поддержки различий качества обслуживания (QoS). В отличие от IKEv1, комбинация конечных точек и селекторов трафика однозначно не идентифицирует SA.

Протокол IKEv2 согласует алгоритмы шифрования, целостности, группы DH и PRF, причём PRF используется для вывода ключевого материала как в IKE SA, так и в IPsec SA. В случае длинных ключей применяется итеративное вычисление:

$$\text{PRF}'(k, s) = (t_1 \parallel t_2 \parallel t_3 \parallel \ldots),$$
$$\text{где} \qquad t_1 = \text{PRF}(k, s \parallel \text{0x01}),\ t_2 = \text{PRF}(k, t_1 \parallel s \parallel \text{0x02}),\ t_3 = \text{PRF}(k, t_2 \parallel s \parallel \text{0x03}), \ldots$$

На основе величины $seed_k$ вычисляются семь секретных ключей: k_d — материал для вывода новых ключей в IPsec SA; k_{a_I} и k_{a_R} — ключи целостности алгоритма аутентификации последующих обменов; k_{e_I} и k_{e_R} — ключи симметричного шифрования; k_{P_I} и k_{P_R} — ключи для вычисления поля $auth$. Изменение ключей изменяет SPI. Для нового SA материал $seed'_k$ вычисляется с использованием существующего k_d, обновлённых $nonce$ и, опционально, новых секретов DH. Итеративная функция вырабатывает ключи в определенном порядке (используется $seed_k$ или $seed'_k$). Если PRF принимает ключи фиксированной длины, то конечные биты n_I и n_R поровну отбрасываются:

$$seed_k = \text{PRF}(n_I \parallel n_R, g^{xy}); \tag{15}$$
$$seed'_k = \text{PRF}(k_d, g^{xy*} \parallel n_I \parallel n_R); \tag{16}$$
$$(k_d \parallel k_{a_I} \parallel k_{a_R} \parallel k_{e_I} \parallel k_{e_R} \parallel k_{P_I} \parallel k_{P_R}) = \text{PRF}'(seed_k, n_I \parallel n_R \parallel SPI_I \parallel SPI_R). \tag{17}$$

В IKEv2 добавлена поддержка работы через NAT посредством инкапсуляции данных в пакеты UDP для маршрутизации и идентификации конечных узлов по номеру порта. Соответствующие уведомления включаются после n_I и n_R в базовых циклах (1) и (2). Предотвращение специальной или некорректной обработки IPsec в узлах NAT

реализуется перенаправлением трафика через порты 500/4500. Для идентификации вложенных протоколов в случае IKEv2 после заголовка UDP добавляются 4 байта 0x00, в AH/ESP здесь находится *SPI*.

Проверка факта нахождения NAT между узлами (или какой из узлов находится за NAT) осуществляется на основе двух дайджестов SHA1, вычисленных отдельно от *SPI*, IP-адреса и порта инициатора и респондента. Получатель таких уведомлений сравнивает принятый и собственноручно вычисленный дайджесты SHA1. Если они не совпадают, то собеседник находится за NAT (на маршруте произведено изменение адреса источника исходного пакета для соответствия адресу узла NAT) и следует включить обход NAT. В этом случае система должна позволять динамическое обновление адресов и отправку поддерживающих соединение пакетов согласно [38]. Если дайджесты не соответствуют адресам заголовков, то необходимо туннелировать будущие пакеты, связанные с этим IKE *SA*, через UDP порт 4500.

В транспортном режиме изменение IP-адреса вызывает сбой MAC. Узел NAT не в состоянии откорректировать MAC, так как он криптографически защищён. Однако для будущей проверки MAC отправитель вводит IP-адрес в селекторы трафика. В туннельном режиме возникают проблемы с маршрутизацией, где трансляция адресов AH/ESP требует специальной логики в NAT, которая не является тривиальной. Решение также заключается в инкапсуляции трафика IPsec в UDP.

Для определения динамического IP-адреса узла, расположенного за защищённым шлюзом NAT, используется специальный параметр удалённого доступа IPsec, который включается в циклы 3 и 4 основного рукопожатия перед SA_I и SA_R. Запрос нового адреса может быть включен в любой запрос на создание IPsec *SA*. Если, например, NAT перезапускается, то хосты соединяются с IP-адресом и портом последнего аутентифицированного пакета, инкапсулированного в UDP. Такие пакеты используются для динамического (обнаружения) изменения IP-адреса и порта.

Изменение сетевых идентификаторов выявляется проверкой MAC, для которого изменён адрес и порт, связанный с *SA* аутентифицированного пакета. В таких ситуациях следует динамически обновлять комбинации адреса и порта установленной *SA*. Узел за NAT не должен выполнять такое обновление, если проверенные пакеты имеют разные значения адреса и порта, так как это открывает возможность DoS-атакам. Кроме того, динамическое обновление адреса должно выполняться в ответ на новый пакет, иначе злоумышленник может вернуть старые воспроизведённые адреса.

Туннели IPsec на основе IKEv1 при обработке трафика сбрасывают индикацию явного уведомления о перегрузке ECN (Explicit Congestion Notification) во внешних заголовках IP, что приносит ущерб сети. В IKEv2 реализуется полноценная поддержка ECN за счёт введения функциональности туннелей, согласно [39], и обработки данных, согласно [40].

Криптонаборы для IKEv2 включают алгоритмы шифрования DES-IV64, DES, 3DES, RC5, IDEA, CAST, Blowfish, 3IDEA, DES-IV32, AES-CBC, AES-CTR и без конфиденциальности. Функции PRF представлены HMAC-MD5, HMAC-SHA1, HMAC-Tiger, AES-128-XCBC. Целостность обеспечивается HMAC-MD5-96, HMAC-SHA1-96, DES-MAC, KPDK-MD5, AES-XCBC-96. Группы DH представлены 768- и 1024-разрядными конечными полями. Сжатие обеспечено алгоритмами DEFLATE, LZS и LZJH.

2.3. Протокол IKEv3

Спецификация IKEv3 [41] заменяет и обновляет [33], а также включает разъяснения [42]. Взамен опциональности, селекторы трафика становятся обязательными для

всех обменов. В связи со сложностью реализации исключена поддержка установки периода использования внутреннего IP-адреса. Удален атрибут конфигурации адреса сервера имен NetBios. Допускается обработка сообщений с произвольным порядком инкапсуляции полезной нагрузки в отличие от отклонения в предыдущей версии. Выработка ключевого материала *SA* теперь требует обязательного включения секрета DH.

Повторная передача сообщений IKE *SA* возможна в трёх случаях: для полуоткрытого соединения респондент дублирует свой ответ; для запроса новой *SA* — создаёт и отправляет сообщение новой *SA*; для существующей и аутентифицированной *SA* — игнорирует сообщение. В новой спецификации недостаточно использовать только *SPI* и IP-адрес инициатора для определения соединений, открытых в одностороннем порядке, так как два разных респондента за NAT могут выбрать одинаковый *SPI* инициатора. Теперь для поиска IKE *SA* используется всё сообщение, его хеш и полезная нагрузка n_I. В общем случае респондентом целесообразна однократная повторная передача на каждое подобное сообщение.

Разъяснено использование критического флага, рассматриваемого в контексте типа полезной нагрузки, а не её содержимого. IKEv2 [41] добавляет такой флаг к каждому заголовку полезной нагрузки для перспектив гибкости и прямой совместимости. Если критический флаг установлен и тип полезной нагрузки не распознан, то сообщение отклоняется и ответчик сигнализирует о неподдерживаемой полезной нагрузке (в сигнализации критический флаг отсутствует). Если критический флаг не установлен и тип полезной нагрузки также не распознан, то полезная нагрузка игнорируется без дополнительного оповещения.

Если инициатор поддерживает AEAD-режимы, то в *SA* он должен предлагать два варианта криптонаборов: комплекты AEAD-режимов и комплекты обычного шифрования с отдельными алгоритмами целостности. В отличие от IKEv2, согласно [41], при смене ключей IPsec *SA* селекторы трафика и криптонаборы следует оставлять без изменений. Параллельные запросы *SA* удаляются по наименьшему значению *nonce* побайтовым сравнением (не целочисленным).

Если инициатор не замечает одновременную смену ключей и изменяет свою IKE *SA*, а затем получает запрос на смену *SA*, то он оповещает об этом респондента и избыточные *SA* не создаются. Если инициатор замечает одновременную смену ключей и получает запрос на удаление старой IKE *SA*, то он фиксирует, что респондент не обнаружил одновременную смену ключей. В этом случае инициатор останавливает попытку смены ключа. Важно отметить, что IKEv3 не имеет отдельного механизма повторной аутентификации и проходит её только путём создания новой *SA*.

Введены требования предпочтительных длин ключей PRF (например, для секретов k_d, k_{P_I} и k_{P_R}). Для PRF на основе HMAC предпочтительный размер ключа равен длине вывода базовой хеш-функции, другие PRF должны явно его указывать.

Протокол IKEv2 [33] допускал обновление ключевого материала без обмена ke_I и ke_R, однако IKEv3 [41] всегда включает разделяемый секрет DH. Функции вычисления ключевого зерна $seed_k/seed'_k$ используют PRF старой IKE *SA* и совпадают с (15) и (16). Для сеансовых ключей в (17) поступают данные нового обмена, в том числе функция PRF новой IKE *SA*.

Источник [41] расширяет список событий, в которых необходимы оповещения об ошибках и соответствующая реакция сторон. Для ошибок за пределами установленных соединений IKE *SA*, например в базовых циклах 1 и 2, аналогичных IKEv2, во избежание участия в DoS-атаках необходимо ограничивать скорость ответов на крипто-

графически незащищенные уведомления. Необходим компромисс между перегрузкой сети и минимальным количеством повторных сообщений для установления безопасного соединения. Подобные уведомления могут быть результатом сбоя узла и требуют инициирования проверки работоспособности соединений на основе соответствующих IP-адресов, портов, *SPI* и идентификаторов сообщений. В таких ситуациях незамедлительная реакции в виде изменений состояния *SA* не допускается.

Ошибки аутентификации IKE *SA*, например недействительный общий секрет, идентификатор, поставщик сертификата, сертификат и т. д., вызывают критические предупреждения. Ошибки после аутентификации IKE *SA*, указывающие на различие состояний сторон, предполагают обновление IKE *SA*. Если проверка MAC проходит, но синтаксис сообщения некорректен, вырабатывается фатальное уведомление, требующее закрытия IKE *SA*.

При обходе NAT спецификация [41] вводит требование поддержки обработки как инкапсулированных, так и не инкапсулированных в UDP пакетов IPsec. Любая сторона решает независимо, использовать инкапсуляцию IPsec в UDP или нет, однако при обнаружении NAT обе стороны должны ее использовать. Инкапсуляция в UDP не должна выполняться на порте 500, только 4500. Реализации должны обрабатывать полученные пакеты IPsec, инкапсулированные в UDP, даже если NAT отсутствует.

В результате перекрытия нескольких обменов может возникать ситуация рассинхронизации состояния *SA*. С ростом размера окна параллельной обработки положение значительно усугубляется, особенно при обработке запросов вне очереди. Поэтому для работы в условиях подобных коллизий при смене ключей и закрытии (дочерних) *SA* разработаны дополнительные уведомления.

Введено уведомление временной невозможности ответа на запрос, например в период операции смены/удаления ключей или закрытия *SA* (избыточные *SA* удаляются на основе минимальных *nonce*). В результате отправитель ожидает завершения временного состояния корреспондента. Если подобные уведомления происходят на протяжении длительного времени, например в течение нескольких минут, то соединение рассинхронизировано и требует закрытия.

Другое уведомление отправляется при получении запроса на смену ключа несуществующей дочерней *SA*, где реакцией получателя является удаление такой *SA*.

Добавлено требование обязательной поддержки HTTP-метода [43] для поиска сертификатов по двум значениям, ссылке URL и дайджесту URL. Другие методы могут использоваться в случае выпуска соответствующих спецификаций.

Сетевая конфигурация IPv6 способами, основанными на аналогичных данных IPv4, не соответствует нормальному функционированию IPv6. При автоконфигурации IPv6 состояние узла не сохраняется, рекламные сообщения маршрутизатора и обнаружение соседей становятся недоступны. Для решения подобных осложнений введён экспериментальный документ [44].

В криптонаборах [41] расширен список параметров DH, в который вошли конечные поля с разрядностью модулей 1536, 2048, 3042, 4096, 6144 и 8192 бит.

2.4. Протокол AHv1

Согласно [45], заголовок аутентификации AHv1 вычисляется на основе симметричных или асимметричных криптографических алгоритмов и занимает положение непосредственно после заголовка IPv4 и перед заголовками протоколов верхнего уровня. В случае IPv6 AH обычно занимает положение после заголовка параметров обра-

ботки маршрутизации (Hop-by-Hop) или списка необходимых промежуточных узлов (Routing) и перед параметрами получателя (Destination Options).

Хеш-функции H для вычисления ICV (Integrity Check Value) заголовка AH принимаются из SA, на которые ссылаются IP-адрес и SPI. Обязательна поддержка ключевого MD5 [46]. Обычные контрольные суммы, например CRC-16, криптографически нестойкие, поэтому запрещены.

Для расчёта ICV принимаются практически все данные сообщения. Модифицируемые при транзите поля, например TTL и CRC в IPv4 или Hop Limit в IPv6, входят в расчёт ICV как нулевые строки для подтверждения факта их использования и длины, но не контроля содержимого. Синтаксис AH в порядке следования включает 8-разрядный заголовок полезной нагрузки NH (Next Header), 8-разрядную длину данных аутентификации ICV_{len} как количество 32-разрядных слов, 16-разрядный резерв R, 32-разрядный SPI и непосредственно данные аутентификации ICV переменной длины, кратной 32-разрядному слову:

$$P_{\text{AHv1}} = (\text{NH} \parallel ICV_{\text{len}} \parallel R \parallel SPI \parallel ICV).$$

Данные ICV принимаются за нулевую строку при расчёте. Все данные $Data$ протокола верхнего уровня в контексте AH рассматриваются как полезная нагрузка и включаются в расчёт ICV полностью. Дополнительные метки, например IPSO, принимаются в расчёт ICV.

2.5. Протокол AHv2

Выпуск AHv2 [47] объявляет AHv1 [45] устаревшим. В [47] заголовок аутентификации претерпел изменения. AH стал заголовком расширений в случае IPv6. Для противодействия атакам воспроизведения отдельно для отправителя и получателя добавлены 32-разрядные беззнаковые инкрементные счетчики SN в позицию между SPI и ICV. Исключение зацикливания достигается инициализацией $SN = 0$ при установлении новой SA ранее чем через каждые 2^{32} сообщений. Номер SN всегда включается в AH, однако его проверка оставлена на усмотрение получателя.

Задокументированы списки изменяемых полей заголовков IP, которые не входят в ICV. Для IPv4 это Type of Service, Flags, Fragment Offset, TTL, CRC, а также экспериментальные поля. Для IPv6 это Class, Flow Label, Hop Limit, а также Hop-by-Hop и Destination в том случае, если установлен бит, указывающий, что опция изменяема во время транзита. Тип и длина опции входят в ICV.

Для обеспечения кратности длины AH 32 (IPv4) или 64 (IPv6) разрядам ICV дополняются. Паддинг располагается сразу за ICV, содержит произвольные значения и входит в расчет ICV как данные верхнего протокола. Алгоритмы аутентификации устанавливаются SA и могут включать ключевые MAC на основе симметричных алгоритмов шифрования, например DES, односторонних хэш-функций или хэш-функций в сочетании с асимметричными алгоритмами подписи. Последний вариант актуален для многоадресных решений. Обязательными объявляются HMAC-MD5-96 и HMAC-SHA1-96.

Логическая структура пакета AHv1 и функция вычисления ICV представлены ниже. Неизменяемые поля заголовка IP, принимающие участие в расчёте ICV, обозначены как IP_{HDR}:

$$P_{\text{AHv2}} = (\text{NH} \parallel ICV_{\text{len}} \parallel R \parallel SPI \parallel SN \parallel ICV); \tag{18}$$

$$ICV = H(\text{IP}_{\text{HDR}} \parallel \text{NH} \parallel ICV_{\text{len}} \parallel R \parallel SPI \parallel SN \parallel ICV \parallel Data). \tag{19}$$

Итоговые изменения относительно [45] направлены на создание полноценной платформы AH с интегрированной службой защиты от воспроизведения на основе порядковых номеров сообщений *SN*, с более стойкими бесключевыми алгоритмами HMAC-MD5-96 и HMAC-SHA1-96 вместо ключевого MD5, а также с фиксацией списка исключённых из *ICV* изменяемых заголовков IPv4/IPv6. Туннельный режим с аутентификацией всех данных инкапсулируемого пакета теперь рассматривается как неотъемлемая часть AH, в ранней версии он только упоминался.

2.6. Протокол AHv3

Спецификация [48] внесла дополнения в AHv2 [47] и объявила его устаревшим. Структура пакета AHv3 соответствует AHv2 (18) и (19). Для корректной обработки коллизий *SPI* между одноадресными и многоадресными архитектурами, в которых сервер ключей или групповой контроллер назначают групповую ассоциацию безопасности, источник [48] вводит алгоритм поиска *SA* в базе данных SAD (Security Association Database) на основе параметров *SPI*, IP-адреса назначения и IP-адреса источника. При отсутствии совпадений производится поиск без IP-адреса источника; затем только по *SPI*; затем по *SPI* и протоколу AH/ESP. Найденная *SA* применяется для обработки AH.

Новое расширение включает 64-разрядный порядковый номер *ESN* (Extended Sequence Number) для высокоскоростных коммуникаций. Он устанавливается по умолчанию, если, например, при согласовании IKE явно не указан 32-разрядный *SN*. Младшие 32 бита *ESN* явно передаются в поле SN, старшие биты сохраняются в памяти, а при расчёте *ICV* помещаются сразу после полезной нагрузки и перед неявным заполнением. В AH не предусмотрена синхронизация *SN/ESN* для многоадресных *SA*, поэтому в таких решениях отсутствует защита от повторов.

Вводятся примеры реализаций *SN*-окна антивоспроизведения на основе *ESN*, механизмов его управления, вычисления неявных старших бит *ESN*, а также процесс восстановления синхронизации *ESN* при потере сообщений. Если получатель не поддерживает антивоспроизведение, он не должен согласовывать *ESN* в *SA*. Это исключает необходимость контроля старших бит *ESN* в контексте *SN*-окна антивоспроизведения и расчёта *ICV*.

В расчёте *ICV* обнуляются новые непредсказуемые поля. Для IPv4 это флаги дополнительного обслуживания на маршрутизаторах (DSCP) и их перегрузки (ECN). Запрещается внесение паддинга больше минимума, требуемого для выравнивания границ по 32/64 битам.

Для AHv3 алгоритмы целостности *ICV* могут включать ключевые MAC, основанные на симметричных алгоритмах, например AES, или на хеш-функциях, например MD5, SHA1, SHA256. Обязательные криптографические алгоритмы собраны в едином ресурсе [49] и включают HMAC-SHA1-96, AES-XCBC-MAC-96 и HMAC-MD5-96.

2.7. Протокол ESPv1

Спецификация [50] содержит вводную информацию и предполагает использование ESPv1 в транспортном и туннельном режимах, в которых способы преобразования данных идейно схожи с AH. ESPv1 располагается после IP-заголовка и состоит из 32-разрядного *SPI* в открытом виде и защищённых данных переменной длины *Data* (TCP, UDP и т. д.). Контроль целостности в ESPv1 источник [50] не описывает. Кроме того, сообщение может включать дополнительную открытую нагрузку, передаваемую вместе с ESP. Получатель идентифицирует *SA* по IP-адресу назначения и *SPI*. В обычных условиях ESP не включает явных меток безопасности, так как *SPI* указывает на

уровень защиты. Однако при развертывании многоуровневой системы безопасности должно поддерживаться применение явных меток, например IPSO:

$$P_{\text{ESPv1}} = (SPI \parallel E_k(Data)). \tag{20}$$

Если криптографические алгоритмы ESPv1 дополнительно не обеспечивают аутентификацию, то для таких целей возможно применение AH. В транспортном режиме AH аутентифицирует нагрузку ESPv1 (20) и заголовок IP. В туннельном режиме AH аутентифицирует все поля инкапсулированного IP-пакета и результирующая ESP с зашифрованными AH и полным внутренним IP-пакетом принимается в качестве полезной нагрузки для внешнего IP-пакета.

Обязательным для поддержки в ESPv1 объявлен DES-CBC, однако без включения расчета MAC это небезопасно. Порядок криптографических преобразований всех версий ESP соответствует Encrypt-and-MAC (E&M).

2.8. Протокол ESPv2

Источник [51] признает устаревшим ESPv1 [50] и дает описание ESPv2 как законченного решения. Состав пакета ESPv2, включая обязательные и опциональные параметры, фиксируется конкретной *SA* на время её действия.

Поля пакета ESPv2 в порядке их следования теперь включают 32-разрядный *SPI*, который в сочетании с IP-адресом назначения и протоколом ESP однозначно идентифицирует *SA*. Далее следует 32-разрядный порядковый номер сообщения *SN* для противодействия атакам воспроизведения, который сбрасывается в 0 при установке новой *SA*. Затем располагается полезная нагрузка *Data* переменной длины, содержащая, при необходимости, вектор инициализации *iv* для алгоритма шифрования. Вектор инициализации указывается явно вместе с длиной и позицией расположения или неявно, может быть открытым или частью зашифрованных данных. В любом случае учитываются требования спецификаций, регламентирующих такое поведение. Заполнение *pad* используется для кратности размеру блока алгоритма шифрования, а также выравнивания окончания зашифрованной нагрузки на 4-байтовой границе для гарантии того, что данные аутентификации *ICV* выровнены корректно или для сокрытия длины полезной нагрузки. Если не определено иное, то по умолчанию для простейшей защиты целостности байты заполнения инициализируются серией целочисленных монотонно возрастающих значений. Затем следуют 8-разрядные длина заполнения pad_{len} и следующий заголовок NH, идентифицирующий тип полезной нагрузки. Завершает пакет ESPv2 значение целостности *ICV*. Поле *ICV* включается только в том случае, если аутентификация определена *SA*:

$$P_{\text{ESPv2}} = (SPI \parallel SN \parallel iv^* \parallel E_k(iv^{**} \parallel Data \parallel pad \parallel pad_{\text{len}} \parallel \text{NH}) \parallel ICV).$$

Значение *ICV* вычисляется от пакета ESPv2, исключая поле *ICV*. В транспортном режиме в первую очередь выполняется обработка ESP, а затем фрагментация. В туннельном режиме ESP может применяться к фрагментам IP:

$$ICV = H(P_{\text{ESPv2}}).$$

ESPv2 может использоваться в транспортном и туннельном режимах. В транспортном режиме ESP вставляется после IP-заголовка и перед протоколами верхнего уровня (TCP, UDP, ICMP и т. д.) или заголовками IPsec, которые вставлены ранее. Конфиденциальность IP-заголовка не обеспечена. В IPv4 заголовок ESP находится после IP-

заголовка. В IPv6 заголовок ESP рассматривается как сквозная полезная нагрузка и находится после расширений Hop-by-Hop, Routing или Fragmentation. В туннельном режиме ESP защищает весь инкапсулированный IP-пакет, включая конечные адреса и заголовки. Положение ESP в туннельном режиме относительно внешнего незащищённого IP-заголовка аналогично ESP в транспортном режиме. В обоих режимах аутентифицируются данные от *SPI* в начале ESP и до NH в конце ESP, где внешний заголовок IP не защищён. Заголовки ESP и AH можно комбинировать разными способами.

Алгоритмы конфиденциальности и аутентификации согласуются в *SA*. Для *ICV* могут применяться ключевые MAC на основе симметричных алгоритмов, например DES, хэш-функций, например MD5 или SHA1. Для многоадресных решений могут поддерживаться алгоритмы подписи. Тем не менее в ESP аутентификация является необязательной.

Защита от атак воспроизведения выполняется по аналогии с AH за счёт скользящего окна контроля принимаемых порядковых номеров. Правильная реализация антивоспроизведения требует своевременного изменения состояния *SN*, поэтому в таких сценариях в целях недопущения зацикливания счётчика ручное распределение ключей не допускается. Взлом *SPI* обнаруживается с помощью аутентификации ESP, однако IP-адрес получателя и тип протокола IPsec (AH/ESP) не защищены от активных воздействий. Корректность значений дополнения и его длины проверяются независимо от поля аутентификации.

В ESPv2 обязательными для поддержки являются DES-CBC с явным вектором инициализации, HMAC-MD5-96, HMAC-SHA1-96, а также отсутствие аутентификации или шифрования.

2.9. Протокол ESPv3

Источник [52] описывает использование ESPv3 для обеспечения конфиденциальности, целостности или их обоих одновременно. Услуга только конфиденциальности объявлена как возможная, но не обязательная. Обеспечение целостности ESPv3 является привлекательной альтернативой AH, так как в этом случае отмечается рост производительности и возможность конвейерной обработки. В любом случае, служба антиповтора может использоваться только при наличии *ICV*.

Введена новая функция TFC (Traffic Flow Confidentiality), способствующая эффективной генерации и утилизации фиктивного трафика. Для идентификации фиктивных пакетов в Next Header зафиксировано значение 59. Такие пакеты содержат все заголовки, характерные ESP, однако полезная нагрузка может формироваться произвольным образом, например как случайная последовательность. Данное нововведение направлено на маскирование отсутствия реального трафика.

Дана расширенная модель данных полезной нагрузки и *ICV* для работы с AEAD-алгоритмами, которые дополнительно вычисляют *ICV*. В этом случае в пакете ESPv3 поле *ICV* опускается. Необходимо выбирать такие AEAD-алгоритмы, которые способны поддерживать только аутентификацию без конфиденциальности для полей *SPI*, *SN* и старших бит *ESN* или репликацию указанных полей в зашифрованную нагрузку для обеспечения их целостности. В любом случае, в рамках антивоспроизведения защищённый *SN* должен соответствовать открытому номеру *SN*. Добавлена опция для дополнительной поддержки 64-разрядного *ESN* в высокоскоростных решениях (десятки, сотни Гбит/с).

Формат пакета ESPv3 претерпел изменения. Поля *SPI* и *SN*, расширенное до *ESN*, предоставляют функционал, полностью аналогичный AH [48], включая механизмы

антивоспроизведения и скользящего *SN*-окна. Контроль *SN* выполняется до проверки целостности и расшифрования, что устраняет затраты на криптографические вычисления в случае повтора. При необходимости вектор iv помещается в начале полезной нагрузки $Data$ в незашифрованном виде. Алгоритм шифрования должен дать чёткую спецификацию о структуре iv, его местоположении и способе обработки. Логическая структура пакета ESPv3 с зашифрованными на сеансовом ключе k данными представлена в следующем виде:

$$P_{\text{ESPv3}} = (SPI \parallel ESN^{*} \parallel iv^{*} \parallel E_k(iv^{**} \parallel Data \parallel \text{TFC}^{*} \parallel pad \parallel pad_{\text{len}} \parallel \text{NH}) \parallel ICV).$$

Значение ICV вычисляется от пакета ESPv3, исключая поле ICV. Если используется расширение *ESN*, то для расчёта ICV старшие 32 разряда порядкового номера помещаются за полем NH:

$$ICV = H(P_{\text{ESPv3}}).$$

Дополнение *pad* ограничено 255 байтами, что недостаточно для скрытия характеристик трафика. Если верхний протокол содержит информацию о длине $Data$, то для решения такой задачи TFC производит вставки после полезной нагрузки. Это верно для туннельного и частично верно для транспортного режима, в зависимости от протокола. Использование TFC согласуется *SA* и отличается от фиктивных пакетов большей вариативностью действий. Остальные поля соответствуют ESPv2.

Перечень криптографических алгоритмов ESPv3 внесён в единую спецификацию [49]. Для конфиденциальности требуется поддержка 3DES-CBC, AES-CBC (128-разрядные ключи), AES-CTR и опция без конфиденциальности, DES-CBC не рекомендуется. Обеспечение целостности требует реализации усеченных дайджестов HMAC-SHA1-96, AES-XCBC-MAC-96, возможна HMAC-MD5-96 и опция без целостности.

Заключение

С опорой на базовые спецификации RFC в работе рассмотрены основные этапы развития криптографических протоколов транспортного и сетевого уровней модели OSI на примерах выпусков SSL версий 2.0 и 3.0 [1, 3], TLS версий от 1.0 до 1.3 [4, 6, 10, 20], а также IPsec версий от 1 до 3, состоящих из IKE [28, 33, 41], AH [45, 47, 48] и ESP [51 – 53] соответственно.

Криптопротокол SSL 2.0 рассмотрен в качестве исторической справки как опорная точка первого представителя линейки SSL/TLS, в то время как SSL 3.0 предоставил платформу для разработки последующих версий TLS. Протоколы SSL/TLS характеризуются порядком криптографических преобразований EtM.

Протокол SSL 3.0 включает рукопожатия создания и восстановления безопасных соединений, обеспечивает подписи RSA/DSA дайджестов MD5/SHA1 открытых криптографических параметров и асимметричное шифрование предварительных секретов. Расчёт главного секрета и ключевого материала обеспечен многократным вычислением хеш-функций MD5/SHA1 от случайных и константных величин. Аутентификация сообщений рукопожатия обеспечивается финальными сообщениями. Поддерживаются небезопасные криптографические алгоритмы DES, Fortezza, IDEA, MD5, RC2, RC4, SHA1, включая экспортные реализации. Режим работы блочных шифров представлен CBC.

Версия TLS 1.0 добавляет параметры расширения в протокол квитирования. Обычное хеширование заменено на функцию PRF, использующую HMAC на основе MD5/SHA1. Относительно SSL 3.0 протокол TLS 1.0 значительных архитектурных изменений не претерпел.

В TLS 1.1 введено новое форматирование сообщений в рамках защиты от атак Блейхенбахера. Для защиты от атак на режим CBC вводится явный вектор инициализации блочных шифров. В шифронаборы добавлен алгоритм AES-128/256, исключены экспортные реализации. В случае некорректного дополнения блочного шифра ошибка расшифрования заменена на оповещение сбоя MAC-кода.

Протокол TLS 1.2 делает акцент на задействование расширений, зачастую указывающих на поддерживаемые комбинации хеш-функций и связанных с ними алгоритмов подписи. Схема подписи RSA заменяет двойной дайджест MD5/SHA1 однократным. Сертификат открытого ключа одного алгоритма может подписываться с использованием другого алгоритма, например ключ RSA, подписанный схемой DSA. В отличие от TLS 1.0 и TLS 1.1, функция PRF TLS 1.2 явно согласуется криптонаборами, структурно упрощена использованием одного HMAC и отсутствием разделения секрета на части. По умолчанию PRF задается HMAC-SHA256. Исключены алгоритмы IDEA, DES. Рост безопасности TLS 1.2 обусловлен поддержкой эллиптических кривых, а также схем AEAD, представленных режимами CCM/GCM. Для финальных сообщений применяется хеш-функция, согласованная криптонаборами, ранние релизы SSL/TLS фиксировали MD5/SHA1. Отключено предупреждение об ошибке предварительного секрета, который рандомизируется в случае некорректности и обрабатывается неотличимым от правильно отформатированных сообщений способом, устраняя утечку информации.

Квитирование TLS 1.3 сокращено до трёх циклов, поддерживаемые расширения и данные после приветствия сервера защищены криптографически. Защита от атак на понижение версий обеспечивается включением фиксированных значений в случайные данные сервера, определённых отдельно для попыток согласования TLS 1.2 или TLS 1.1 и более ранних версий. Добавлена поддержка и согласование режимов обмена PSK, TLS 1.3 фиксирует KDF для связки с PSK. Функционал 0-RTT обеспечивает передачу защищенных ранних данных приложений на PSK до окончания квитирования. В рамках предотвращения атак с выбранным рандомным значением клиента на подписи ранних TLS изменена структура сообщений верификации сертификатов. Выработка ключевого материала TLS 1.3 использует более сложный механизм расчёта для большего количества секретов, где PRF заменена на HKDF. Перерасчёт ключей происходит после финишного сообщения. Финишные сообщения используют HMAC совместно с HKDF. Протокол TLS 1.3 чётко разграничивает механизмы аутентификации и обмена ключами от алгоритмов защиты записи, одна хэш-функция используется для вывода ключей и MAC. Данные приложений защищены исключительно режимами AEAD. Асимметричные алгоритмы определены (EC) DHE, RSA, ECDSA, EdDSA. Применение эллиптической криптографии ставит упор на использование фиксированной надёжной точки кривой взамен её согласования. Статические RSA и DH, алгоритм DSA и сертификаты OpenPGP, а также MD5 и SHA224 запрещены TLS 1.3. Дайджест SHA1 отмечен только для целей обратной совместимости.

Протокол RuTLS, построенный на основе TLS 1.3, корректирует логику взаимодействия участников соединения, используемые алгоритмы/криптонаборы и систему выработки ключей, призванные повысить безопасность системы. Наложены ограничения на использование 0-RTT, введены требования на порядок передачи данных приложений, генерации случайных чисел/криптографических ключей и используемые при этом идентификаторы. Изменения основаны на включении поддержки криптографических схем Лимонник-3, Крокус и Эхинацея-2(3) Российской Федерации.

Протокол IPsec прошёл три основных этапа развития. Протокол IKEv1 поддерживает DES, 3DES, IDEA, Blowfish, RC5, CAST в режиме CBC, а также MD5, SHA1 и Tiger в PRF. Методы аутентификации основаны на PSK, DSA, RSA, DH и ECC с использованием PRF, структура которой не определяется. Поддерживается аутентификация с одновременным использованием симметричных и асимметричных методов. Ключевой материал выводится функцией PRF на основе переданных идентификаторов, одноразовых номеров, выработанных секретов, значений DH и других параметров. Включение DH обеспечивает свойство PFS. Секретные ключи AH/ESP генерируются PRF по данным, полученным в предыдущих сообщениях.

Механизм аутентификации IKEv2 применяет общие ключи, PRF и опирается на единственное одноименное поле взамен распределения по всему обмену IKEv1. Зафиксированы требования к одноразовым номерам. Введены селекторы трафика, содержащие параметры пакетов *SA*. Добавлены расширяемые методы аутентификации EAP, однако без выработки общего ключа они небезопасны. В рамках борьбы с DoS-атаками описан пример реализации cookie-данных на основе одноразовых номеров, IP, *SPI* и секрета. Выработка ключевого материала фиксирует порядок вычисления секретов аутентификации и шифрования. Спецификация IKEv2 добавляет возможности работы через NAT. Нахождение за NAT идентифицируется на основе дайджеста SHA1 от идентификаторов участников взаимодействия. Введён способ определения динамического IP участника, расположенного за NAT. Реализована полноценная поддержка явного уведомления о перегрузке сети ECN. В криптонаборы добавлены алгоритмы AES в режимах CBC/CTR. Функции целостности и PRF представлены HMAC на основе MD5, SHA1, Tiger, AES-128-XCBC, DES-MAC, KPDK-MD5. Сжатие содержит DEFLATE, LZS и LZJH.

Протокол IKEv3 делает обязательным применение селекторов трафика, обновление ключевого материала всегда включает разделяемый секрет DH. Допускается произвольный порядок инкапсуляции полезной нагрузки. Идентификация *SA* использует всё сообщение и его дайджест. Разъяснено применение критического флага контроля типа полезной нагрузки. Введены требования предпочтительных длин ключей PRF. Даже при отсутствии NAT требуется поддержка обработки инкапсулированных и не инкапсулированных в UDP пакетов. Криптокомбинации разделены на комплекты с режимами AEAD и обычным шифрованием с отдельными алгоритмами целостности. Расширен список событий при аутентификации, различии состояния сторон, проверки MAC, в которых необходимы оповещения об ошибках. Преодоление коллизий и рассинхронизации состояний *SA* обеспечено добавлением уведомлений временной невозможности ответа на запрос и отсутствия дочерних ассоциаций безопасности.

Формирование заголовка AHv1 происходит за счёт хеш-функции, согласованной *SA*. Для расчёта *ICV* принимаются неизменяемые поля, модифицируемые поля обнуляются. Обязательна поддержка ключевого MD5, корректирующие коды исключены. Выпуск AHv2 стал самодостаточной платформой с интегрированной службой защиты от воспроизведения на основе порядковых номеров, обязательными функциями HMAC-MD5-96/HMAC-SHA1-96, фиксированными списками исключённых из расчёта *ICV* изменяемых данных IP. Выравнивание обеспечено паддингом, расположенным за *ICV*. Туннельный режим рассматривается как неотъемлемая часть AHv2. В AHv3 корректная обработка коллизий *SPI* между одноадресными и многоадресными архитектурами решена новым алгоритмом поиска *SA* в базе данных SAD. Новое расширение включает 64-разрядный порядковый номер *ESN* и способы его управления. Внесение паддинга больше достаточного минимума запрещено. Обязательны HMAC-

SHA1-96, AES-XCBC-MAC-96 и HMAC-MD5-96. Порядок криптографических преобразований IPsec соответствует E&M.

Обработка ESPv1 идейно схожа с AH. Целостность ESPv1 не описывается. Обязательным объявлен алгоритм DES-CBC. Версия ESPv2 описана как законченное решение. Добавлен 32-разрядный порядковый номер сообщений, управляемый аналогично AH. Вектор инициализации включается явно или неявно, заполнение используется для кратности размеру блока алгоритма шифрования. Поле *ICV* вычисляется от пакета ESPv2. Обязательными для поддержки являются DES-CBC с явным вектором инициализации, HMAC-MD5-96, HMAC-SHA1-96. Обеспечение целостности за счёт ESPv3 отмечено как более производительная альтернатива AH. Введена функция TFC для маскирования отсутствия реального трафика. Описана модель данных полезной нагрузки и *ICV* для работы с режимами AEAD, которые способны отдельно вычислять *ICV*. Механизм защиты от повторов использует ESN и совпадает с методами AHv3. Дополнение ограничено 255 байтами. Криптографические алгоритмы ESPv3 содержат 3DES-CBC, AES-CBC/CTR, а DES-CBC не рекомендуется. Целостность требует реализации HMAC-SHA1-96, AES-XCBC-MAC-96 и возможность HMAC-MD5-96. Заголовки ESP и AH можно комбинировать в разных вариантах.

В рассмотренных протоколах безопасности прослеживается тенденция отказа от применения нескольких слабо исследованных и нестойких криптографических функций, например для задач расчёта ключевого материала, обеспечения аутентификации и конфиденциальности, в пользу использования одного исследованного и надёжного алгоритма или криптографического параметра. Вместе с тем развитие криптопротоколов SSL/TLS и IPsec значительно увеличивает количество вариантов их состояний, характеризуемых комбинациями криптографических величин, применяемых алгоритмов, оповещений об ошибках и других параметров, что в значительной степени усложняет анализ безопасности системы.

ЛИТЕРАТУРА

1. *Kipp E. B. H.* The SSL Protocol. Netscape Communications Corp., 1995 (Expires 10 / 95). 26 p. `https://tools.ietf.org/html/draft-hickman-netscape-ssl-00`.
2. *Polk T. and Turner S.* Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176. Internet Engineering Task Force (IETF), 2011. 4 p. `https://tools.ietf.org/html/rfc6176`.
3. *Freier A., Karlton P., and Kocher P.* The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101. Internet Engineering Task Force (IETF), 2011. 67 p. `https://tools.ietf.org/html/rfc6101`.
4. *Allen C. and Dierks T.* The TLS Protocol Version 1.0. RFC 2246. Network Working Group, 1999. 80 p. `https://tools.ietf.org/html/rfc2246`.
5. *Kaliski B.* PKCS#1: RSA Encryption Standard, version 1.5. RFC 2313. Network Working Group, 1998. 19 p. `https://tools.ietf.org/html/rfc2313`.
6. *Dierks T. and Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346. Network Working Group, 2006. 87 p. `https://tools.ietf.org/html/rfc4346`.
7. *Jonsson J. and Kaliski B.* Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447. Network Working Group, 2003. 72 p. `https://tools.ietf.org/html/rfc3447`.
8. *Ford W., Housley R., Polk W., and Solo D.* Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280. Network Working Group, 2002. 129 p. `https://tools.ietf.org/html/rfc3280`.

9. http://www.openssl.org/~bodo/tls-cbc.txt — Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures, 2004.
10. *Dierks T. and Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. Network Working Group, 2008. 104 p. https://tools.ietf.org/html/rfc5246.
11. *Eastlake D. 3rd.* Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066. Internet Engineering Task Force (IETF), 2011. 25 p. https://tools.ietf.org/html/rfc6066.
12. *Dworkin M.* Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C, 2004. 27 p.
13. *Dworkin M.* Recommendation for Block Cipher Modes of Operation: Galois / Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007. 39 p.
14. *McGrew D.* An Interface and Algorithms for Authenticated Encryption. RFC 5116. Network Working Group, 2008. 22 p. https://tools.ietf.org/html/rfc5116.
15. *Mavrogiannopoulos N.* Using OpenPGP Keys for Transport Layer Security (TLS) Authentication. RFC 5081. Network Working Group, 2007. 8 p. https://tools.ietf.org/html/rfc5081.
16. *Blake-Wilson S., Bolyard N., Gupta V., et al.* Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492. Network Working Group, 2006. 35 p. https://tools.ietf.org/html/rfc4492.
17. *Bleichenbacher D.* Chosen ciphertext attacks against protocols based on RSA Encryption Standard PKCS#1 // CRYPTO'98. LNCS. 1998. V. 1462. P. 1–12.
18. *Klima V., Pokorny O., and Rosa T.* Attacking RSA-based Sessions in SSL/TLS. Cryptology ePrint Archive: Report 2003/052, 2003. 23 p.
19. https://csrc.nist.gov/publications/detail/fips/186/3/archive/2009-06-25 — Digital Signature Standard (DSS). NIST FIPS PUB 186-3, 2009. 131 p.
20. *Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. Internet Engineering Task Force (IETF), 2018. 160 p. https://tools.ietf.org/html/rfc8446.
21. *Eronen P. and Krawczyk H.* HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869. Internet Engineering Task Force (IETF), 2010. 14 p. https://www.rfc-editor.org/info/rfc5869.
22. https://standards.ieee.org/standard/1363-2000.html — IEEE Standard Specifications for Public Key Cryptography. IEEE Std 1363-2000, 2000. 236 p.
23. *Hamburg M., Langley A., and Turner S.* Elliptic Curves for Security. RFC 7748. Internet Research Task Force (IRTF), 2016. 22 p. https://tools.ietf.org/html/rfc7748.
24. *Гребнев С. В., Лазарева Е. В., Лебедев П. А. и др.* Интеграция отечественных протоколов выработки общего ключа в протокол TLS 1.3 // Прикладная дискретная математика. Приложение. 2018. № 11. С. 62–65.
25. *Матюхин Д. В.* О некоторых свойствах схем выработки общего ключа, использующих инфраструктуру открытых ключей, в контексте разработки стандартизированных криптографических решений. 2011. https://www.ruscrypto.ru/resource/archive/rc2011/files/02_matyukhin.pdf.
26. *Нестеренко А. Ю.* Об одном подходе к построению защищенных соединений // Математические вопросы криптографии. 2013. № 4:2. С. 101–111.
27. *Гребнев С. В.* О возможности стандартизации протоколов выработки общего ключа. РусКрипто, М., 2014. https://www.ruscrypto.ru/resource/archive/rc2014/files/03_grebnev.pdf.
28. *Carrel D. and Harkins D.* The Internet Key Exchange (IKE). RFC 2409. Network Working Group, 1998. 41 p. https://tools.ietf.org/html/rfc2409.

29. *Orman H.* The Oakley Key Determination Protocol. RFC 2412. Network Working Group, 1998. 55 p. https://tools.ietf.org/html/rfc2412.
30. *Krawczyk H.* SKEME: A versatile secure key exchange mechanism for Internet // Proc. Internet Society Symp. on Network and Distributed Systems Security, San Diego, CA, USA, 1996. P. 114–127.
31. *Maughhan D., Schertler M., Schneider M., and Turner J.* Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408. 1998. https://tools.ietf.org/html/rfc2408.
32. *Schneier B.* Applied Cryptography: Protocols, Algorithms and Source Code in C, 2nd ed. N.Y.: Wiley, 1996. 783 p.
33. *Kaufman C.* Internet Key Exchange (IKEv2) Protocol. RFC 4306. Network Working Group, 2005. 99 p. https://tools.ietf.org/html/rfc4306.
34. *Piper D.* The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407. Network Working Group, 1998. 32 p. https://tools.ietf.org/html/rfc2407.
35. *Aboba B., Blunk L., Carlson J., et al.* Extensible Authentication Protocol (EAP). RFC 3748. Network Working Group, 2004. 67 p. https://tools.ietf.org/html/rfc3748.
36. *Asokan N., Nierni V., and Nyberg K.* Man-in-the-Middle in Tunneled Authentication Protocols. Cryptology ePrint Archive: Report 2002/163, 2002. 15 p.
37. *Monsour B., Pereira R., Shacham A., and Thomas M.* IP Payload Compression Protocol (IPComp). RFC 3173. Network Working Group, 2001. 13 p. https://tools.ietf.org/html/rfc3173.
38. *DiBurro L., Huttunen A., Stenberg M., et al.* UDP Encapsulation of IP Security ESP Packets. RFC 3948. Network Working Group, 2005. 15 p. https://tools.ietf.org/html/rfc3948.
39. *Black D., Floyd S., and Ramakrishnan K.* The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. Network Working Group, 2001. 63 p. https://tools.ietf.org/html/rfc3168.
40. *Kent S. and Seo K.* Security Architecture for the Internet Protocol. RFC 4301. Network Working Group, 2005. 101 p. https://tools.ietf.org/html/rfc4301.
41. *Eronen P., Hoffman P., Kaufman C., and Nir Y.* Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996. Internet Engineering Task Force (IETF), 2010. 138 p. https://tools.ietf.org/html/rfc5996.
42. *Eronen P. and Hoffman P.* IKEv2 Clarifications and Implementation Guidelines. RFC 4718. Network Working Group, 2006. 58 p. https://tools.ietf.org/html/rfc4718.
43. *Berners-Lee T., Fielding R., Frystyk H., et al.* Hypertext Transfer Protocol — HTTP / 1.1. RFC 2616. Network Working Group, 1999. 176 p. https://tools.ietf.org/html/rfc2616.
44. *Eronen P., Laganier J., and Madson C.* IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5739. Internet Engineering Task Force (IETF), 2010. 32 p. https://tools.ietf.org/html/rfc5739.
45. *Atkinson R.* The IP Authentication Header. RFC 1826. Network Working Group, 1995. 13 p. https://tools.ietf.org/html/rfc1826.
46. *Metzger P. and Simpson W.* IP Authentication with Keyed MD5. RFC 1828. Network Working Group, 1995. 5 p. https://tools.ietf.org/html/rfc1828.
47. *Atkinson R. and Kent S.* IP Authentication Header. RFC 2402. Network Working Group, 1998. 22 p. https://tools.ietf.org/html/rfc2402.
48. *Kent S.* IP Authentication Header. RFC 4302. Network Working Group, 2005. 34 p. https://tools.ietf.org/html/rfc4302.

49. *Eastlake D. 3rd.* Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4305. Network Working Group, 2005. 9 p. https://tools.ietf.org/html/rfc4305.
50. *Atkinson R.* IP Encapsulating Security Payload (ESP). RFC 1827. Network Working Group, 1995. 12 p. https://tools.ietf.org/html/rfc1827.
51. *Atkinson R. and Kent S.* IP Encapsulating Security Payload (ESP). RFC 2406. Network Working Group, 1998. 22 p. https://tools.ietf.org/html/rfc2406.
52. *Kent S.* IP Encapsulating Security Payload (ESP). RFC 4303. Network Working Group, 2005. 44 p. https://tools.ietf.org/html/rfc4303.

REFERENCES

1. *Kipp E. B. H.* The SSL Protocol. Netscape Communications Corp., 1995 (Expires 10 / 95). 26 p. https://tools.ietf.org/html/draft-hickman-netscape-ssl-00.
2. *Polk T. and Turner S.* Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176. Internet Engineering Task Force (IETF), 2011. 4 p. https://tools.ietf.org/html/rfc6176.
3. *Freier A., Karlton P., and Kocher P.* The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101. Internet Engineering Task Force (IETF), 2011. 67 p. https://tools.ietf.org/html/rfc6101.
4. *Allen C. and Dierks T.* The TLS Protocol Version 1.0. RFC 2246. Network Working Group, 1999. 80 p. https://tools.ietf.org/html/rfc2246.
5. *Kaliski B.* PKCS#1: RSA Encryption Standard, version 1.5. RFC 2313. Network Working Group, 1998. 19 p. https://tools.ietf.org/html/rfc2313.
6. *Dierks T. and Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346. Network Working Group, 2006. 87 p. https://tools.ietf.org/html/rfc4346.
7. *Jonsson J. and Kaliski B.* Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447. Network Working Group, 2003. 72 p. https://tools.ietf.org/html/rfc3447.
8. *Ford W., Housley R., Polk W., and Solo D.* Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280. Network Working Group, 2002. 129 p. https://tools.ietf.org/html/rfc3280.
9. http://www.openssl.org/~bodo/tls-cbc.txt — Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures, 2004.
10. *Dierks T. and Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. Network Working Group, 2008. 104 p. https://tools.ietf.org/html/rfc5246.
11. *Eastlake D. 3rd.* Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066. Internet Engineering Task Force (IETF), 2011. 25 p. https://tools.ietf.org/html/rfc6066.
12. *Dworkin M.* Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C, 2004. 27 p.
13. *Dworkin M.* Recommendation for Block Cipher Modes of Operation: Galois / Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007. 39 p.
14. *McGrew D.* An Interface and Algorithms for Authenticated Encryption. RFC 5116. Network Working Group, 2008. 22 p. https://tools.ietf.org/html/rfc5116.
15. *Mavrogiannopoulos N.* Using OpenPGP Keys for Transport Layer Security (TLS) Authentication. RFC 5081. Network Working Group, 2007. 8 p. https://tools.ietf.org/html/rfc5081.

16. *Blake-Wilson S., Bolyard N., Gupta V., et al.* Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492. Network Working Group, 2006. 35 p. https://tools.ietf.org/html/rfc4492.
17. *Bleichenbacher D.* Chosen ciphertext attacks against protocols based on RSA Encryption Standard PKCS#1. CRYPTO'98, LNCS, 1998, vol. 1462, pp. 1–12.
18. *Klima V., Pokorny O., and Rosa T.* Attacking RSA-based Sessions in SSL/TLS. Cryptology ePrint Archive: Report 2003/052, 2003. 23 p.
19. https://csrc.nist.gov/publications/detail/fips/186/3/archive/2009-06-25 — Digital Signature Standard (DSS). NIST FIPS PUB 186-3, 2009. 131 p.
20. *Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. Internet Engineering Task Force (IETF), 2018. 160 p. https://tools.ietf.org/html/rfc8446.
21. *Eronen P. and Krawczyk H.* HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869. Internet Engineering Task Force (IETF), 2010. 14 p. https://www.rfc-editor.org/info/rfc5869.
22. https://standards.ieee.org/standard/1363-2000.html — IEEE Standard Specifications for Public Key Cryptography. IEEE Std 1363-2000, 2000. 236 p.
23. *Hamburg M., Langley A., and Turner S.* Elliptic Curves for Security. RFC 7748. Internet Research Task Force (IRTF), 2016. 22 p. https://tools.ietf.org/html/rfc7748.
24. *Grebnev S. V., Lazareva E. V., Lebedev P. A., et al.* Integratsiya otechestvennykh protokolov vyrabotki obshchego klyucha v protokol TLS 1.3 [Integration of domestic shared key generation protocols into TLS 1.3]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2018, no. 11, pp. 62–65. (in Russian)
25. *Matyukhin D. V.* O nekotorykh svoystvakh skhem vyrabotki obshchego klyucha, ispol'zuyushchikh infrastrukturu otkrytykh klyuchey, v kontekste razrabotki standartizirovannykh kriptograficheskikh resheniy [On some properties of public key generation schemes that use the public key infrastructure in the context of developing standardized cryptographic solutions]. 2011. https://www.ruscrypto.ru/resource/archive/rc2011/files/02_matyukhin.pdf. (in Russian)
26. *Nesterenko A. Y.* Ob odnom podkhode k postroyeniyu zashchishchennykh soyedineniy [About one approach to building secure connections]. Matematicheskiye Voprosy Kriptografii, 2013, no. 4:2, pp. 101–111. (in Russian)
27. *Grebnev S. V.* O vozmozhnosti standartizatsii protokolov vyrabotki obshchego klyucha [On the possibility of standardizing protocols for generating a shared key]. RusKripto, Moscow, 2014. https://www.ruscrypto.ru/resource/archive/rc2014/files/03_grebnev.pdf. (in Russian)
28. *Carrel D. and Harkins D.* The Internet Key Exchange (IKE). RFC 2409. Network Working Group, 1998. 41 p. https://tools.ietf.org/html/rfc2409.
29. *Orman H.* The Oakley Key Determination Protocol. RFC 2412. Network Working Group, 1998. 55 p. https://tools.ietf.org/html/rfc2412.
30. *Krawczyk H.* SKEME: A versatile secure key exchange mechanism for Internet. Proc. Internet Society Symp. on Network and Distributed Systems Security, San Diego, CA, USA, 1996, pp. 114–127.
31. *Maughhan D., Schertler M., Schneider M., and Turner J.* Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408. 1998. https://tools.ietf.org/html/rfc2408.
32. *Schneier B.* Applied Cryptography: Protocols, Algorithms and Source Code in C, 2nd ed. N.Y., Wiley, 1996. 783 p.

33. *Kaufman C.* Internet Key Exchange (IKEv2) Protocol. RFC 4306. Network Working Group, 2005. 99 p. https://tools.ietf.org/html/rfc4306.
34. *Piper D.* The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407. Network Working Group, 1998. 32 p. https://tools.ietf.org/html/rfc2407.
35. *Aboba B., Blunk L., Carlson J., et al.* Extensible Authentication Protocol (EAP). RFC 3748. Network Working Group, 2004. 67 p. https://tools.ietf.org/html/rfc3748.
36. *Asokan N., Nierni V., and Nyberg K.* Man-in-the-Middle in Tunneled Authentication Protocols. Cryptology ePrint Archive: Report 2002/163, 2002. 15 p.
37. *Monsour B., Pereira R., Shacham A., and Thomas M.* IP Payload Compression Protocol (IPComp). RFC 3173. Network Working Group, 2001. 13 p. https://tools.ietf.org/html/rfc3173.
38. *DiBurro L., Huttunen A., Stenberg M., et al.* UDP Encapsulation of IP Security ESP Packets. RFC 3948. Network Working Group, 2005. 15 p. https://tools.ietf.org/html/rfc3948.
39. *Black D., Floyd S., and Ramakrishnan K.* The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. Network Working Group, 2001. 63 p. https://tools.ietf.org/html/rfc3168.
40. *Kent S. and Seo K.* Security Architecture for the Internet Protocol. RFC 4301. Network Working Group, 2005. 101 p. https://tools.ietf.org/html/rfc4301.
41. *Eronen P., Hoffman P., Kaufman C., and Nir Y.* Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996. Internet Engineering Task Force (IETF), 2010. 138 p. https://tools.ietf.org/html/rfc5996.
42. *Eronen P. and Hoffman P.* IKEv2 Clarifications and Implementation Guidelines. RFC 4718. Network Working Group, 2006. 58 p. https://tools.ietf.org/html/rfc4718.
43. *Berners-Lee T., Fielding R., Frystyk H., et al.* Hypertext Transfer Protocol — HTTP / 1.1. RFC 2616. Network Working Group, 1999. 176 p. https://tools.ietf.org/html/rfc2616.
44. *Eronen P., Laganier J., and Madson C.* IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5739. Internet Engineering Task Force (IETF), 2010. 32 p. https://tools.ietf.org/html/rfc5739.
45. *Atkinson R.* The IP Authentication Header. RFC 1826. Network Working Group, 1995. 13 p. https://tools.ietf.org/html/rfc1826.
46. *Metzger P. and Simpson W.* IP Authentication with Keyed MD5. RFC 1828. Network Working Group, 1995. 5 p. https://tools.ietf.org/html/rfc1828.
47. *Atkinson R. and Kent S.* IP Authentication Header. RFC 2402. Network Working Group, 1998. 22 p. https://tools.ietf.org/html/rfc2402.
48. *Kent S.* IP Authentication Header. RFC 4302. Network Working Group, 2005. 34 p. https://tools.ietf.org/html/rfc4302.
49. *Eastlake D. 3rd.* Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4305. Network Working Group, 2005. 9 p. https://tools.ietf.org/html/rfc4305.
50. *Atkinson R.* IP Encapsulating Security Payload (ESP). RFC 1827. Network Working Group, 1995. 12 p. https://tools.ietf.org/html/rfc1827.
51. *Atkinson R. and Kent S.* IP Encapsulating Security Payload (ESP). RFC 2406. Network Working Group, 1998. 22 p. https://tools.ietf.org/html/rfc2406.
52. *Kent S.* IP Encapsulating Security Payload (ESP). RFC 4303. Network Working Group, 2005. 44 p. https://tools.ietf.org/html/rfc4303.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.021

СХЕМА ГРУППОВОЙ АУТЕНТИФИКАЦИИ НА ОСНОВЕ ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

Е. А. Шляхтина, Д. Ю. Гамаюнов

Московский государственный университет имени М. В. Ломоносова, г. Москва, Россия

Рассматривается проблема взаимной аутентификации пользователей в децентрализованных системах обмена сообщениями в отсутствие доверенной третьей стороны. Предложен алгоритм взаимной аутентификации пользователей групп на основе доказательства с нулевым разглашением. Алгоритм позволяет аутентифицировать пользователей децентрализованной сети без установки общего секрета по стороннему каналу, опираясь на существующие в сети цепочки доверия. В основе метода лежит протокол демократичной групповой подписи DGS и алгоритм выработки общего ключа для больших и динамических групп CCEGK. Проведены анализ безопасности, устойчивости схемы аутентификации, в том числе к атаке Сивиллы, и оценка сложности предлагаемого алгоритма. Алгоритм реализован на модельном децентрализованном приложении на основе P2P топологии Chord, с помощью модельной реализации сделаны оценки накладных расходов схемы аутентификации и времени сходимости для некоторых частных случаев конфигураций групп пользователей и начальных цепочек доверия.

Ключевые слова: *аутентификация, доказательство с нулевым разглашением, децентрализованные коммуникации.*

DOI 10.17223/20710410/51/3

GROUP AUTHENTICATION SCHEME BASED ON ZERO-KNOWLEDGE PROOF

E. A. Shliakhtina, D. Y. Gamayunov

Lomonosov Moscow State University, Moscow, Russia

E-mail: ea.shlyakhtina@gmail.com, gamajun@seclab.cs.msu.su

In this paper, we address the problem of mutual authentication in user groups in decentralized messaging systems without trusted third party. We propose a mutual authentication algorithm for groups using zero-knowledge proof. Using the algorithm, which is based on trust chains existing in decentralized network, users are able to authenticate each other without establishing a shared secret over side channel. The proposed algorithm is based on Democratic Group Signature protocol (DGS) and Communication-Computation Efficient Group Key algorithm for large and dynamic groups (CCEGK). We have performed security analysis of the proposed mutual authentication scheme against several attacks including Sybil attack and have made

complexity estimation for the algorithm. The algorithm is implemented in an experimental P2P group messaging application, and using this implementation we estimate overhead of the authentication scheme and convergence time for several initial configurations of user groups and trust chains.

Keywords: *authentication, zero-knowledge proof, decentralized communications.*

Введение

В настоящее время значительно выросла роль систем мгновенного обмена сообщениями и требования к обеспечению конфиденциальности таких систем коммуникаций. При этом все коммерческие системы, доступные в Интернете, представляют собой централизованнные системы с доверенной третьей стороной, в роли которой выступает поставщик сервиса. В то же время существует значительный исследовательский интерес к одноранговым сетям для реализации систем обмена сообщениями, в которых обеспечивается равноправие всех участников сети и не требуется доверенная третья сторона. В таких системах участник сети одновременно является клиентом и выполняет функции сервера для формирования распределённой децентрализованной топологии и обеспечения возможности пересылки сообщений других участников сети. Среди достоинств таких сетей — гибкость при изменении структуры сети, надёжность и отказоустойчивость. Принципы децентрализации хорошо подходят для создания мессенджеров — приложений для обмена мгновенными сообщениями.

В работе [1] предложена схема многопользовательских защищённых коммуникаций на основе протокола multi-party Off-The-Record (mpOTR) [2] со свойствами секретности будущих сообщений, отказуемости и согласованности текста переписки. Протокол основан на криптосистеме с открытым ключом. Данная схема была использована при разработке криптографического группового чата [3, 4]. При реализации чата был также использован алгоритм аутентификации долговременных открытых ключей пользователей [5] на основе протокола Социалистов-Миллионеров [6], являющегося протоколом с нулевым разглашением. Аутентификация открытых ключей служит защитой от атаки «человек в середине». Протокол Социалистов-Миллионеров позволяет сравнивать общий секрет двух пользователей, содержащий отпечатки[1] их открытых ключей и секретную фразу, о которой участники договариваются по стороннему каналу связи. Однако не всегда есть возможность безопасной установки общего секрета или задания вопроса, на который собеседник, и только он, точно знает ответ.

В данной работе предлагается метод аутентификации открытых ключей пользователей без использования общего секрета, установленного по стороннему каналу связи, обрабатывающий данные об уже существующих цепочках доверия в сети. Метод применим в системе со свойствами отказуемости. В его основе лежит механизм групповой подписи с нулевым разглашением [7], позволяющий членам аутентифицированных групп подписывать сообщения от их имени и обеспечивающий невозможность подделки групповой подписи участниками, не являющимися членами группы. Если пользователь получит информацию об открытом ключе другого пользователя, подтверждённую цепочкой доверенных участников, связывающих их, то он может доверять этой информации.

[1]Применяя криптографическую хеш-функцию к открытому ключу, можно получить отпечаток ключа, идентифицирующий его.

1. Модель

Сетевая модель: децентрализованная топология сети, каждый участник сети обладает уникальным идентификатором id, парой открытый/секретный ключ (pk, sk).

Модель нарушителя: участник, имеющий возможность перехватывать трафик и контролировать узлы сети. Цель нарушителя — выдать себя за другого человека, подменив соответствующие открытые ключи.

Модельное приложение: децентрализованное приложение для защищённого обмена сообщениями в групповых конференциях, реализованное с помощью браузерной технологии WebRTC [8] (Web Real Time Communication), позволяющей пользователям установить P2P (Peer-to-Peer)-соединение. Пиринговая сеть основана на распределённой хэш-таблице Chord [9], размещённой в узлах сети. На основе этого протокола строится одноранговая сеть с топологией «кольцо», которая используется в качестве P2P-транспорта для чата. Chord служит для организации эффективного доступа к узлам распределённой системы и хранения информации. Безопасность групповых коммуникаций обеспечивается протоколом mpOTR [2], в основе которого лежит криптосистема с открытым ключом. В реализации используется система защищённых групповых коммуникаций, предложенная в [1] и обладающая свойствами конфиденциальности, секретности будущих сообщений, аутентичности, отказуемости, целостности передачи данных и согласованности текста переписки.

2. Краткое описание используемых протоколов

Механизм групповой подписи позволяет членам группы анонимно подписываться от имени группы так, что любой проверяющий может убедиться в том, что документ был подписан участником группы. Особенностью групповой подписи является наличие менеджера группы, который может раскрывать личность выпустившего ту или иную подпись. В обязанности менеджера входит контроль всех динамических изменений в группе (например, добавление или исключение участников), а потому он должен поддерживать группу на протяжении всего её существования. Кроме того, остальные участники должны ему доверять. Очевидно, наличие такого доверенного участника является узким местом в схемах групповых подписей. Для повышения надёжности в некоторых из них существуют модификации, позволяющие разделить обязанности по добавлению новых участников и раскрытию личности подписавшегося между двумя менеджерами, однако уровень доверия к ним остаётся по-прежнему высоким.

В 2006 г. опубликована работа [7] Марка Манулиса, в которой представлена схема демократичной групповой подписи DGS, где роль доверенного менеджера группы отсутствует. DGS позволяет инициализировать группу, принимать решение о принятии нового члена в группу и об исключении старых коллективно всеми участниками, а возможность выявить личность подписавшего появляется у всех членов группы. Вариант групповой подписи без такого элемента централизации, как доверенный менеджер, является наиболее естественным для реализации в условиях одноранговой сети.

Предполагается выполнение следующих условий:

1) члены группы не разглашают данные, которые могут быть использованы посторонними для раскрытия издателя подписи;
2) в группе есть хотя бы один честный пользователь;
3) пользователи аутентифицируют свои сообщения в процессе коммуникации.

Рассматривается пассивный противник, не являющийся членом группы, так как в результате работы протокола все участники будут обладать групповым секретом.

В основе протокола лежат следующие блоки:

— протокол выработки общего секрета группы CGKA (Contributory Group Key Agreement protocol);
— подпись знания SK (Signature of Knowledge).

Прежде чем перейти к схеме групповой подписи, приведём описание этих блоков.

2.1. Протокол выработки общего секрета

$$\mathbf{CGKA} = \{\mathbf{Setup}, (\mathbf{Join}_i, \mathbf{Join}_u), \mathbf{Leave}\}$$

Setup — интерактивный алгоритм, исполняющийся каждым из n пользователей, желающих выработать общий секрет; инициализирует группу.

Вход: индивидуальный секрет[2] i-го пользователя k_i, соответствующий вклад z_i.

Выход: групповой секрет k_G.

В результате взаимодействия участников у каждого формируется множество вкладов остальных пользователей $Z = \{z_j : j \in \{1, \ldots, n\}, j \neq i\}$.

$(\mathbf{Join}_i, \mathbf{Join}_u)$ — пара интерактивных алгоритмов, выполняющихся группой и новым членом группы соответственно для добавления нового участника в группу.

Вход $\mathbf{Join}_i$: индивидуальный секрет i-го пользователя k_i, вклад нового пользователя z_u, структура группы.

Вход $\mathbf{Join}_u$: индивидуальный секрет нового пользователя k_u, множество вкладов участников Z, структура группы.

Выход $(\mathbf{Join}_i, \mathbf{Join}_u)$: обновлённое множество вкладов пользователей Z, новый групповой секрет k_G.

Leave — интерактивный алгоритм, выполняющийся между $(n-1)$ остающимися участниками для исключения j-го участника из группы. В результате изменяются индивидуальные секреты некоторых пользователей, обновляется список вкладов пользователей, структура группы и групповой секрет k_G.

Вход: индивидуальный секрет участника k_i, вклад покидающего группу z_j, структура группы.

Выход: обновлённые список вкладов пользователей Z, структура группы и групповой секрет k_G.

В работе [7] не фиксируется определённый протокол выработки общего секрета, но требуется наличие следующих ключевых свойств у протокола:

— для пассивного противника вычислительно сложно найти групповой секрет;
— для пассивного противника вычислительно сложно отличить случайные биты от битов группового секрета;
— секретность будущих сообщений (forward secrecy) — обладая некоторым множеством старых групповых секретов, пассивный противник не сможет вычислить новые;
— секретность прошлых сообщений (backward secrecy) — обладая некоторым множеством текущих групповых секретов, пассивный противник не сможет вычислить старые;
— независимость групповых секретов.

[2]Каждый пользователь в группе имеет индивидуальный секрет, на основе совокупности индивидуальных секретов участников группы вырабатывается общий секрет.

Кроме того, определён вид вкладов участников: $z_i = g^{k_i}$. Здесь g — образующий элемент мультипликативной группы $\mathbb{Z}_p^*$, где p — большое простое число.

В качестве протокола выработки общего секрета в [7] рекомендуется протокол TGDH (Tree-based Group Diffe — Hellman), представляющий собой процедуру генерации общего секрета на основе протокола Диффи — Хеллмана и древовидной структуры группы. Древовидная структура позволяет эффективно пересчитывать общий секрет при динамических событиях, что важно в случае больших групп. Протоколами, основанными на этом принципе и удовлетворяющими требованиям, являются EGK [10], STR [11, 12], CCEGK [13]. Последний является улучшенной версией TGDH с точки зрения вычислительной и коммуникационной сложности, что показано в работе [13]. Поэтому именно этот протокол выбран для построения схемы аутентификации.

2.2. Подпись знания

Подпись знания [14] представляет собой неинтерактивное доказательство знания некоторого секрета с нулевым разглашением, зависящее от сообщения. Неинтерактивным оно становится благодаря эвристическому методу Фиата — Шамира [15].

$$\mathbf{SK} = \{\mathbf{SKSig}, \mathbf{SKVer}\}$$

SKSig — вероятностный алгоритм генерации подписи; **SKVer** — детерминированный алгоритм ее проверки.

Определение 1. Предположение DL (Discrete Logarithm assumption) о сложности дискретного логарифмирования. Пусть G — циклическая группа с образующим элементом g. Не существует вероятностного полиномиального алгоритма A, который с непренебрежимо малой вероятностью вычисляет $\log_g(g^a)$, получая на вход (G, g, g^a), где $a \in \mathbb{Z}_{\mathrm{ord}(G)}$.

Предположение DL выполняется в G, если $Pr[A(G, g, g^a) = a] \leqslant negl(n)$, где $n = \|\mathrm{ord}(G)\|$. Здесь $\|x\|$ — длина записи двоичного представления числа x; $negl$ — пренебрежимо малая функция.

Определение 2. Пусть G — циклическая группа и выполняется предположение DL. Тогда

$$(c, s) \in \{0,1\}^k \times \mathbb{Z}_{\mathrm{ord}(G)}, \quad (c, s) = SK[(\alpha) : y = g^\alpha](m)$$

есть подпись знания дискретного логарифма элемента $y \in G$ по основанию g для сообщения m, если $c = H(m||y||g||g^s y^c)$, где $H : \{0,1\}^* \to \{0,1\}^k$ — криптографическая хэш-функция.

Алгоритм её вычисления при знании $x = \log_g(y)$:

1) $r \in_R \mathbb{Z}^*_{\mathrm{ord}(G)}$;
2) $c = H(m||y||g||g^r)$;
3) $s = r - cx$.

Этот подход позволяет конструировать подписи знания более сложных отношений. В [7] для генерации групповой подписи используются такие подписи знания:

1) $SK[(\alpha_i, \beta) : y = g_1^\beta g_2^{\alpha_i} \wedge (z_1 = g_2^{\alpha_1} \vee \ldots \vee z_n = g_2^{\alpha_n})](m)$ — подпись знания дискретного логарифма по основанию g_2 одного значения множества $\{z_1, \ldots, z_n\}$ и равенства этой величины экспоненте g_2-составляющей y;
2) $SK[(\alpha, \beta) : y_1 = g_2^\beta \wedge y_2 = g_1^\beta g_2^\alpha](m)$ — подпись знания представления y_2 по основаниям g_1 и g_2 и равенства дискретного логарифма y_1 по основанию g_2 и экспоненты g_1-составляющей величины y_2.

2.3. Схема групповой подписи

$$\mathbf{DGS} = \{\mathbf{Setup}, \mathbf{Join}, \mathbf{Leave}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{Trace}, \mathbf{TraceVerify}\}$$

Введём следующие обозначения:

- t — счётчик, инициализированный нулём; нужен для отслеживания конфигурации группы и увеличивается при любом динамическом изменении. Все параметры групповой подписи связаны со счётчиком;
- $Y_{[t]}$ — открытый ключ группы, $Y_{[t]} = (g^{\hat{x}_{[t]}}, Z_{[t]}) = (\hat{y}_{[t]}, Z_{[t]})$, где $Z_{[t]} = \{z_{1[t]}, \ldots, z_{n[t]}\}$ — множество вкладов всех участников;
- $x_{i[t]}$ — индивидуальный секрет i-го члена группы;
- $\hat{x}_{[t]}$ — общий секрет группы.

Алгоритм **Setup** выполняется всеми участниками для формирования группы. Счётчик t инициализируется нулём. Пользователи устанавливают индивидуальные секреты $x_{i[0]} \in_R \mathbb{Z}_{p-1}$ и вычисляют свои вклады $z_{i[0]}$. Далее выполняется алгоритм **CGKA**.**Setup**$(x_{i[0]}, z_{i[0]})$ для формирования структуры группы и выработки группового секрета $\hat{x}_{[0]} \in \mathbb{Z}_{p-1}$. Из полученной информации создаётся открытый ключ группы $Y_{[t]}$.

Алгоритм **Join** позволяет новому участнику U присоединиться к группе. U устанавливает свой секрет $x_{u[t+1]} \in_R \mathbb{Z}_{p-1}$ и вычисляет соответствующий вклад $z_{u[t+1]}$, где t — текущий счётчик конфигурации группы. Затем новый участник и группа выполняют алгоритмы протокола выработки группового секрета **CGKA**.**Join**$_u(x_{u[t+1]})$, **CGKA**.**Join**$_i(z_{u[t+1]})$ соответственно. В результате обновляется структура группы и формируется новый групповой секрет.

Ситуация выхода участника из группы обрабатывается алгоритмом **Leave** с помощью алгоритма **CGKA**.**Leave**. Все оставшиеся участники обновляют структуру группы. Один из участников группы должен изменить свой секрет, пересчитать групповой секрет и предоставить данные остальным членам группы для вычисления группового секрета и обновления открытого ключа группы.

Алгоритм **Sign** генерирует групповую подпись $\sigma = (\widetilde{g}, \widetilde{y}, S)$ сообщения $m \in \{0,1\}^*$, где

$$S = SK[(\alpha_i, \beta) : \widetilde{g} = g^{\beta} \wedge \widetilde{y} = \hat{y}_{[t]}^{\beta} g^{\alpha_i} \wedge (z_{1[t]} = g^{\alpha_1} \vee \ldots \vee z_{n[t]} = g^{\alpha_n})](m).$$

Данная подпись является комбинацией подписей знания, описанных в п. 2.2. Тем самым предъявляющий подпись доказывает, что пара $(\widetilde{g}, \widetilde{y})$ шифрует вклад $z_{i[t]}$ в множестве вкладов $Z_{[t]}$, не раскрывая его, а знание дискретного логарифма, используемого для вычисления $z_{i[t]}$, показывает, что подписавшийся действительно владелец вклада $z_{i[t]}$.

Алгоритм **Verify** реализует проверку подписи σ. Для его работы необходим открытый ключ группы $Y_{[t]}$.

Для раскрытия личности подписавшего используется алгоритм **Trace**. На первом этапе осуществляется проверка подписи сообщения с помощью алгоритма **Verify**. Далее с помощью группового секрета $\hat{x}_{[t]}$ выделяется вклад подписавшего сообщение пользователя $z_{i[t]}$, который идентифицирует внутри группы личность подписавшегося. Также на этом шаге формируется доказательство того, что идентификация личности была проведена корректно, т. е. с использованием группового секрета.

Алгоритм **VerifyTrace** реализует проверку корректности результатов работы функции **Trace**; эта процедура может быть проведена не членом группы.

В схеме DGS размер групповой подписи и открытого ключа группы линейно зависят от количества членов группы. В настоящее время существуют групповые подписи (например, [14, 16]), где оба эти параметра являются константами, однако это достигается при помощи доверенного менеджера, который обрабатывает все изменения в составе группы и обновляет базу данных, содержащую информацию об участниках, размер которой также линейно зависит от числа участников.

Безопасность DGS основана на предположениях о дискретном логарифмировании DL и о сложности задачи отличия DDH (Decisional Diffie — Hellman assumption). Основными свойствами протокола DGS являются корректность, отслеживаемость, анонимность, несвязываемость, невозможность фальсификации.

3. Описание схемы аутентификации

Приведём описание схемы взаимной аутентификации пользователей. Предполагается существование в сети групп аутентифицированных между собой пользователей, т. е. каждый член группы имеет достоверную информацию об открытых ключах пользователей в этой группе. Протокол, с помощью которого происходила эта аутентификация, не фиксируется. Это может быть метод [5], именно такой вариант использован для реализации в данной работе. Аутентифицированная группа использует протокол групповой подписи DGS для выработки открытого ключа группы. Таким образом, сеть разбивается на множество групп, члены которых могут доказать свою принадлежность к группе третьим лицам, подписывая сообщения от имени группы.

После формирования группы аутентифицированных пользователей один из участников публикует информацию о группе:

$$groupID : (Y, idList, Sig_G(groupID, Y, idList)),$$

где $groupID$ — идентификатор группы; Y — открытый ключ, $idList = \{id_1, \ldots, id_n\}$ — список идентификаторов членов группы; Sig_G — групповая подпись.

В данной реализации этот участник определяется своим положением в текущей структуре группы. В протоколе CCEGK им является участник, соответствующий самому правому листу в дереве группы.

Наличие цепочки из пересекающихся между собой групп, объединяющей пользователей $U_1, \ldots, U_n$, позволяет им использовать предлагаемый метод аутентификации.

3.1. Поиск цепочек групп

Сначала каждый из участников получает из сети данные о существующих аутентифицированных группах. Затем, используя эту информацию, участники ищут пути друг до друга, которые в дальнейшем используются для аутентификации.

Начнём с описания поиска цепочек групп, связывающих двух пользователей. Цепочки либо состоят из одного звена, либо их длина не больше некоторого заранее установленного числа. Поиск цепочек выполняется алгоритмами 1 и 2.

Получая на вход список групп в сети $groups$ и идентификаторы пользователей id_A, id_B, алгоритм 1 ($findChains$) находит цепочки групп, ведущие от A к B: $groupsA$ — список групп, в которых состоит A; $groupsB$ — список групп, в которых состоит B. Если данные списки не пусты, находится список $sharedGroups$, являющийся пересечением списков $groupsA$ и $groupsB$. Если он не пуст, то эти группы являются маршрутом от A к B и вносятся в список $output$ найденных цепочек. Если список $sharedGroups$ пуст, то поочередно перебираются все группы $group$ из списка $groupsA$ и для каждой из них выполняется процедура $findPath$ поиска маршрута от группы $group$ к списку групп $groupsB$.

Алгоритм 1. $findChains$: поиск цепочек групп от A к B

Вход: $groups$ — список групп; id_A; id_B.
Выход: $output$ — список, содержащий все найденные цепочки.

1: $groupsA := \{group \in groups : id_A \in group\}$;
2: $groupsB := \{group \in groups : id_B \in group\}$;
3: $viewedGroups := [\,]$;
4: $output := [\,]$;
5: **Если** $groupsA \neq \varnothing \ \wedge \ groupsB \neq \varnothing$, **то**
6: $sharedGroups := groupsA \cap groupsB$.
7: **Если** $sharedGroups \neq \varnothing$, **то**
8: $output := output \cup sharedGroups$,
9: **иначе**
10: **Для всех** $group \in groupsA$:
11: $currentPath := []$;
12: $findPath(viewedGroups, group, groupsB, currentPath, groups, output)$.
13: **Вернуть** $output$.

Алгоритм 2 ($findPath$) — рекурсивная процедура поиска маршрута от заданной группы $group$ к списку групп $groupsB$. Входные параметры:

— $viewedGroups$ — список уже рассмотренных групп (не рассматриваются в последующих шагах);
— $group$ — исходная группа маршрута;
— $groupsB$ — конечный список групп маршрута;
— $currentPath$ — список групп, а именно маршрут, пройденный от пользователя A на текущий момент времени;
— $groups$ — опубликованные данные обо всех существующих в сети группах;
— $output$ — список найденных маршрутов.

Вычисляем список $neighbors$ — это окружение группы $group$, т. е. те группы, с которыми она пересекается, за исключением уже рассмотренных $viewedGroups$. Обновляем список рассмотренных групп $newViewedGroups$ и пройденный маршрут $path$ от A на текущий момент времени. Перебираем все группы из списка $neighbors$: если группа содержит пользователя B, то она является завершением маршрута от A к B и данный маршрут вносится в список $output$; в противном случае если число звеньев в текущей цепочке не превосходит $maxNum$, то выполняем процедуру $findPath$ поиска маршрута к $groupsB$.

Расширим алгоритм поиска путей между двумя участниками на группу из n участников. Каждый пользователь сортирует список из идентификаторов членов группы $idList$ и циклически сдвигает его так, что его собственный идентификатор оказывается последним в списке.

Определим $subgroups$ как список множеств пользователей, где в одно множество входят пользователи, если алгоритмом 1 найдены цепочки, объединяющие их. Сначала в соответствии с расположением пользователей в списке для U_i $subgroups$ имеет вид

$$subgroups = (\{U_{i+1}\}, \ldots, \{U_n\}, \{U_1\}, \ldots, \{U_{i-1}\}, \{U_i\}).$$

Затем с помощью алгоритма 3 ($search$) каждый участник находит первого пользователя в списке, до которого есть путь из пересекающихся групп, либо определяет

Алгоритм 2. *findPath*: поиск маршрута от группы *group* к списку групп *groupsB*

Вход: *viewedGroups*, *group*, *groupsB*, *currentPath*, *groups*, *output*.

1: $path := currentPath \cup \{group\}$;
2: $neighbors := \{g \in groups \mid g \cap group \neq \varnothing \wedge g \neq group \wedge g \notin viewedGroups\}$;
3: $newViewedGroups := viewedGroups \cup neighbors$.
4: **Для всех** $g \in neighbors$:
5: **Если** $g \in groupsB$, **то**
6: $output := output \cup \{path \cup \{g\}\}$,
7: **иначе**
8: **Если** $|path| < maxNum$, **то**
9: $findPath(newViewedGroups, g, groupsB, path, groups, output)$.

отсутствие связи с остальными участниками. Для хранения идентификаторов пользователей, до которых участнику не удалось найти путь, существует список недостижимых пользователей *unreachableList*. Это позволяет избегать в будущем проверки существования связи с этими пользователями.

Алгоритм 3. *search*: поиск доступного пользователя для U_i

1: **Для всех** $subgroup \in subgroups$:
2: **Если** $U_i \notin subgroup$, **то**
3: **Для всех** $user \in subgroup$:
4: **Если** $user \notin unreachableList$, **то**
5: $chains := findChains(U_i, user, groups)$.
6: **Если** $chains = \varnothing$, **то**
7: $unreachableList := unreachableList \cup \{user\}$,
8: **иначе**
9: **Вернуть** $(user, chains)$.

Далее $U_1, \ldots, U_n$ обмениваются сообщениями с результатами. Получив сообщение, пользователь U_i осуществляет проверку корректности полученной цепочки:

1) все группы из цепочки должны существовать в списке *groups*;
2) если U_i состоит в данной группе, то происходит проверка данных группы, указанных в *groups*;
3) если U_i не состоит в данной группе, то его индентификатор не должен находиться в списке *idList* группы;
4) полученный список должен быть цепочкой пересекающихся групп.

Если проверка пройдена успешно, то подгруппы в *subgroups*, содержащие пользователей, связанных цепочками, объединяются в одну. Так, если между всеми пользователями $U_1, \ldots, U_n$ существует связь через группы, найденная за установленное количество шагов, то понадобится один запуск описанной процедуры и *subgroups* будет состоять из одной подгруппы, куда входят все пользователи, так как каждый найдёт путь до первого пользователя из своего списка.

Иначе список *subgroups* сортируется и циклически сдвигается так, что множество, содержащее U_i, становится последним в списке. Процесс повторяется, пока число подгрупп в *subgroups* не перестанет изменяться — это даст участникам полную информа-

цию о найденных связях между ними, однако метод аутентификации для такой группы неприменим.

3.2. Аутентификация пользователя

Пусть A и B связаны цепочкой групп $chain = \{G_1, \ldots, G_n\}$. Чтобы подтвердить пользователю B данные о своём открытом ключе, A формирует сообщение

$$m = (id_A, pk_A, target, chain),$$

где id_A — идентификатор A; pk_A — отпечаток его открытого ключа; $target = id_B$ — идентификатор получателя сообщения. Затем A отправляет в G_1 сообщение с групповой подписью $(m, Sig_{G_1}(m))$.

Получив такое сообщение, члены группы G_i действуют по следующему алгоритму:

1) Если $i = 1$:
   те члены группы, которые состоят в G_2, проверяют подпись группы G_1 и отпечаток открытого ключа пользователя с id_A. Если все проверки прошли успешно, то каждый из них отправляет в G_2 сообщение с цепочкой подписей $(m, Sig_{G_2}(Sig_{G_1}(m)))$.
2) Если $i \in \{2, \ldots, n-1\}$:
   те члены группы, которые состоят в G_{i+1}, проверяют подписи групп G_i, G_{i-1}, $\ldots, G_1$ и дожидаются получения сообщений от всех пользователей, принадлежащих одновременно G_{i-1} и G_i. Если все сообщения получены и информация в них совпадает, то каждый из них отправляет эти данные в G_{i+1}, предварительно дополнив цепочку подписей.
3) Если $i = n$:
   получатель, указанный в сообщении, проверяет все подписи групп $G_1, \ldots, G_n$. Так же, как и в предыдущем случае, дожидается получения сообщений от всех пользователей, принадлежащих одновременно G_{n-1} и G_n. Если все сообщения получены, информация в них совпадает и отпечаток открытого ключа отправителя сообщения равен тому, что был получен ранее, то пользователь принимает доказательство.

Надёжность такой схемы можно повысить, рассылая сообщения по множеству найденных цепочек, если таковые имеются. Тогда, приняв сообщение от A, B также находит цепочки до A. В этом случае B примет доказательство от A, если по всем установленным цепочкам групп пришло одинаковое сообщение.

3.3. Групповой вариант аутентификации

Чтобы использовать предложенный метод для аутентификации группы пользователей, воспользуемся методом Круговой аутентификации [5]. Приведём его краткое описание.

Создавая ориентированное кольцо, пользователи проходят процесс парной аутентификации. Каждый участник аутентифицирует своего соседа по часовой стрелке; если он не прошел проверку, то выбирается следующий за ним участник. В случае успешной аутентификации пара формирует группу доверенных пользователей. Процесс повторяется до первого успешного результата. При этом составляются списки успешно и неуспешно прошедших проверку. Под статусом участника понимается степень доверия к нему. Статус может принимать значения GOOD, BAD, UNKNOWN. Далее пользователи обмениваются полученными результатами. Алгоритм обработки сообщений:

— если статус отправителя UNKNOWN: сообщение хранится до момента уточнения статуса;

— если статус отправителя GOOD: пользователи со статусом UNKNOWN, состоящие в полученном списке доверенных или нарушителей, приобретают статусы GOOD и BAD соответственно и добавляются в соответствующие списки получателя;
— если статус отправителя BAD: пользователи со статусом UNKNOWN, состоящие в полученном списке доверенных участников, получают статус BAD после того, как подтвердится, что все они принадлежат одной группе доверенных пользователей.

В результате каждый из участников будет обладать информацией об аутентичности остальных членов группы. Безопасность метода обоснована стойкостью схемы цифровой подписи (в данном случае используется RSA, стойкость основана на вычислительной сложности задачи факторизации) и метода парной аутентификации. Применяя метод цепочек групп в качестве парной процедуры аутентификации, получаем групповой вариант предложенного метода аутентификации.

3.4. Исследование безопасности

Рассмотрим парную аутентификацию пользователей A и B через цепочку групп $G_1, \ldots, G_n$, где за пересылку сообщений отвечают пересечения $G_1 \cap G_2$, …, $G_i \cap G_{i+1}$, …, $G_{n-1} \cap G_n$. При этом B примет доказательство A, если до него дойдут сообщения с данными пользователя A, подписанные участниками всех этих групп, и проверки всех групповых подписей пройдут успешно.

Аутентификация методом цепочки групп предполагает пересылку сообщений с доказательствами из группы в группу. Под группой понимается множество аутентифицированных между собой пользователей, находящихся на этапе коммуникации протокола mpOTR. В этом случае внешний нарушитель, не обладающий групповым секретом и индивидуальным секретом проверенного собеседника, не сможет подменить передаваемые данные.

Перейдём к случаю, когда нарушитель — член некоторых групп из цепочки, состоящий в их пересечении. Рассмотрим несколько ситуаций, когда под контролем злоумышленника находится множество пользователей в пересечении групп:

1) Злоумышленник контролирует все узлы, находящиеся в пересечении G_i и G_{i+1}. Тогда ему необходимо состоять во всех группах цепочки, чтобы уметь переподписывать подменённое сообщение от A к B и от B к A.
2) Злоумышленник контролирует все узлы, находящиеся в пересечении G_1 и G_2. Тогда данные пользователя A могут быть подменены. Аналогично: нужно контролировать все узлы пересечения групп G_{n-1} и G_n для изменения отпечатка открытого ключа B.
3) Подконтрольные злоумышленнику узлы существуют в каждом пересечении цепочки $G_1, \ldots, G_n$, но полностью не контролируется ни одно из них. Тогда неискажённые данные достигнут честных пользователей цепочки и, получив противоречивые данные, пользователи этого пересечения не станут отправлять их дальше, будет превышено время ожидания и процесс аутентификации завершится неуспешно.

Если сообщения посылаются по множеству найденных цепочек и принимаются только в случае совпадения всех присланных данных, то злоумышленнику для подмены отпечатков открытых ключей нужно контролировать процесс передачи доказательств по всем этим цепочкам.

Однако если злоумышленник состоит в пересечении групп некоторой цепочки, он может помешать успешной аутентификации двух пользователей, если удержит валид-

ное сообщение с доказательством. В этом случае будет превышено время ожидания и процесс завершится неуспехом.

В случае атаки Сивиллы, в ходе которой жертва — пользователь A — окружён только узлами, контролируемыми злоумышленником, как описано в п. 2, B получит подменённый отпечаток ключа A. Для того чтобы убедить A принять искажённое доказательство от B, злоумышленнику необходимо состоять во всех группах всех найденных цепочек либо полностью окружать пользователя B.

Пусть злоумышленник имеет подконтрольные ему узлы в каждой аутентифицированной группе сети. Для контроля передачи данных по цепочке он должен полностью контролировать некоторое пересечение групп в ней. Поэтому рассмотрим случай, когда были созданы аутентифицированные группы из честного пользователя сети и контролируемого узла. В этом случае можно полагать, что для любых двух пользователей существует цепочка G_1, G_2, G_3, объединяющая их (рис. 1).

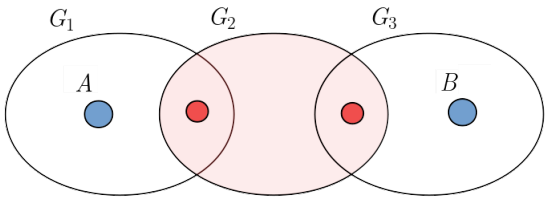


Рис. 1. Цепочка групп G_1, G_2, G_3, где красным обозначены подконтрольные злоумышленнику объекты

Этот случай подходит под описание п. 2. Если цепочка такого типа является единственной, объединяющей A и B, то атака сработает и открытые ключи будут подменены. В противном случае существует цепочка, сформированная до начала атаки, где пересечение не контролируется полностью, а значит данные, полученные по ней, не совпадут с данными первой цепочки.

Таким образом, необходимым условием успешной работы атаки является контроль некоторого пересечения групп в цепочке, причём нельзя получить полный контроль над пересечением групп, созданных до атаки, это возможно лишь при создании новых групп. Поэтому нельзя гарантировать успешную работу атаки даже в случае, когда злоумышленнику удалось войти в доверие к каждому пользователю сети.

3.5. Оценка сложности

Пусть n — число участников в группе. В табл. 1 приведены оценки сложности используемых процедур.

Таблица 1

Алгоритм	**Число раундов**	**Количество возведений в степень**	**Число сообщений**
CCEGK.Setup	$\log n$	$2\log n - 2$	$2n - 1$
CCEGK.Join	1	1	2
CCEGK.Leave	1	$3\log n - 3$	1
DGS.Sign	–	$7n - 2$	–
DGS.Verify	–	$7n$	–

Каждый пользователь в пересечении групп некоторой цепочки, число звеньев которой ограничено числом $maxNum$, выполняет $maxNum$ запусков процедуры DGS.Verify и два запуска процедуры DGS.Sign. Таким образом, число операций возведения в степень при условии, что N — среднее число участников в группах цепочки, равно

$$maxNum \cdot 7N + 2(7N - 2).$$

Участники, запустившие процесс аутентификации, находятся в концах цепочки и выполняют

$$maxNum \cdot 7N + (7N - 2)$$

операций возведения в степень.

4. Реализация алгоритма аутентификации

Алгоритм аутентификации методом цепочек групп реализован [17] и протестирован авторами данной работы. Реализация выполнена на языке JavaScript в существующем прототипе децентрализованного криптографического чата, построенного на основе P2P-топологии Chord поверх WebRTC.

Параметр $maxNum$, ограничивающий число звеньев в цепочке при её поиске, в данной реализации равен 6. В 2008 г. опубликована работа [18], в которой исследователями Microsoft проведён анализ, подтверждающий теорию о шести рукопожатиях. Основываясь на данных, полученных за месяц общения 180 миллионов пользователей мессенджера Microsoft Instant Messenger, исследователи пришли к выводу, что наиболее короткий путь, связывающий любых двух пользователей этой сети, составляет 6,6 человек. Аналогичные исследования проводились на базе социальной сети Facebook: в 2011 г. сеть состояла из 721 миллиона пользователей и размер минимальной цепочки был 4,7 [19], а в 2016 г. этот параметр стал равен 3,5, когда число пользователей достигло 1,59 миллиарда [20].

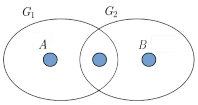
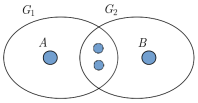
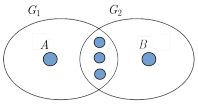
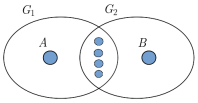
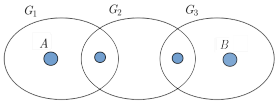
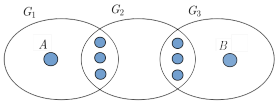
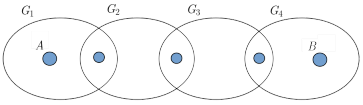
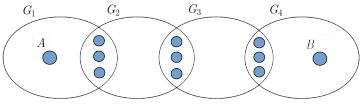
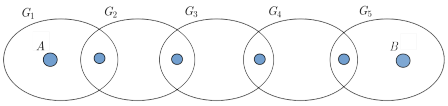
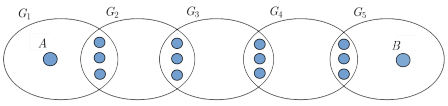
4.1. Тестирование

Тестирование проведено в браузере Chromium 81.0.4044.138. Были созданы конфигурации цепочек, описанные в табл. 2. В браузере запускались несколько экземпляров приложения — по числу пользователей, изображённых в этой таблице. Приведено время работы алгоритма аутентификации с помощью цепочек групп по уже установленной цепочке.

На рис. 2 приведён график зависимости времени работы алгоритма аутентификации от числа пользователей в пересечении групп, построенный по данным табл. 2 (конфигурации № 1–4), а на рис. 3 — график зависимости времени работы алгоритма аутентификации от числа звеньев в цепочке. График *1* соответствует конфигурациям № 1, 5, 7 и 9, где в пересечениях по одному пользователю, график *2* — конфигурациям № 3, 6, 8 и 10, где в пересечениях по три пользователя. Из рис. 2 и 3 видно, что время аутентификации возрастает практически линейно как с увеличением числа пользователей в пересечении групп, так и с увеличением числа звеньев в цепочке из групп.

Таблица 2

Время процесса парной аутентификации A и B по установленной цепочке

№ п/п	Конфигурация	Время, мс
1		2899
2		7006
3		14706
4		17330
5		12074
6		40841
7		25022
8		90752
9		30717
10		130314

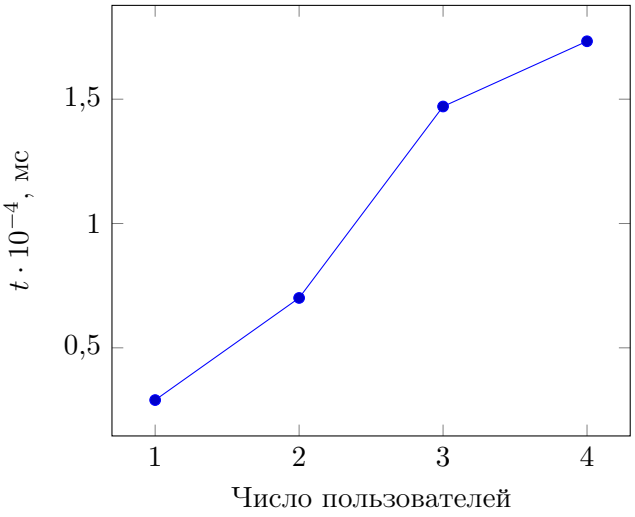


Рис. 2. Зависимость времени аутентификации t от числа пользователей в пересечении групп

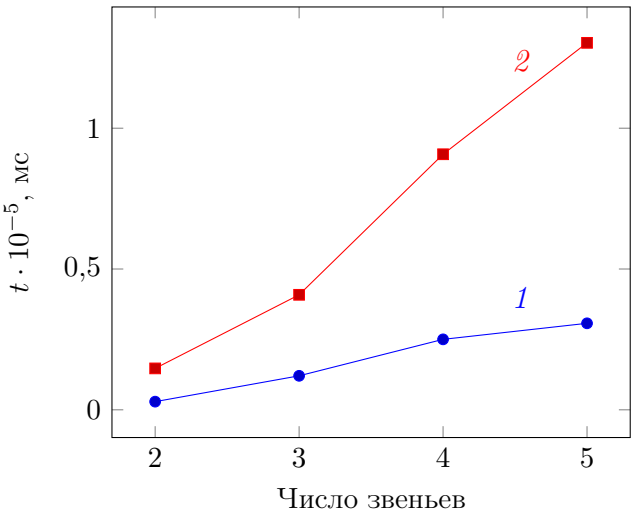


Рис. 3. Зависимость времени аутентификации t от числа звеньев в цепочке из групп пользователей

Заключение

В работе предложен алгоритм взаимной аутентификации пользователей без установления общего секрета по сторонним каналам и проведено исследование его устойчивости к атаке «человек в середине» и атаке Сивиллы.

Результаты экспериментального исследования на прототипе децентрализованной системы обмена сообщениями позволяют считать схему аутентификации применимой для сетей с развитыми социальными связями и большим количеством групп. Алгоритм взаимной аутентификации пользователей в децентрализованной системе может стать основой для создания полностью децентрализованных систем мгновенного обмена сообщениями с поддержкой свойств конфиденциальности в групповом общении, что в настоящее время не реализуется существующими системами мгновенного обмена сообщениями, прежде всего в силу наличия доверенной третьей стороны в виде провайдера сервиса.

В качестве направлений дальнейших исследований отметим вопрос безопасного хранения групповых ключей в распределённой одноранговой сети клиентов, часть

которых может быть инструментирована или модифицирована потенциальным злоумышленником. Для повышения эффективности предложенного алгоритма возможно использование механизма балансировки дерева группы, описанного в работе [13].

ЛИТЕРАТУРА

1. *Коростелева М. В., Гамаюнов Д. Ю.* Обеспечение криптографически защищенных групповых коммуникаций с функцией отказуемости // Проблемы информационной безопасности. Компьютерные системы. 2014. № 3. С. 74–79.
2. *Goldberg I. et al.* Multi-party off-the-record messaging // Proc. 16th Conf. Computer Commun. Security. ACM, 2009. P. 358–368.
3. *Шейдаев В. Ф., Гамаюнов Д. Ю.* Отказуемые групповые коммуникации в модели глобального неограниченного злоумышленника // Прикладная дискретная математика. 2018. № 40. С. 72–86.
4. `https://bitbucket.org/Enr1g/p2p_mpotr.js` — Moscow State University Seclab mpOTR.
5. *Нгуен К. К.* Доказательство с нулевым разглашением для взаимной аутентификации пользователей группового чата. Выпускная квалификационная работа. М.: МГУ, ВМК, 2018.
6. *Boudot F., Schoenmakers B., and Traore J.* A fair and efficient solution to the socialist millionaires' problem // Discr. Appl. Math. 2001. V. 111. No. 1. P. 23–36.
7. *Manulis M.* Democratic group signatures: on an example of joint ventures // Proc. 2006 ACM Symp. Inform. Comput. Commun. Security. 2006. P. 365.
8. `https://webrtc.org/` — Real-time communication for the web.
9. *Stoica I. et al.* Chord: A scalable peer-to-peer lookup service for internet applications // ACM SIGCOMM Comput. Commun. Rev. 2001. V. 31. No. 4. P. 149–160.
10. *Alves-Foss J.* An efficient secure authenticated group key exchange algorithm for large and dynamic groups // Proc. 23rd National Inform. Systems Security Conf. 2000. P. 254–266.
11. *Kim Y., Perrig A., and Tsudik G.* Communication-efficient group key agreement // IFIP Intern. Inform. Security Conf. Boston, MA: Springer, 2001. P. 229–244.
12. *Kim Y., Perrig A., and Tsudik G.* Group key agreement efficient in communication // IEEE Trans. Computers. 2004. V. 53. No. 7. P. 905–921.
13. *Zheng S., Manz D., and Alves-Foss J.* A communication-computation efficient group key algorithm for large and dynamic groups // Computer Networks. 2007. V. 51. No. 1. P. 69–93.
14. *Camenisch J. and Stadler M.* Efficient group signature schemes for large groups // Ann. Intern. Cryptology Conf. Berlin; Heidelberg: Springer, 1997. P. 410–424.
15. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems // Conf. Theory Appl. Cryptogr. Techniques. Berlin; Heidelberg: Springer, 1986. P. 186–194.
16. *Boneh D., Boyen X., and Shacham H.* Short group signatures // Ann. Intern. Cryptology Conf. Berlin; Heidelberg: Springer, 2004. P. 41–55.
17. `https://github.com/naruneph/Chord_chat` — Реализация метода цепочек групп в децентрализованном чате.
18. *Leskovec J. and Horvitz E.* Planetary-scale views on an instant-messaging network // Proc. 17th Intern. Conf. World Wide Web. 2008. P. 915–924.
19. *Ugander J. et al.* The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503. 2011.
20. `https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/` — Three and a half degrees of separation — Facebook Research. 2016.

REFERENCES

1. *Korosteleva M. V. and Gamayunov D. Y.* Obespecheniye kriptograficheski zashchishchennykh gruppovykh kommunikatsiy s funktsiyey otkazuyemosti [Protocol for secure group communications with deniability features]. Problemy Informatsionnoy Bezopasnosti. Komp'yuternyye Sistemy, 2014, no. 3, pp. 74–79. (in Russian)
2. *Goldberg I. et al.* Multi-party off-the-record messaging. Proc. 16th Conf. Comput. Commun. Security, ACM, 2009, pp. 358–368.
3. *Sheidaev V. F. and Gamayunov D. Y.* Otkazuemye gruppovye kommunikacii v modeli global'nogo neogranichennogo zloumyshlennika [Deniable group communications in the presence of global unlimited advisory]. Prikladnaya Diskretnaya Matematika, 2018, no. 40, pp. 72–86. (in Russian)
4. `https://bitbucket.org/Enr1g/p2p_mpotr.js` — Moscow State University Seclab mpOTR.
5. *Nguen K. K.* Dokazatel'stvo s nulevym razglasheniem dlya vzaimnoj autentifikacii pol'zovatelej gruppovogo chata [Zero-knowledge proof based authenticathion for group chat users]. Graduation Project, Moscow, MSU, 2018. (in Russian)
6. *Boudot F., Schoenmakers B., and Traore J.* A fair and efficient solution to the socialist millionaires' problem. Discr. Appl. Math., 2001, vol. 111, no. 1, pp. 23–36.
7. *Manulis M.* Democratic group signatures: on an example of joint ventures. Proc. 2006 ACM Symp. Inform. Comput. Commun. Security, 2006, p. 365.
8. `https://webrtc.org/` — Real-time communication for the web.
9. *Stoica I. et al.* Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev., 2001, vol. 31, no. 4, pp. 149–160.
10. *Alves-Foss J.* An efficient secure authenticated group key exchange algorithm for large and dynamic groups. Proc. 23rd National Inform. Systems Security Conf., 2000, pp. 254–266.
11. *Kim Y., Perrig A., and Tsudik G.* Communicationefficient group key agreement. IFIP Intern. Inform. Security Conf., Boston, MA, Springer, 2001, pp. 229–244.
12. *Kim Y., Perrig A., and Tsudik G.* Group key agreement efficient in communication. IEEE Trans. Computers, 2004, vol. 53, no. 7, pp. 905–921.
13. *Zheng S., Manz D., and Alves-Foss J.* A communication-computation efficient group key algorithm for large and dynamic groups. Computer Networks, 2007, vol. 51, no. 1, pp. 69–93.
14. *Camenisch J. and Stadler M.* Efficient group signature schemes for large groups. Ann. Intern. Cryptology Conf., Berlin, Heidelberg, Springer, 1997, pp. 410–424.
15. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems. Conf. Theory Appl. Cryptogr. Techniques, Berlin, Heidelberg, Springer, 1986, pp. 186–194.
16. *Boneh D., Boyen X., and Shacham H.* Short group signatures. Ann. Intern. Cryptology Conf., Berlin, Heidelberg, Springer, 2004, pp. 41–55.
17. `https://github.com/naruneph/Chord_chat` — Chain of groups method realisation for decentralized chat.
18. *Leskovec J. and Horvitz E.* Planetary-scale views on an instant-messaging network. Proc. 17th Intern. Conf. World Wide Web, 2008, pp. 915–924.
19. *Ugander J. et al.* The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503, 2011.
20. `https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/` — Three and a half degrees of separation — Facebook Research, 2016.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ НАДЁЖНОСТИ ВЫЧИСЛИТЕЛЬНЫХ И УПРАВЛЯЮЩИХ СИСТЕМ

УДК 519.718.7

О СХЕМАХ, ДОПУСКАЮЩИХ КОРОТКИЕ ЕДИНИЧНЫЕ ПРОВЕРЯЮЩИЕ ТЕСТЫ ПРИ ПРОИЗВОЛЬНЫХ НЕИСПРАВНОСТЯХ ФУНКЦИОНАЛЬНЫХ ЭЛЕМЕНТОВ[1]

К. А. Попков

Институт прикладной математики им. М. В. Келдыша РАН, г. Москва, Россия

Доказано, что любую неконстантную булеву функцию от n переменных можно реализовать неизбыточной схемой из функциональных элементов в базисе $\{\&, \oplus, \neg\}$, содержащей не более одной фиктивной входной переменной и допускающей единичный проверяющий тест длины не более $2n + 3$ относительно произвольных неисправностей элементов.

Ключевые слова: *схема из функциональных элементов, булева функция, неисправность, единичный проверяющий тест.*

DOI 10.17223/20710410/51/4

ON LOGIC NETWORKS ALLOWING SHORT SINGLE FAULT DETECTION TESTS UNDER ARBITRARY FAULTS OF GATES

K. A. Popkov

Keldysh Institute of Applied Mathematics, Moscow, Russia

E-mail: kirill-formulist@mail.ru

It is proved that one can implement any non-constant Boolean function in n variables by an irredundant logic network in the basis $\{\&, \oplus, \neg\}$, containing not more than one dummy input variable and allowing a single fault detection test with length not more than $2n + 3$ regarding arbitrary faults of logic gates.

Keywords: *logic network, Boolean function, fault, single fault detection test.*

Введение

В работе рассматривается задача синтеза легкотестируемых схем, реализующих заданные булевы функции. Логический подход к тестированию электрических схем предложен С. В. Яблонским и И. А. Чегис в [1]; этот подход также применим к тестированию схем из функциональных элементов (СФЭ) [2 – 4]. Пусть имеется СФЭ S с одним выходом, реализующая булеву функцию $f(\tilde{x}^n)$, где $\tilde{x}^n = (x_1, \ldots, x_n)$. Представим, что под воздействием некоторого источника неисправностей один из элементов

[1]Работа выполнена при поддержке гранта РНФ, проект № 19-71-30004.

схемы S может перейти в неисправное состояние. В результате данная схема вместо исходной функции $f(\tilde{x}^n)$ будет реализовывать некоторую булеву функцию $g(\tilde{x}^n)$, вообще говоря, отличную от f. Все такие функции $g(\tilde{x}^n)$ называются *функциями неисправности* схемы S. *Единичным проверяющим тестом* (ЕПТ) для схемы S называется такое множество T наборов значений переменных $x_1, \ldots, x_n$, что для любой отличной от $f(\tilde{x}^n)$ функции неисправности $g(\tilde{x}^n)$ данной схемы в T найдётся набор $\tilde{\sigma}$, на котором $f(\tilde{\sigma}) \neq g(\tilde{\sigma})$. Число наборов в T называется *длиной* теста. В качестве тривиального ЕПТ длины 2^n для схемы S всегда можно взять множество, состоящее из всех двоичных наборов длины n. Единичные проверяющие тесты обычно рассматривают для *неизбыточных схем* [4, с. 110–111], т. е. для таких схем, в которых любая допустимая неисправность любого одного элемента приводит к функции неисправности, отличной от исходной функции, реализуемой данной схемой.

Любое множество булевых функций будем называть *базисом*.

Пусть зафиксированы вид неисправностей элементов и функционально полный базис B, в котором строятся схемы, и T — ЕПТ для некоторой СФЭ S в базисе B. Введём следующие обозначения [4]: $D(T)$ — длина теста T; $D(S) = \min D(T)$, где минимум берётся по всем ЕПТ T для схемы S; $D(f) = \min D(S)$, где минимум берётся по всем неизбыточным схемам S в базисе B, реализующим функцию f; $D(n) = \max D(f)$, где максимум берётся по всем булевым функциям f от n переменных, для которых определено значение $D(f)$. Функция $D(n)$ называется *функцией Шеннона* длины ЕПТ.

Ранее в качестве неисправностей функциональных элементов традиционно рассматривались константные либо инверсные неисправности на входах и/или выходах элементов. Константная неисправность на входе (выходе) функционального элемента означает, что значение на этом входе (выходе) становится равно некоторой булевой константе. Неисправности на входах и/или выходах элементов называются однотипными константными типа p, если эта константа одна и та же для каждого неисправного входа/выхода элемента и равна p, и произвольными константными, если эта константа может быть равна как 0, так и 1 для каждого неисправного входа/выхода элемента независимо от неисправностей других входов/выходов элементов. Инверсная неисправность на входе (выходе) функционального элемента означает, что значение на этом входе (выходе) меняется на противоположное по сравнению со случаем, когда данный элемент исправен.

Перечислим основные результаты, касающиеся ЕПТ для схем из функциональных элементов. Для удобства разобьём эти результаты на группы в соответствии с видом неисправностей элементов, рассматривавшихся в схемах. Предполагаем, что n — произвольное натуральное число.

1. *Однотипные константные неисправности типа p, $p \in \{0, 1\}$, на выходах элементов.* Ю. В. Бородина в [5] при $B = \{\&, \oplus, 1, 0\}$, $p = 1$ установила равенство $D(n) = 1$. В [6, теорема 2] для любого полного базиса B, содержащего монотонные булевы функции, кроме константы 1, из которых отождествлением и переименованием переменных можно получить функции $x\&y$ и $x \vee y$, а также содержащего функции вида $\overline{x}_1 \vee h$, где h — произвольная булева функция, и для любого $n \geqslant 2$ получено равенство $D(n) = 2$ при $p = 1$. В [7] при $B = \{\&, \neg\}$ и любых $p \in \{0, 1\}$, $n \geqslant 2$ доказано равенство $D(n) = 3$.

2. *Произвольные константные неисправности на выходах элементов.* В [4, с. 116, теорема 10] с использованием метода синтеза СФЭ, предложенного С. М. Редди [8], для базиса $B = \{\&, \oplus, 1, 0\}$ получена оценка $D(n) \leqslant n + 3$. В дальнейшем этот результат

был обобщён С. С. Колядой в [9, 10] на случай произвольного полного конечного базиса B при $n \geqslant 3$. Д. С. Романов в [11] для любого полного базиса B установил соотношение $2 \leqslant \hat{D}(n) \leqslant 4$; наличие крышки над буквой D обусловлено тем, что в указанной работе рассматривается несколько другое определение неизбыточных схем. В [12, теорема 1] заявлено получение неравенства $\hat{D}(n) \leqslant 3$ для схем в произвольном полном конечном базисе при $n \geqslant 2$. В [6, теорема 4] для произвольного полного базиса B, состоящего из функций от не более чем двух переменных, доказано неравенство $D(n) \geqslant 3$ при $n \geqslant 3$. В [13, следствие 1] при $B = \{x\&y, \overline{x}, x \oplus y \oplus z\}$ установлено, что $D(n) = 2$.

3. *Инверсные неисправности на выходах элементов.* С. В. Коваценко для базиса $B = \{\&, \oplus, 1, 0\}$ установил равенство $D(n) = 1$ [14]. Н. П. Редькин в [15] при $B = \{\&, \vee, \neg\}$ получил оценку $D(n) \leqslant 2$, а в [16] для схем в произвольном полном конечном базисе — оценку $D(n) \leqslant 3$.

4. *Однотипные константные неисправности типа* 0 *на входах и выходах элементов.* В [17, следствие 1] для базиса $B = \{xyz \vee \overline{x}\,\overline{y}\,\overline{z}, \overline{x}\}$ установлено равенство $D(n) = 2$ (в предположении, что параметр k, определённый в указанной работе и равный максимально возможному числу неисправностей в схемах, равен 1. Точно так же этот параметр определяется в работе [20], ссылку на которую см. в следующем абзаце).

5. *Произвольные константные неисправности на входах и выходах элементов.* В рассуждениях на с. 116 работы [4], следующих после формулировки теоремы 10, показано, что $D(n) \leqslant n + 3$ для схем в базисе $\{\&, \oplus, 1, 0\}$. В [18, 19] для схем в любом из базисов $\{\&, \oplus, 1\}$, $\{\&, \oplus, \sim\}$ получено неравенство $D(n) \leqslant 16$. В [20, следствие 2] (при $k = 1$) для базиса $B = \{\varphi(x, y, z, t), x \sim y, \overline{x}, 0\}$, где $\varphi(x, y, z, t)$ — некоторая булева функция, и любого $n \geqslant 2$ установлено равенство $D(n) = 3$.

6. *Инверсные неисправности на входах и выходах элементов.* В [21, утверждение 2] доказано равенство $D(n) = 1$ при $B = \{\&, \oplus, 1\}$.

В отличие от всех перечисленных работ, в данной работе рассматриваются произвольные неисправности функциональных элементов: предполагается, что каждый неисправный элемент E вместо исходной приписанной ему булевой функции $\varphi_E(\tilde{x}^m)$ реализует произвольную другую булеву функцию $\varphi'_E(\tilde{x}^m)$ (от своих входов). Таким образом, элемент E может находиться в любом (неизменном в ходе тестирования) из $2^{2^m} - 1$ неисправных состояний, характеризующихся функцией $\varphi'_E(\tilde{x}^m)$. Например, в случае константной неисправности типа p (в случае инверсной неисправности) на выходе этого элемента имеем $\varphi'_E \equiv p$ (соответственно $\varphi'_E = \overline{\varphi}_E$), а в случае константной неисправности типа p (инверсной неисправности) на входе элемента E, отвечающем переменной x_1, имеем $\varphi'_E(\tilde{x}^m) = \varphi_E(p, x_2, \ldots, x_m)$ (соответственно $\varphi'_E(\tilde{x}^m) = \varphi_E(\overline{x}_1, x_2, \ldots, x_m)$).

В соответствии с [22, с. 105] будем говорить, что СФЭ *содержит k фиктивных входных переменных и реализует функцию $f(\tilde{x}^n)$*, если данная схема содержит k входных переменных, отличных от переменных $x_1, \ldots, x_n$, и реализует булеву функцию, не зависящую существенно от этих k переменных и равную функции $f(\tilde{x}^n)$. Будем также предполагать, что все наборы из любого ЕПТ для такой схемы имеют длину $n + k$ (по общему числу её входных переменных).

Введём обозначения $\tilde{0}^l = \underbrace{0, \ldots, 0}_{l}$, $\tilde{1}^l = \underbrace{1, \ldots, 1}_{l}$, где $l \in \mathbb{N} \cup \{0\}$. В случае $l = 0$ они обозначают пустую строку: например, $(\tilde{1}^n, \tilde{0}^0) = (\tilde{1}^n)$.

Будем говорить, что функциональный элемент E является *верхним* (*нижним*) элементом некоторой цепочки, если ни один вход элемента E не соединён с выходом

какого-то элемента из этой цепочки (соответственно если выход элемента E не соединён ни с одним входом ни одного элемента из этой цепочки).

Вместо «вход схемы S, отвечающий переменной x_i» для краткости будем писать «вход „x_i“ схемы S».

Формулировка и доказательство основного результата

Рассмотрим базис $B = \{\&, \oplus, \neg\}$. Любой функциональный элемент, реализующий функцию вида $x\&y$ (вида $x \oplus y$, $\overline{x}$) от своих входов, будем называть *конъюнктором* (соответственно *сумматором*, *инвертором*). Вход любого конъюнктора или сумматора, отвечающий переменной x, будем считать *левым*, а другой вход этого элемента — *правым*. Отметим, что для любой булевой переменной x справедливо тождество

$$\overline{x} \equiv x \oplus 1. \tag{1}$$

Сформулируем основной результат данной работы.

Теорема 1. Любую неконстантную булеву функцию $f(\tilde{x}^n)$ можно реализовать:

1) неизбыточной СФЭ в базисе B, содержащей одну фиктивную входную переменную и допускающей ЕПТ длины не более $2n + 4$, подмножеством которого является множество $\{(\tilde{0}^r, \tilde{1}^{n+1-r}), (\tilde{1}^s, 0, \tilde{1}^{n-s}) : r \in \{0, \ldots, n+1\}, s \in \{1, \ldots, n\}\}$;
2) неизбыточной СФЭ в базисе B, содержащей одну фиктивную входную переменную и допускающей ЕПТ длины не более $2n + 3$.

Замечание 1. Константные булевы функции ни для какого $k \in \mathbb{N} \cup \{0\}$ нельзя реализовать неизбыточными СФЭ в базисе B, содержащими не более k фиктивных входных переменных. Действительно, константная неисправность типа α на выходе выходного элемента любой схемы в базисе B, содержащей k' фиктивных входных переменных и реализующей функцию $f(\tilde{x}^n) \equiv \alpha$, где $0 \leqslant k' \leqslant k$, $\alpha \in \{0, 1\}$, т. е. схемы, реализующей функцию $f'(\tilde{x}^{n+k'}) \equiv \alpha$, приводит к тривиальной функции неисправности $g(\tilde{x}^{n+k'}) \equiv \alpha$.

Замечание 2. У автора есть гипотеза, что любую неконстантную булеву функцию $f(\tilde{x}^n)$ можно реализовать неизбыточной СФЭ (без фиктивных входных переменных) в базисе B, допускающей ЕПТ линейной по n длины. Утверждение 1 теоремы 1 сформулировано именно с целью его использования в предполагаемом доказательстве данной гипотезы.

Для доказательства теоремы 1 введём некоторые обозначения и докажем две вспомогательные леммы. Пусть $f(\tilde{x}^n)$ — произвольная неконстантная булева функция. Представим её полиномом Жегалкина

$$f(\tilde{x}^n) = K_1 \oplus \ldots \oplus K_m \oplus c, \tag{2}$$

где $m \geqslant 1$; $c \in \{0, 1\}$; K_1 — самая короткая конъюнкция в этом полиноме. Пусть $K_i = x_{j_1(i)}\& \ldots \& x_{j_{t_i}(i)}$, где $i = 1, \ldots, m$ и

$$1 \leqslant j_1(i) < \ldots < j_{t_i}(i) \leqslant n; \tag{3}$$

$\tilde{\sigma}_{K_1}$ — двоичный $(n+1)$-разрядный набор, содержащий единицы только в компонентах с номерами $j_1(1), \ldots, j_{t_1}(1)$, а $f^{(+1)}(\tilde{x}^{n+1})$ — булева функция, не зависящая существенно от переменной x_{n+1} и равная функции $f(\tilde{x}^n)$.

В случае $m \geqslant 2$ введём следующие обозначения: $h(\tilde{x}^n) = K_2 \oplus \ldots \oplus K_m$;

$$a = \begin{cases} 1, & \text{если } K_1 \leqslant h \text{ или } K_1 \leqslant h \oplus 1, \\ 0 & \text{иначе.} \end{cases}$$

Неравенство $f_1 \leqslant f_2$, где $f_1(\tilde{x}^n)$, $f_2(\tilde{x}^n)$ — булевы функции, означает, что $f_1(\tilde{\pi}) \leqslant f_2(\tilde{\pi})$ для любого двоичного n-разрядного набора $\tilde{\pi}$.

Лемма 1. Существует такой двоичный n-разрядный набор $\tilde{\rho}$, что в случае $a = 1$ выполнены равенства $K_1(\tilde{\rho}) = 0$ и $h(\tilde{\rho}) = 1$, а в случае $a = 0$ — равенства $K_1(\tilde{\rho}) = 1$ и $h(\tilde{\rho}) = \overline{h(\tilde{1}^n)}$.

Доказательство. Пусть сначала $a = 1$. Заметим, что функция K_1 отлична от функции h, поскольку в противном случае в силу (2) выполнялось бы соотношение

$$0 \equiv K_1 \oplus h = K_1 \oplus K_2 \oplus \ldots \oplus K_m = f \oplus c,$$

однако это противоречит тому, что функция f отлична от константы c. Кроме того, функция h отлична от константы 0 в силу единственности представления каждой булевой функции полиномом Жегалкина [23, с. 32]. Равенство $a = 1$ влечёт за собой выполнение одного из двух неравенств: $K_1 \leqslant h$ или $K_1 \leqslant h \oplus 1$. В случае $K_1 \leqslant h$ с учётом различия функций K_1 и h существует такой n-разрядный набор $\tilde{\rho}$, для которого $K_1(\tilde{\rho}) = 0$ и $h(\tilde{\rho}) = 1$, что и требовалось доказать. В случае $K_1 \leqslant h \oplus 1$ возьмём в качестве $\tilde{\rho}$ любой набор, на котором функция $h(\tilde{x}^n)$ принимает значение 1. Тогда

$$K_1(\tilde{\rho}) \leqslant h(\tilde{\rho}) \oplus 1 = 1 \oplus 1 = 0,$$

т. е. $K_1(\tilde{\rho}) = 0$ и $h(\tilde{\rho}) = 1$, что и требовалось доказать.

Пусть теперь $a = 0$. Тогда не выполнено ни одно из неравенств $K_1 \leqslant h$, $K_1 \leqslant h \oplus 1$, поэтому существуют такие n-разрядные наборы $\tilde{\pi}_1$ и $\tilde{\pi}_2$, что $K_1(\tilde{\pi}_1) = K_1(\tilde{\pi}_2) = 1$, $h(\tilde{\pi}_1) = 0$ и $h(\tilde{\pi}_2) \oplus 1 = 0$, т. е. $h(\tilde{\pi}_2) = 1$. Положим

$$\tilde{\rho} = \begin{cases} \tilde{\pi}_1, & \text{если } h(\tilde{1}^n) = 1, \\ \tilde{\pi}_2, & \text{если } h(\tilde{1}^n) = 0. \end{cases}$$

Тогда $K_1(\tilde{\rho}) = 1$ и $h(\tilde{\rho}) = \overline{h(\tilde{1}^n)}$. Лемма 1 доказана. ■

Обозначим через $\tilde{\tau}$ двоичный $(n+1)$-разрядный набор, получающийся из набора $\tilde{\rho}$ добавлением справа $(n+1)$-й компоненты, равной единице.

Лемма 2. Любую булеву функцию $f(\tilde{x}^n)$, в представлении (2) которой $m \geqslant 2$, можно реализовать неизбыточной схемой в базисе B, содержащей одну фиктивную входную переменную и допускающей ЕПТ $T = \{(\tilde{0}^r, \tilde{1}^{n+1-r}), (\tilde{1}^s, 0, \tilde{1}^{n-s}), \tilde{\sigma}_{K_1}, \tilde{\tau} : r \in \in \{0, \ldots, n+1\}, s \in \{1, \ldots, n\}\}$ (длины не более $2n + 4$).

Доказательство. Представим конъюнкцию K_1 в виде $K_1\overline{x}_{n+1} \oplus K_1 x_{n+1}$, где x_{n+1} — булева переменная, отличная от $x_1, \ldots, x_n$. Тогда представление (2) примет вид

$$f^{(+1)}(\tilde{x}^{n+1}) = K_1\overline{x}_{n+1} \oplus K_1 x_{n+1} \oplus K_2 \oplus \ldots \oplus K_m \oplus c.$$

Перепишем последнее равенство в виде

$$f^{(+1)}(\tilde{x}^{n+1}) = K_1\overline{x}_{n+1} \oplus K_2 \oplus \ldots \oplus K_m \oplus x_{n+1}(K_1 \oplus a) \oplus a x_{n+1} \oplus c. \qquad (4)$$

Реализуем функцию $f^{(+1)}$ схемой S в базисе B в соответствии с представлением (4) (см. рис. 1). Каждую конъюнкцию K_i, $i = 1, \ldots, m$, реализуем цепочкой Z_i из $t_i - 1$ конъюнкторов $E_{i,1}, \ldots, E_{i,t_i-1}$, занумерованных «сверху вниз», на входы которой последовательно подаются переменные $x_{j_1(i)}, \ldots, x_{j_{t_i}(i)}$, причём в случае $t_i \geqslant 2$ переменные $x_{j_2(i)}, \ldots, x_{j_{t_i}(i)}$ подаются на правые входы этих конъюнкторов (тогда переменная $x_{j_1(i)}$ будет подана на левый вход элемента $E_{i,1}$ — верхнего конъюнктора цепочки Z_i; в случае $t_i = 1$ в этой цепочке не содержится элементов, а её выход совпадает со входом «$x_{j_1(i)}$» схемы S). Переменную x_{n+1} подадим на вход инвертора I; выход цепочки Z_1 и выход элемента I соединим соответственно с левым и правым входами конъюнктора $E_{\&}$. Если $a = 1$, то выход цепочки Z_1 соединим также со входом инвертора I'. В случае $a = 0$ выход цепочки Z_1, а в случае $a = 1$ — выход элемента I' соединим с левым входом конъюнктора $E'_{\&}$, на правый вход которого подадим переменную x_{n+1}. Затем выход элемента $E_{\&}$, выходы всех цепочек $Z_2, \ldots, Z_m$, выход элемента $E'_{\&}$ и — в случае $a = 1$ — вход «x_{n+1}» схемы последовательно соединим со входами цепочки $Z_{\oplus}$, состоящей из сумматоров $E^{\oplus}_1, \ldots, E^{\oplus}_{m+a}$, занумерованных «сверху вниз», и — в случае $c = 1$ — из инвертора I'', вход которого соединён с выходом элемента $E^{\oplus}_{m+a}$, причём выходы цепочек $Z_2, \ldots, Z_m$, выход конъюнктора $E'_{\&}$ и — в случае $a = 1$ — вход «x_{n+1}» схемы соединим с правыми входами сумматоров $E^{\oplus}_1, \ldots, E^{\oplus}_{m+a}$ соответственно (тогда выход конъюнктора $E_{\&}$ соединяется с левым входом сумматора $E^{\oplus}_1$). Выход нижнего элемента цепочки $Z_{\oplus}$ объявим выходом схемы S.

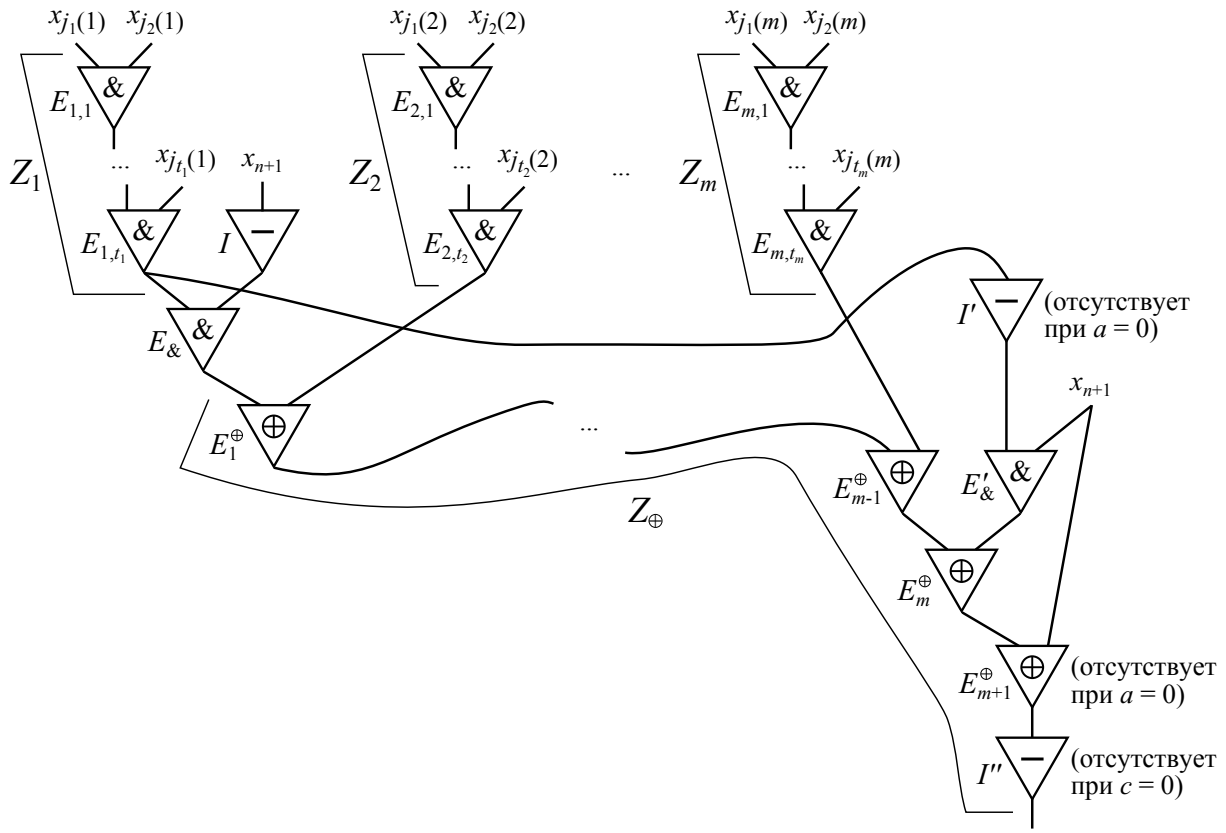


Рис. 1. Схема S

Из построения схемы, представления (4) и тождества (1) вытекает, что схема S при отсутствии в ней неисправностей реализует функцию $f^{(+1)}(\tilde{x}^{n+1})$. Тогда, согласно данному выше определению, схема S содержит одну фиктивную входную переменную

и реализует функцию $f(\tilde{x}^n)$. Докажем, что эта схема неизбыточна и допускает ЕПТ T относительно произвольных неисправностей элементов. Достаточно доказать, что любая неисправность любого одного элемента схемы S обнаруживается хотя бы на одном наборе из множества T.

Обозначим через $Z^d_\oplus$, $d = 1, \ldots, m + a + 1$, цепочку, являющуюся нижней частью цепочки $Z_\oplus$ и получающуюся из неё отбрасыванием сумматоров $E^\oplus_1, \ldots, E^\oplus_{d-1}$ (считаем, что в случае $d = 1$ цепочки $Z^d_\oplus$ и $Z_\oplus$ совпадают, а в случае $d = m + a + 1$ и $c = 0$ цепочка $Z^d_\oplus$ не содержит элементов и её выход совпадает с её единственным входом и с выходом элемента $E^\oplus_{m+a}$).

Рассмотрим произвольную неисправность произвольного одного элемента E в схеме S. При этой неисправности на выходе элемента E реализуется булева функция (от его входов), отличающаяся от исходной приписанной ему функции хотя бы на одном наборе $\tilde{\alpha}$; данный набор содержит две компоненты (одну компоненту), если элемент E — конъюнктор или сумматор (соответственно если элемент E — инвертор). Для любого $d \in \{1, \ldots, m + a + 1\}$ цепочка $Z^d_\oplus$ при условии, что в ней не содержится элемента E, очевидно, реализует линейную булеву функцию от своих входов, поэтому изменение значения ровно на одном входе этой цепочки приводит к изменению значения на выходе всей схемы S. Таким образом, если на некотором входном наборе $\tilde{\sigma} \in T$ схемы S при переходе элемента E в неисправное состояние меняется значение ровно на одном входе цепочки $Z^d_\oplus$ для некоторого $d \in \{1, \ldots, m+a+1\}$, не содержащей этого элемента, то указанная неисправность обнаруживается на наборе $\tilde{\sigma}$. Возможны девять случаев:

1. Элемент E — конъюнктор $E_{i,q}$ для некоторых $i \in \{1, \ldots, m\}$, $q \in \{1, \ldots, t_i - 1\}$ (и при этом $t_i \geqslant 2$). В силу построения схемы S на правый вход данного конъюнктора подаётся переменная $x_{j_{q+1}(i)}$, а его левый вход соединяется либо со входом «$x_{j_1(i)}$» схемы (в случае $q = 1$), либо с выходом цепочки из конъюнкторов $E_{i,1}, \ldots, E_{i,q-1}$, на входы которой подаются переменные $x_{j_1(i)}, \ldots, x_{j_q(i)}$ (в случае $q \geqslant 2$); выход элемента $E_{i,q}$ либо является выходом цепочки Z_i (в случае $q = t_i - 1$), либо соединяется с одним из входов цепочки из конъюнкторов $E_{i,q+1}, \ldots, E_{i,t_i-1}$, на другие входы которой подаются переменные $x_{j_{q+2}(i)}, \ldots, x_{j_t(i)}$ и выход которой совпадает с выходом цепочки Z_i (в случае $q \leqslant t_i - 2$). При этом выполнено соотношение (3). Отсюда следует, что при последовательной подаче на входы схемы S наборов

$$(\tilde{0}^{j_{q+1}(i)}, \tilde{1}^{n+1-j_{q+1}(i)}), (\tilde{0}^{j_{q+1}(i)-1}, \tilde{1}^{n+2-j_{q+1}(i)}), (\tilde{1}^{j_{q+1}(i)-1}, 0, \tilde{1}^{n+1-j_{q+1}(i)}), (\tilde{1}^{n+1}) \in T$$

на входах элемента E возникают наборы $(0,0), (0,1), (1,0), (1,1)$ соответственно, а значение на выходе элемента E совпадает со значением на выходе цепочки Z_i (поскольку в случае $q \leqslant t_i - 2$ у каждого из указанных четырёх наборов из множества T все компоненты, начиная с компоненты с номером $j_{q+1}(i) + 1$, равны 1 и $j_{q+1}(i) + 1 \leqslant j_{q+2}(i) < \ldots < j_t(i)$, т. е. на правые входы всех конъюнкторов $E_{i,q+1}, \ldots, E_{i,t_i-1}$ поступают единицы). Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$, а значение на выходе элемента E совпадает со значением на выходе цепочки Z_i, причём $(n+1)$-я (последняя) компонента набора $\tilde{\sigma}$ равна 1, так как $j_{q+1}(i) \leqslant n$. Рассмотрим два подслучая:

1.1. Пусть $i \in \{2, \ldots, m\}$. На входном наборе $\tilde{\sigma}$ схемы S при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе цепочки Z_i, а значит, ровно на одном входе цепочки $Z^1_\oplus$, что и требовалось доказать.

1.2. Пусть $i = 1$. Выход цепочки Z_1 в схеме S соединяется с левым входом конъюнктора $E_\&$, а также — непосредственно или через инвертор I' — с левым входом конъ-

юнктора $E'_{\&}$. Правый вход конъюнктора $E_{\&}$ соединяется с выходом инвертора I, и на этом выходе реализуется функция $\overline{x}_{n+1}$, принимающая значение 0 на наборе $\tilde{\sigma}$. Поэтому при подаче на входы схемы S набора $\tilde{\sigma}$ на выходе конъюнктора $E_{\&}$ возникает нуль вне зависимости от значения, возникающего на выходе цепочки Z_1. Далее, на правый вход конъюнктора $E'_{\&}$ подаётся переменная x_{n+1}, равная единице на наборе $\tilde{\sigma}$. Поэтому при подаче на входы схемы S данного набора значение на выходе конъюнктора $E'_{\&}$ в случае $a = 0$ совпадает со значением на выходе цепочки Z_1, а в случае $a = 1$ — со значением на выходе инвертора I' и противоположно значению на выходе цепочки Z_1. В каждом из случаев $a = 0$, $a = 1$ на входном наборе $\tilde{\sigma}$ схемы S при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе цепочки Z_1, а значит, и на выходе конъюнктора $E'_{\&}$, т. е. ровно на одном входе цепочки $Z^1_{\oplus}$, что и требовалось доказать.

2. Элемент E — инвертор I. При последовательной подаче на входы схемы S наборов $(\tilde{1}^n, 0), (\tilde{1}^{n+1}) \in T$ на входах элемента E, очевидно, возникают наборы $(0), (1)$ соответственно, а значение на выходе элемента E совпадает со значением на выходе конъюнктора $E_{\&}$ (поскольку на каждом из указанных двух наборов из множества T на выходе цепочки Z_1 возникает значение 1, которое подаётся на левый вход конъюнктора $E_{\&}$). Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$, а значение на выходе элемента E совпадает со значением на выходе конъюнктора $E_{\&}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе данного конъюнктора, а значит, ровно на одном входе цепочки $Z^1_{\oplus}$, что и требовалось доказать.

3. Элемент E — инвертор I' (и при этом $a = 1$). При последовательной подаче на входы схемы S наборов $(\tilde{0}^n, 1), (\tilde{1}^{n+1}) \in T$ на входах элемента E, очевидно, возникают наборы $(0), (1)$ соответственно, а значение на выходе элемента E совпадает со значением на выходе конъюнктора $E'_{\&}$ (поскольку на каждом из указанных двух наборов из множества T переменная x_{n+1} принимает значение 1, которое подаётся на правый вход конъюнктора $E'_{\&}$). Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$, а значение на выходе элемента E совпадает со значением на выходе конъюнктора $E'_{\&}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе данного конъюнктора, а значит, ровно на одном входе цепочки $Z^1_{\oplus}$, что и требовалось доказать.

4. Элемент E — конъюнктор $E_{\&}$. Легко видеть, что при последовательной подаче на входы схемы S наборов $(\tilde{0}^n, 1), (\tilde{0}^{n+1}), (\tilde{1}^{n+1}), (\tilde{1}^n, 0) \in T$ на входах элемента E возникают наборы $(0, 0), (0, 1), (1, 0), (1, 1)$ соответственно. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение ровно на одном входе цепочки $Z^1_{\oplus}$, что и требовалось доказать.

5. Элемент E — конъюнктор $E'_{\&}$. Легко видеть, что при последовательной подаче на входы схемы S наборов $(\tilde{0}^{n+1}), (\tilde{0}^n, 1), (\tilde{1}^n, 0), (\tilde{1}^{n+1}) \in T$ на входах элемента E возникают наборы $(0, 0), (0, 1), (1, 0), (1, 1)$ соответственно в случае $a = 0$ и наборы $(1, 0), (1, 1), (0, 0), (0, 1)$ — в случае $a = 1$. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение ровно на одном входе цепочки $Z^1_{\oplus}$, что и требовалось доказать.

6. Элемент E — сумматор $E_i^{\oplus}$ для некоторого $i \in \{1, \ldots, m-1\}$. Тогда правый вход элемента E по построению схемы S соединён с выходом цепочки Z_{i+1}, на котором реализуется конъюнкция K_{i+1}. При подаче на входы схемы набора $(\tilde{0}^{n+1}) \in T$ на входах элемента E, очевидно, возникает набор $(0,0)$. При подаче на входы схемы набора $\tilde{\sigma}_{K_1} \in T$ на выходе как цепочки Z_1, так и инвертора I, а значит, и на выходе конъюнктора $E_{\&}$ возникает значение 1 (см. определение набора $\tilde{\sigma}_{K_1}$; достаточно лишь отметить, что $j_1(1) \leqslant n, \ldots, j_{t_1}(1) \leqslant n$), а на выходе каждой из цепочек $Z_2, \ldots, Z_m$ — значение 0, поскольку K_1 — самая короткая конъюнкция в полиноме Жегалкина функции $f(\tilde{x}^n)$ и, как следствие, в каждую из конъюнкций $K_2, \ldots, K_m$ входит хотя бы одна переменная, отличная от $x_{j_1(1)}, \ldots, x_{j_{t_1}(1)}$. Отсюда вытекает, что на наборе $\tilde{\sigma}_{K_1}$ на входах каждого из сумматоров $E_1^{\oplus}, \ldots, E_i^{\oplus}$ возникает набор $(1,0)$. Далее, при последовательной подаче на входы схемы S наборов $(\tilde{1}^n, 0), (\tilde{1}^{n+1}) \in T$ на выходах всех цепочек $Z_1, \ldots, Z_m$ возникают единицы, а на выходе инвертора I — значения $1, 0$ соответственно; такие же значения возникают на выходе конъюнктора $E_{\&}$. Тогда на указанных двух наборах из множества T на входы сумматора $E_1^{\oplus}$ поступают наборы $(1,1), (0,1)$ соответственно; в случае $i \geqslant 2$ на выходе этого сумматора возникают значения $0, 1$ соответственно и на входы сумматора $E_2^{\oplus}$ поступают наборы $(0,1), (1,1)$; в случае $i \geqslant 3$ на выходе сумматора $E_2^{\oplus}$ возникают значения $1, 0$ соответственно и на входы сумматора $E_3^{\oplus}$ поступают наборы $(1,1), (0,1)$, и т. д. В итоге получаем, что на наборах $(\tilde{1}^n, 0), (\tilde{1}^{n+1})$ на входы сумматора $E_i^{\oplus}$, т. е. элемента E, при нечётном i поступают наборы $(1,1), (0,1)$ соответственно, а при чётном i — наборы $(0,1), (1,1)$ соответственно. С учётом предыдущих рассуждений делаем вывод, что при подаче на входы схемы S некоторых наборов из множества T на входах элемента E возникают все четыре набора $(0,0), (0,1), (1,0), (1,1)$. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение ровно на одном (а именно на левом верхнем) входе цепочки $Z_{\oplus}^{i+1}$, что и требовалось доказать.

7. Элемент E — сумматор $E_m^{\oplus}$. Тогда левый и правый входы элемента E по построению схемы S соединены с выходами элементов $E_{m-1}^{\oplus}$ и $E'_{\&}$ соответственно и на этих выходах реализуются функции $K_1\overline{x}_{n+1} \oplus K_2 \oplus \ldots \oplus K_m$ и $x_{n+1}(K_1 \oplus a)$ соответственно. При подаче на входы схемы набора $(\tilde{0}^{n+1}) \in T$ на входах элемента E, очевидно, возникает набор $(0,0)$. При подаче на входы схемы набора $\tilde{\sigma}_{K_1} \in T$ на правом входе элемента E возникает значение 0, поскольку $(n+1)$-я компонента данного набора равна нулю, а на левом входе — значение $1 \oplus \underbrace{0 \oplus \ldots \oplus 0}_{m-1} = 1$, так как K_1 — самая короткая конъюнкция в полиноме Жегалкина функции $f(\tilde{x}^n)$; следовательно, на входах элемента E возникает набор $(1,0)$. Докажем, что при подаче на входы схемы S некоторых двух наборов из множества T на входах элемента E возникают наборы $(0,1)$ и $(1,1)$. Рассмотрим два подслучая:

7.1. Пусть $a = 1$. На входном наборе $(\tilde{0}^n, 1) \in T$ схемы на входы элемента E поступает набор $(0,1)$, так как на этом наборе функции $K_1, K_2, \ldots, K_m, x_{n+1}, K_1 \oplus a$ (рассматриваемые как функции от переменных $x_1, \ldots, x_{n+1}$) принимают значения $\underbrace{0, \ldots, 0}_{m}, 1, 1$ соответственно. На входном наборе $\tilde{\tau} \in T$ схемы на входы элемента E поступает набор $(1,1)$, так как на этом наборе функции $K_1\overline{x}_{n+1}, K_2 \oplus \ldots \oplus K_m, x_{n+1}, K_1 \oplus a$ принимают значения $0, 1, 1, 1$ соответственно (см. определения наборов $\tilde{\tau}$, $\tilde{\rho}$ и функции $h(\tilde{x}^n)$).

7.2. Пусть $a = 0$. На входном наборе $(\tilde{1}^{n+1}) \in T$ схемы на входы элемента E поступает набор $(h(\tilde{1}^n), 1)$, так как на этом наборе функции $K_1\overline{x}_{n+1}, K_2 \oplus \ldots \oplus K_m, x_{n+1}, K_1 \oplus a$

принимают значения $0, h(\tilde{1}^n), 1, 1$ соответственно. На входном наборе $\tilde{\tau} \in T$ схемы на входы элемента E поступает набор $\left(\overline{h(\tilde{1}^n)}, 1\right)$, так как на этом наборе функции $K_1\overline{x}_{n+1}, K_2 \oplus \ldots \oplus K_m, x_{n+1}, K_1 \oplus a$ принимают значения $0, \overline{h(\tilde{1}^n)}, 1, 1$ соответственно (см. определения наборов $\tilde{\tau}$ и $\tilde{\rho}$).

В каждом из подслучаев 7.1 и 7.2 доказано, что при подаче на входы схемы S некоторых двух наборов из множества T на входах элемента E возникают наборы $(0,1)$ и $(1,1)$. С учётом предыдущих рассуждений из разбора случая 7 делаем вывод, что при подаче на входы схемы S некоторых наборов из множества T на входах элемента E возникают все четыре набора $(0,0), (0,1), (1,0), (1,1)$. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение ровно на одном (а именно на левом верхнем) входе цепочки $Z_{\oplus}^{m+1}$, что и требовалось доказать.

8. Элемент E — сумматор $E_{m+1}^{\oplus}$ (и при этом $a = 1$). Тогда левый и правый входы элемента E по построению схемы S соединены с выходом сумматора $E_m^{\oplus}$ и со входом «x_{n+1}» схемы соответственно и на эти входы элемента E подаются функции $K_1\overline{x}_{n+1} \oplus K_2 \oplus \ldots \oplus K_m \oplus x_{n+1}(K_1 \oplus 1)$ и x_{n+1} соответственно. Используя рассуждения из разбора случая 7 и подслучая 7.1, получаем, что при последовательной подаче на входы схемы S наборов $(\tilde{0}^{n+1}), \tilde{\sigma}_{K_1}, (\tilde{0}^n, 1), \tilde{\tau} \in T$ на входах элемента E возникают наборы $(0 \oplus 0, 0), (1 \oplus 0, 0), (0 \oplus 1, 1), (1 \oplus 1, 1)$ соответственно, т. е. наборы $(0,0), (1,0), (1,1), (0,1)$. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение ровно на одном (а именно на левом верхнем) входе цепочки $Z_{\oplus}^{m+2}$, что и требовалось доказать.

9. Элемент E — инвертор I'' (и при этом $c = 1$). Тогда вход элемента E по построению схемы S соединён с выходом сумматора $E_m^{\oplus}$ в случае $a = 0$ и с выходом сумматора $E_{m+1}^{\oplus}$ в случае $a = 1$. Используя рассуждения из разбора случаев 7 и 8, получаем, что в каждом из случаев $a = 0$, $a = 1$ при последовательной подаче на входы схемы S наборов $(\tilde{0}^{n+1}), \tilde{\sigma}_{K_1} \in T$ на входах элемента E возникают наборы $(0 \oplus 0), (1 \oplus 0)$ соответственно, т. е. наборы $(0), (1)$. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе данного элемента, совпадающем с выходом схемы S, что и требовалось доказать. Лемма 2 доказана. ■

Доказательство теоремы 1. Представим функцию f полиномом Жегалкина (2), где $m \geqslant 1$; $c \in \{0,1\}$; K_1 — самая короткая конъюнкция в этом полиноме. Рассмотрим два случая:

1. Пусть $m = 1$. Реализуем функцию $f^{(+1)}$ схемой S в базисе B в соответствии с представлением $f^{(+1)}(\tilde{x}^{n+1}) = K_1 \oplus c$. Конъюнкцию K_1 реализуем цепочкой Z из $t_1 - 1$ конъюнкторов $E_1, \ldots, E_{t_1-1}$, занумерованных «сверху вниз», на входы которой последовательно подаются переменные $x_{j_1(1)}, \ldots, x_{j_{t_1}(1)}$, причём в случае $t_1 \geqslant 2$ переменные $x_{j_2(1)}, \ldots, x_{j_{t_1}(1)}$ подаются на правые входы этих конъюнкторов (в случае $t_1 = 1$ в цепочке Z не содержится элементов, а её выход совпадает со входом «$x_{j_1(1)}$» схемы S). Если $c = 0$, то выход цепочки Z будем считать выходом схемы S; если $c = 1$, то соединим выход цепочки Z со входом инвертора I, выход которого будем считать выходом схемы S.

Из построения схемы и тождества (1) вытекает, что схема S при отсутствии в ней неисправностей реализует функцию $f^{(+1)}(\tilde{x}^{n+1})$. Тогда схема S содержит одну фиктивную входную переменную и реализует функцию $f(\tilde{x}^n)$. Докажем, что эта схема неизбыточна и допускает ЕПТ $T = \{(\tilde{0}^r, \tilde{1}^{n+1-r}), (\tilde{1}^s, 0, \tilde{1}^{n-s}) : r \in \{0, \ldots, n+1\}, s \in \{1, \ldots, n\}\}$ длины $2n+2$ относительно произвольных неисправностей элементов; отсюда следует справедливость утверждений 1 и 2 теоремы в случае 1. Достаточно доказать, что любая неисправность любого одного элемента схемы S обнаруживается хотя бы на одном наборе из множества T.

Рассмотрим произвольную неисправность произвольного одного элемента E в схеме S. При этой неисправности на выходе элемента E реализуется булева функция (от его входов), отличающаяся от исходной приписанной ему функции хотя бы на одном наборе $\tilde{\alpha}$; данный набор содержит две компоненты (одну компоненту), если элемент E — конъюнктор (соответственно инвертор). Возможны два подслучая:

1.1. Элемент E — конъюнктор E_q для некоторого $q \in \{1, \ldots, t_1 - 1\}$ (при этом $t_1 \geqslant 2$). Рассуждая по аналогии с разбором случая 1 в доказательстве леммы 2, получаем, что при последовательной подаче на входы схемы S наборов

$$(\tilde{0}^{j_{q+1}(1)}, \tilde{1}^{n+1-j_{q+1}(1)}), (\tilde{0}^{j_{q+1}(1)-1}, \tilde{1}^{n+2-j_{q+1}(1)}), (\tilde{1}^{j_{q+1}(1)-1}, 0, \tilde{1}^{n+1-j_{q+1}(1)}), (\tilde{1}^{n+1}) \in T$$

на входах элемента E возникают наборы $(0,0), (0,1), (1,0), (1,1)$ соответственно, а значение на выходе элемента E совпадает со значением на выходе цепочки Z. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$, а значение на выходе элемента E совпадает со значением на выходе цепочки Z. На этом наборе при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе цепочки Z, а значит, и на выходе схемы S (вне зависимости от того, содержится в ней инвертор I или нет), что и требовалось доказать.

1.2. Элемент E — инвертор I (и при этом $c = 1$). При последовательной подаче на входы схемы S наборов $(\tilde{0}^{n+1}), (\tilde{1}^{n+1}) \in T$ на входах элемента E (число которых равно 1), очевидно, возникают наборы $(0), (1)$ соответственно. Значит, существует такой набор $\tilde{\sigma} \in T$, при подаче которого на входы схемы S на входах элемента E возникает набор $\tilde{\alpha}$. Тогда на наборе $\tilde{\sigma}$ при переходе элемента E в рассматриваемое неисправное состояние меняется значение на выходе данного элемента, совпадающем с выходом схемы S, что и требовалось доказать.

2. Пусть $m \geqslant 2$. Утверждение 1 теоремы 1 сразу следует из леммы 2. Идея доказательства утверждения 2 теоремы состоит в подходящей перенумерации переменных функции $f(\tilde{x}^n)$, при которой набор $\tilde{\tau}$ совпадёт с одним из наборов вида $(\tilde{0}^r, \tilde{1}^{n+1-r})$, где $r \in \{0, \ldots, n\}$. Тогда в описании множества T встретятся два одинаковых набора и верхнюю оценку мощности этого множества, т. е. длины теста, можно будет понизить с $2n+4$ до $2n+3$.

Пусть набор $\tilde{\rho}$ для функции $f(\tilde{x}^n)$, удовлетворяющий условию леммы 1, равен $(\rho_1, \ldots, \rho_n)$ и имеет u нулевых и $n-u$ единичных компонент, причём в случае $u \geqslant 1$ номера его нулевых компонент равны $i_1, \ldots, i_u$, где $i_1 < \ldots < i_u$, а в случае $u \leqslant n-1$ номера его единичных компонент равны $i_{u+1}, \ldots, i_n$, где $i_{u+1} < \ldots < i_n$. Тогда $i_1, \ldots, i_n$ — попарно различные индексы от 1 до n, поэтому существуют такие индексы $i'_1, \ldots, i'_n$ от 1 до n, что $i_{i'_1} = 1, \ldots, i_{i'_n} = n$. Положим $f'(\tilde{x}^n) = f(x_{i'_1}, \ldots, x_{i'_n})$, $K'_1(\tilde{x}^n) = K_1(x_{i'_1}, \ldots, x_{i'_n}), \ldots, K'_m(\tilde{x}^n) = K_m(x_{i'_1}, \ldots, x_{i'_n})$. Пусть $\varphi(\tilde{x}^n)$ — произвольная

булева функция и $\varphi'(\tilde{x}^n) = \varphi(x_{i'_1}, \ldots, x_{i'_n})$. Имеем

$$\varphi'(x_{i_1}, \ldots, x_{i_n}) = \varphi(x_{i_{i'_1}}, \ldots, x_{i_{i'_n}}) = \varphi(\tilde{x}^n), \tag{5}$$

$$f'(\tilde{x}^n) = f(x_{i'_1}, \ldots, x_{i'_n}) = K_1(x_{i'_1}, \ldots, x_{i'_n}) \oplus \ldots \oplus K_m(x_{i'_1}, \ldots, x_{i'_n}) \oplus c = \\ = K'_1(\tilde{x}^n) \oplus \ldots \oplus K'_m(\tilde{x}^n) \oplus c,$$

т. е.

$$f'(\tilde{x}^n) = K'_1 \oplus \ldots \oplus K'_m \oplus c.$$

Последнее равенство задаёт представление булевой функции $f'(\tilde{x}^n)$ полиномом Жегалкина, причём K'_1 — самая короткая конъюнкция в этом полиноме, поскольку ранги конъюнкций $K'_1, \ldots, K'_m$ совпадают с рангами конъюнкций $K_1, \ldots, K_m$ соответственно. Пусть $h'(\tilde{x}^n) = K'_2 \oplus \ldots \oplus K'_m$,

$$a' = \begin{cases} 1, & \text{если } K'_1 \leqslant h' \text{ или } K'_1 \leqslant h' \oplus 1, \\ 0 & \text{иначе.} \end{cases}$$

Тогда

$$h'(\tilde{x}^n) = K_2(x_{i'_1}, \ldots, x_{i'_n}) \oplus \ldots \oplus K_m(x_{i'_1}, \ldots, x_{i'_n}) = h(x_{i'_1}, \ldots, x_{i'_n}).$$

Отсюда и из определения функции $K_1(\tilde{x}^n)$ следует, что неравенство $K'_1 \leqslant h'$ равносильно неравенству $K_1 \leqslant h$, а неравенство $K'_1 \leqslant h' \oplus 1$ — неравенству $K_1 \leqslant h \oplus 1$. Таким образом, $a' = a$. Далее, пусть $\tilde{\sigma}_{K'_1}$ — двоичный $(n+1)$-разрядный набор, содержащий единицы только в тех компонентах, которые отвечают переменным, входящим в конъюнкцию K'_1, а $\tilde{\rho}' = (\tilde{0}^u, \tilde{1}^{n-u})$. В силу определения набора $\tilde{\rho}$ в случае $a = 1$ выполнены равенства $K_1(\tilde{\rho}) = 0$ и $h(\tilde{\rho}) = 1$, а в случае $a = 0$ — равенства $K_1(\tilde{\rho}) = 1$ и $h(\tilde{\rho}) = \overline{h(\tilde{1}^n)}$. Тогда в случае $a' = 1$, т. е. $a = 1$, в силу (5) выполнены равенства

$$0 = K_1(\tilde{\rho}) = K'_1(\rho_{i_1}, \ldots, \rho_{i_u}, \rho_{i_{u+1}}, \ldots, \rho_{i_n}) = K'_1(\tilde{0}^u, \tilde{1}^{n-u}) = K'_1(\tilde{\rho}')$$

и, аналогично, $1 = h(\tilde{\rho}) = h'(\tilde{\rho}')$, а в случае $a' = 0$, т. е. $a = 0$ — равенства $1 = K_1(\tilde{\rho}) = K'_1(\tilde{\rho}')$, $h(\tilde{\rho}) = h'(\tilde{\rho}')$ и $h(\tilde{1}^n) = \overline{h'(\tilde{1}^n)}$ (набор $(\tilde{1}^n)$ не меняется при перестановке его компонент), откуда $h'(\tilde{\rho}') = \overline{h'(\tilde{1}^n)}$. Обозначим через $\tilde{\tau}'$ двоичный $(n+1)$-разрядный набор, получающийся из набора $\tilde{\rho}'$ добавлением справа $(n+1)$-й компоненты, равной единице. Тогда $\tilde{\tau}' = (\tilde{0}^u, \tilde{1}^{n-u+1})$. В силу леммы 2 (в исходных обозначениях которой к $f, K_1, \ldots, K_m, h, a, \tilde{\rho}, \tilde{\tau}$ добавлены штрихи) функцию $f'(\tilde{x}^n)$ можно реализовать неизбыточной схемой S' в базисе B, содержащей одну фиктивную входную переменную и допускающей ЕПТ $T' = \{(\tilde{0}^r, \tilde{1}^{n+1-r}), (\tilde{1}^s, 0, \tilde{1}^{n-s}), \tilde{\sigma}_{K'_1}, \tilde{\tau}' : r \in \{0, \ldots, n+1\}, s \in \in \{1, \ldots, n\}\}$. Заметим, что $\tilde{\tau}' = (\tilde{0}^r, \tilde{1}^{n+1-r})$ при $r = u$, поэтому мощность множества T' не превосходит $2n + 3$. Схема S', согласно определению, реализует булеву функцию $f'^{(+1)}(\tilde{x}^{n+1})$, не зависящую существенно от переменной x_{n+1} и равную функции $f'(\tilde{x}^n)$, т. е. функции $f(x_{i'_1}, \ldots, x_{i'_n})$. Тогда $f'^{(+1)}(\tilde{x}^{n+1}) = f^{(+1)}(x_{i'_1}, \ldots, x_{i'_n}, x_{n+1})$. Нетрудно показать, что если на входы схемы S' вместо переменных $x_{i'_1}, \ldots, x_{i'_n}, x_{n+1}$ подать переменные $x_1, \ldots, x_n, x_{n+1}$ соответственно, то полученная СФЭ в базисе B реализует функцию $f^{(+1)}(\tilde{x}^{n+1})$ (а значит, содержит одну фиктивную входную переменную и реализует функцию $f(\tilde{x}^n)$), неизбыточна и допускает ЕПТ длины не более $2n + 3$, получающийся из множества T' перестановкой компонент каждого набора, содержащегося в T', при которой i'_1-я, …, i'_n-я, $(n+1)$-я компоненты становятся 1-й, …, n-й, $(n+1)$-й компонентами. Утверждение 2 теоремы 1 в случае 2, а вместе с ним и вся теорема 1 доказаны. ■

Заключение

В работе предложен метод реализации любой неконстантной булевой функции $f(\tilde{x}^n)$ схемой из функциональных элементов в базисе $\{\&, \oplus, \neg\}$, содержащей одну фиктивную входную переменную, неизбыточной и допускающей единичный проверяющий тест длины не более $2n+3$ относительно произвольных неисправностей элементов, существенно более короткий, чем тривиальный тест из 2^{n+1} наборов. Данный метод может быть использован на практике для построения легкотестируемых интегральных схем в случае, когда тип допустимых неисправностей содержащихся в них элементов никак не ограничивается (в частности, не ограничивается хорошо изученными константными либо инверсными неисправностями на входах и/или выходах элементов).

Следующим шагом, по мнению автора, должно стать получение линейной по n верхней оценки длины единичного проверяющего теста для неизбыточных схем из функциональных элементов в том же базисе, реализующих произвольные неконстантные булевы функции от n переменных и не содержащих фиктивных переменных (в предположении, что рассматриваются произвольные неисправности элементов). У автора имеется задел в этом направлении.

ЛИТЕРАТУРА

1. *Чегис И. А., Яблонский С. В.* Логические способы контроля работы электрических схем // Труды МИАН. 1958. Т. 51. С. 270–360.
2. *Яблонский С. В.* Надёжность и контроль управляющих систем // Материалы Всесоюзного семинара по дискретной математике и её приложениям (Москва, 31 января–2 февраля 1984 г.). М.: Изд-во МГУ, 1986. С. 7–12.
3. *Яблонский С. В.* Некоторые вопросы надёжности и контроля управляющих систем // Математические вопросы кибернетики. Вып. 1. М.: Наука, 1988. С. 5–25.
4. *Редькин Н. П.* Надёжность и диагностика схем. М.: Изд-во МГУ, 1992. 192 с.
5. *Бородина Ю. В.* О схемах, допускающих единичные тесты длины 1 при константных неисправностях на выходах элементов // Вестник Московского университета. Сер. 1. Математика. Механика. 2008. № 5. С. 49–52.
6. *Попков К. А.* Нижние оценки длин единичных тестов для схем из функциональных элементов // Дискретная математика. 2017. Т. 29. Вып. 2. С. 53–69.
7. *Попков К. А.* Единичные проверяющие тесты для схем из функциональных элементов в базисе «конъюнкция-отрицание» // Прикладная дискретная математика. 2017. № 38. С. 66–88.
8. *Reddy S. M.* Easily testable realizations for logic functions // IEEE Trans. Comput. 1972. V. C-21. Iss. 11. P. 1183–1188.
9. *Коляда С. С.* Единичные проверяющие тесты для схем из функциональных элементов // Вестник Московского университета. Сер. 1. Математика. Механика. 2013. № 4. С. 32–34.
10. *Коляда С. С.* Верхние оценки длины проверяющих тестов для схем из функциональных элементов: дис. ... канд. физ.-мат. наук. М., 2013. 77 с.
11. *Романов Д. С.* Метод синтеза легкотестируемых схем, допускающих единичные проверяющие тесты константной длины // Дискретная математика. 2014. Т. 26. Вып. 2. С. 100–130.
12. *Романов Д. С.* О тестах для схем при неисправностях на выходах элементов // Материалы XVIII Междунар. конф. «Проблемы теоретической кибернетики» (Пенза, 19–23 июня 2017 г.). М.: МАКС Пресс, 2017. С. 213–216.

13. *Попков К. А.* Короткие единичные тесты для схем при произвольных константных неисправностях на выходах элементов // Дискретная математика. 2018. Т. 30. Вып. 3. С. 99–116.
14. *Коваценко С. В.* Синтез легкотестируемых схем в базисе Жегалкина для инверсных неисправностей // Вестник Московского университета. Сер. 15. Вычислительная математика и кибернетика. 2000. № 2. С. 45–47.
15. *Редькин Н. П.* О единичных проверяющих тестах схем при инверсных неисправностях элементов // XII Междунар. конф. по проблемам теоретической кибернетики (Н. Новгород, 1999). Тез. докл. М.: Изд-во механико-математического факультета МГУ, 1999. С. 196.
16. *Редькин Н. П.* Единичные проверяющие тесты для схем при инверсных неисправностях элементов // Математические вопросы кибернетики. Вып. 12. М.: Физматлит, 2003. С. 217–230.
17. *Попков К. А.* Синтез легкотестируемых схем при однотипных константных неисправностях на входах и выходах элементов // Интеллектуальные системы. Теория и приложения. 2018. Т. 23. Вып. 3. С. 131–147.
18. *Романов Д. С., Романова Е. Ю.* Короткие тесты для схем в базисе Жегалкина // Интеллектуальные системы. Теория и приложения. 2016. Т. 20. Вып. 3. С. 73–78.
19. *Романов Д. С., Романова Е. Ю.* Метод синтеза неизбыточных схем, допускающих единичные проверяющие тесты константной длины // Дискретная математика. 2017. Т. 29. Вып. 4. С. 87–105.
20. *Попков К. А.* Синтез легкотестируемых схем при произвольных константных неисправностях на входах и выходах элементов // Прикладная дискретная математика. 2019. № 43. С. 78–100.
21. *Попков К. А.* Метод построения легко диагностируемых схем из функциональных элементов относительно единичных неисправностей // Прикладная дискретная математика. 2019. № 46. С. 38–57.
22. *Попков К. А.* Короткие полные проверяющие тесты для схем из двухвходовых функциональных элементов // Дискретный анализ и исследование операций. 2019. Т. 26. № 1. С. 89–113.
23. *Яблонский С. В.* Введение в дискретную математику. М.: Наука, 1986. 384 с.

REFERENCES

1. *Chegis I. A. and Yablonskiy S. V.* Logicheskie sposoby kontrolya raboty elektricheskikh skhem [Logical methods of control of work of electric circuits]. Trudy Mat. Inst. Steklov, 1958, vol. 51, pp. 270–360. (in Russian)
2. *Yablonskiy S. V.* Nadezhnost' i kontrol' upravlyayushchikh sistem [Reliability and verification of control systems]. Materialy Vsesoyuznogo seminara po diskretnoy matematike i ee prilozheniyam (Moscow, 31 Jan.–2 Feb. 1984). Moscow., MSU Publ., 1986, pp. 7–12. (in Russian)
3. *Yablonskiy S. V.* Nekotorye voprosy nadezhnosti i kontrolya upravlyayushchikh sistem [Some questions of reliability and verification of control systems]. Matematicheskie Voprosy Kibernetiki, iss. 1, Moscow, Nauka Publ., 1988, pp. 5–25. (in Russian)
4. *Red'kin N. P.* Nadezhnost' i diagnostika skhem [Circuits Reliability and Diagnostics]. Moscow, MSU Publ., 1992. 192 p. (in Russian)
5. *Borodina Yu. V.* Circuits admitting single-fault tests of length 1 under constant faults at outputs of elements. Mosc. Univ. Math. Bull., 2008, vol. 63, iss. 5, pp. 202–204.
6. *Popkov K. A.* Lower bounds for lengths of single tests for Boolean circuits. Discrete Math. Appl., 2019, vol. 29, iss. 1, pp. 23–33.

7. *Popkov K. A.* Edinichnye proveryayushchie testy dlya skhem iz funktsional'nykh elementov v bazise "kon'yunktsiya-otritsanie" [Single fault detection tests for logic networks in the basis "conjunction-negation"]. Prikladnaya Diskretnaya Matematika, 2017, no. 38, pp. 66–88. (in Russian)
8. *Reddy S. M.* Easily testable realizations for logic functions. IEEE Trans. Comput., 1972, vol. C-21, iss. 11, pp. 1183–1188.
9. *Kolyada S. S.* Single fault detection tests for circuits of functional elements. Mosc. Univ. Math. Bull., 2013, vol. 68, iss. 4, pp. 192–193.
10. *Kolyada S. S.* Verkhnie otsenki dliny proveryayushchikh testov dlya skhem iz funktsional'nykh elementov [Upper bounds on length of fault detection tests for logic networks]. Cand. Diss., Moscow, 2013. 77 p. (in Russian)
11. *Romanov D. S.* Method of synthesis of easily testable circuits admitting single fault detection tests of constant length. Discrete Math. Appl., 2014, vol. 24, iss. 4, pp. 227–251.
12. *Romanov D. S.* O testakh dlya skhem pri neispravnostyakh na vykhodakh elementov [On tests for logic networks under faults at outputs of gates]. Materialy XVIII Mezhdunarodnoy konferentsii "Problemy teoreticheskoy kibernetiki" (Penza, 19–23 June 2017). Moscow, MAKS Press Publ., 2017, pp. 213–216. (in Russian)
13. *Popkov K. A.* Short single tests for circuits with arbitrary stuck-at faults at outputs of gates. Discrete Math. Appl., 2019, vol. 29, iss. 5, pp. 321–333.
14. *Kovatsenko S. V.* Sintez legkotestiruemykh skhem v bazise Zhegalkina dlya inversnykh neispravnostey [Synthesis of easily testable logic networks in the Zhegalkin basis for inverse faults]. Vestnik MSU, Ser. 15, 2000, no. 2, pp. 45–47. (in Russian)
15. *Red'kin N. P.* O edinichnykh proveryayushchikh testakh skhem pri inversnykh neispravnostyakh elementov [On single fault detection tests of logic networks under inverse faults of gates]. XII Mezhdunarodnaya konferentsiya po problemam teoreticheskoy kibernetiki (Nizhny Novgorod, 1999). Tezisy dokladov, Moscow, Mech.-Math. Faculty of MSU Publ., 1999, p. 196. (in Russian)
16. *Red'kin N. P.* Edinichnye proveryayushchie testy dlya skhem pri inversnykh neispravnostyakh elementov [Single fault detection tests for logic networks under inverse faults of gates]. Matematicheskie Voprosy Kibernetiki, iss. 12, Moscow, Fizmatlit Publ., 2003, pp. 217–230. (in Russian)
17. *Popkov K. A.* Sintez legkotestiruemykh skhem pri odnotipnykh konstantnykh neispravnostyakh na vkhodakh i vykhodakh elementov [Synthesis of easily testable logic networks under one-type stuck-at faults at inputs and outputs of gates]. Intellektual'nye Sistemy. Teoriya i Prilozheniya, 2018, vol. 23, iss. 3, pp. 131–147. (in Russian)
18. *Romanov D. S., Romanova E. Yu.* Korotkie testy dlya skhem v bazise Zhegalkina [Short tests for circuits in the Zhegalkin basis]. Intellektual'nye Sistemy. Teoriya i Prilozheniya, 2016, vol. 20, iss. 3, pp. 73–78. (in Russian)
19. *Romanov D. S., Romanova E. Yu.* A method of synthesis of irredundant circuits admitting single fault detection tests of constant length. Discrete Math. Appl., 2019, vol. 29, iss. 1, pp. 35–48.
20. *Popkov K. A.* Sintez legkotestiruemykh skhem pri proizvol'nykh konstantnykh neispravnostyakh na vkhodakh i vykhodakh elementov [Synthesis of easily testable logic networks under arbitrary stuck-at faults at inputs and outputs of gates]. Prikladnaya Diskretnaya Matematika, 2019, no. 43, pp. 78–100. (in Russian)
21. *Popkov K. A.* Metod postroeniya legko diagnostiruemykh skhem iz funktsional'nykh elementov otnositel'no edinichnykh neispravnostey [A method of constructing of easily diagnosable logic networks regarding single faults]. Prikladnaya Diskretnaya Matematika, 2019, no. 46, pp. 38–57. (in Russian)

22. *Popkov K. A.* Short complete fault detection tests for logic networks with fan-in two. Journ. Appl. Industr. Math., 2019, vol. 13, iss. 1, pp. 118–131.
23. *Yablonskiy S. V.* Vvedenie v diskretnuyu matematiku [Introduction to Discrete Mathematics]. Moscow, Nauka Publ., 1986. 384 p. (in Russian)

ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ

УДК 519.725

ПИРАМИДАЛЬНАЯ СХЕМА ПОСТРОЕНИЯ БИОРТОГОНАЛЬНЫХ ВЕЙВЛЕТ-КОДОВ НАД КОНЕЧНЫМИ ПОЛЯМИ

Д. В. Литичевский

Челябинский государственный университет, г. Челябинск, Россия

Конструктивным образом доказывается существование биортогонального разбиения векторного пространства V размерности n над полем $\mathrm{GF}(q)$, а именно двух его представлений в виде прямых сумм подпространств $V = W_0 \oplus W_1 \oplus \ldots \oplus W_J \oplus V_J$ и $V = \tilde{W}_0 \oplus \tilde{W}_1 \oplus \ldots \oplus \tilde{W}_J \oplus \tilde{V}_J$, таких, что на j-м уровне разложения $(0 < j \leqslant J)$ $V_{j-1} = V_j \oplus W_j$, $\tilde{V}_{j-1} = \tilde{V}_j \oplus \tilde{W}_j$, подпространство V_j ортогонально $\tilde{W}_j$, а подпространство W_j ортогонально $\tilde{V}_j$. Для этого используются пары биортогональных фильтров (h, g) и $(\tilde{h}, \tilde{g})$. Разбиение пространства на j-м уровне разложения осуществляется при помощи пар уровневых фильтров (h^j, g^j) и $(\tilde{h}^j, \tilde{g}^j)$, для построения которых разработаны и теоретически обоснованы соответствующие алгоритмы. На основе многоуровневой схемы вейвлет-разложения строится новое семейство биортогональных вейвлет-кодов со скоростью кодирования 2^{-L}, где L — количество использованных уровней разложения, и приводятся примеры таких кодов.

Ключевые слова: *дискретные биортогональные вейвлет-преобразования, мультиразрешения, вейвлет-коды.*

DOI 10.17223/20710410/51/5

PYRAMID SCHEME FOR CONSTRUCTING BIORTHOGONAL WAVELET CODES OVER FINITE FIELDS

D. V. Litichevskiy

Chelyabinsk state university, Chelyabinsk, Russia

E-mail: litichevskiydv@gmail.com

The existence of a biorthogonal decomposition of the space V of dimension n over the field $\mathrm{GF}(q)$ is constructively proved, namely, two representations of it are obtained as direct sums of subspaces $V = W_0 \oplus W_1 \oplus \ldots \oplus W_J \oplus V_J$ and $V = \tilde{W}_0 \oplus \tilde{W}_1 \oplus \ldots \oplus \tilde{W}_J \oplus \tilde{V}_J$, such that at the j-th level of the decomposition, for $0 < j \leqslant J$, $V_{j-1} = V_j \oplus W_j$, $\tilde{V}_{j-1} = \tilde{V}_j \oplus \tilde{W}_j$, the subspace V_j is orthogonal to $\tilde{W}_j$, and the subspace W_j is orthogonal to $\tilde{V}_j$. The partition of the space at the j-th level is made with the help of pairs of level filters (h^j, g^j) and $(\tilde{h}^j, \tilde{g}^j)$, for the construction of which the corresponding algorithms have been developed and theoretically proved. A new family of biorthogonal wavelet codes is built on the basis of the multilevel wavelet decomposition scheme with coding rate 2^{-L}, where L is the number of used decomposition levels, and examples of such codes are given.

Keywords: *discrete biorthogonal wavelet transforms, multiresolutions, wavelet codes.*

Введение

В то время как теории непрерывных и дискретных вейвлет-преобразований над полями вещественных и комплексных чисел интенсивно разрабатываются в течение многих лет [1, 2] и широко применяются для анализа сигналов различной природы во многих областях науки, теории вейвлет-преобразований над конечными полями уделяется не слишком много внимания ввиду ограниченности возможностей её применения. Первое видение теории ортогональных вейвлет-преобразований над конечными полями представлено в [3], однако случай конечных полей характеристики два в данной работе считается исключительным и потому не рассмотрен. Дальнейшее развитие представленная теория получила в [4], где учитывается также и случай полей характеристики два. Однако, в отличие от [3], авторы [4] не строят описанную в [1] цепочку вложенных подпространств и ограничиваются лишь одним уровнем разложения.

Первые прикладные результаты использования построенной теории ортогональных вейвлет-преобразований над конечными полями представлены в [5, 6], где описан подход к построению кодов, корректирующих ошибки и названных ортогональными вейвлет-кодами. Этот подход базируется на одном уровне вейвлет-разложения и имеет ряд конструктивных особенностей, ограничивших впоследствии практическое применение ортогональных вейвлет-кодов. Кроме того, построение ортогональных вейвлет-преобразований над конечными полями является вычислительно сложной задачей, эффективное решение для её упрощенной версии было в дальнейшем представлено в [7].

Помимо ортогональных вейвлет-преобразований, в теории цифровой обработки сигналов существует более гибкая конструкция биортогональных вейвлет-преобразований, адаптация которой для случая конечных полей произведена в [8]. Термин «биортогональное вейвлет-преобразование» введён в [1], хотя и в [8], и в данной работе предлагается иная конструкция биортогональных вейвлетов. Отметим, что, так же как в [4], автор в [8] не строит цепочку вложенных подпространств, ограничиваясь одним уровнем вейвлет-разложения. Полученные результаты, тем не менее, позволили расширить класс вейвлет-кодов биортогональными вейвлет-кодами, схема помехоустойчивого кодирования которых изображена на рис. 1. Данная схема описывает код с длиной кодовых и информационных слов n и $n/2$, определённый над полем $\mathrm{GF}(q)$, $q = p^m$, где p — простое число, m — натуральное. Основная идея, лежащая в её основе, заключается в том, что кодовое слово c восстанавливается по векторам вейвлет-коэффициентов a и d:

$$c = H^{\mathrm{T}} a + G^{\mathrm{T}} d,$$

которые, в свою очередь, получаются из информационного слова v при помощи соотношений

$$a = v, \quad d = \lambda_1 \Pi v.$$

Матрицы H, G, $\tilde{H}$, $\tilde{G}$ являются 2-циркулянтными матрицами размера $n/2 \times n$, определяемыми как

$$H = \mathrm{cir}_2(h_0, h_1, \ldots, h_{n-1}), \quad G = \mathrm{cir}_2(g_0, g_1, \ldots, g_{n-1}),$$
$$\tilde{H} = \mathrm{cir}_2(\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_{n-1}), \quad \tilde{G} = \mathrm{cir}_2(\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{n-1}).$$

2-Циркулянтная матрица задаётся первой строкой, последующая строка получается из предыдущей циклическим сдвигом на две позиции вправо. Последовательности элементов поля $\mathrm{GF}(q)$ длины n, $h = (h_0, h_1, \ldots, h_{n-1})$, $g = (g_0, g_1, \ldots, g_{n-1})$ и $\tilde{h} = (\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_{n-1})$, $\tilde{g} = (\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{n-1})$, далее именуемые фильтрами синтеза и

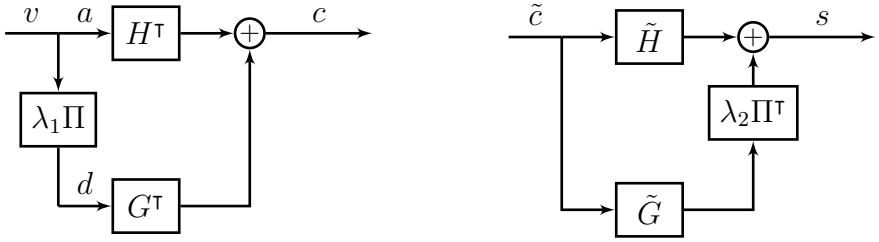
анализа соответственно, задают две масштабирующих последовательности и два вейвлета биортогонального вейвлет-преобразования. Матрица Π — циркулянтная матрица размера $n/2 \times n/2$, первая строка которой имеет вид $(0, 0, 0, \ldots, 1)$. Условие биортогональности, которому должны удовлетворять фильтры анализа и синтеза, записанное с использованием введённых матриц, имеет вид

$$\begin{aligned} \tilde{H}H^{\mathrm{T}} = I_{n/2}, \;\; \tilde{G}G^{\mathrm{T}} = I_{n/2}, \\ \tilde{H}G^{\mathrm{T}} = O_{n/2}, \;\; \tilde{G}H^{\mathrm{T}} = O_{n/2}. \end{aligned} \tag{1}$$

Условие точного восстановления записывается в виде

$$\begin{bmatrix} H^{\mathrm{T}} & G^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \tilde{H} \\ \tilde{G} \end{bmatrix} = I_n. \tag{2}$$

Символами I и O обозначены единичная и нулевая матрицы соответственно, нижний индекс указывает их размерность. В [8] доказано, что фильтры анализа и синтеза удовлетворяют условию биортогональности (1) тогда и только тогда, когда они удовлетворяют условию точного восстановления (2).



Кодирование сообщения Вычисление синдрома

Рис. 1. Схема помехоустойчивого кодирования биортогональных вейвлет-кодов

Биортогональные вейвлет-коды, определяемые схемой помехоустойчивого кодирования на рис. 1, являются линейными кодами, порождающая и проверочная матрицы которых равны

$$H^{\mathrm{T}} + \lambda_1 G^{\mathrm{T}}\Pi \;\; \text{и} \;\; \tilde{H} + \lambda_2 \Pi^{\mathrm{T}} \tilde{G}$$

соответственно, параметры λ_1 и λ_2 — элементы поля $\mathrm{GF}(q)$, такие, что

$$\lambda_1 \lambda_2 = (p-1) \bmod p.$$

Кодовое слово c получается из информационного слова v при помощи соотношения

$$c = \left(H^{\mathrm{T}} + \lambda_1 G^{\mathrm{T}}\Pi\right) v,$$

синдром s принятого зашумлённого кодового слова $\tilde{c}$ находится как

$$s = \left(\tilde{H} + \lambda_2 \Pi^{\mathrm{T}} \tilde{G}\right) \tilde{c}.$$

Согласно [9], описанные биортогональные вейвлет-коды относятся к подклассу квазициклических кодов с циклическим сдвигом кодовых слов на две позиции.

Дальнейшие исследования биортогональных вейвлет-кодов были связаны с изучением их отдельных подклассов, а именно представленных в [10 – 12] кодов с максимально возможным и заданным кодовым расстоянием, в которых также используется один уровень вейвлет-разложения. Целью данной работы является углубление теории биортогональных вейвлет-преобразований над конечными полями, а именно построение описанной в [1] цепочки вложенных подпространств и создание биортогональных вейвлет-кодов, использующих несколько уровней вейвлет-разложения.

1. Биортогональные вейвлет-преобразования над конечными полями

Через $V = \big(\mathrm{GF}(q)\big)^n$ обозначим векторное пространство размерности n над полем $\mathrm{GF}(q)$. Тогда, согласно результатам [8], векторное пространство V представимо в виде прямой суммы подпространств

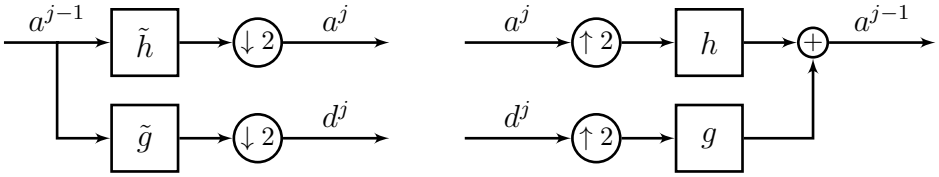
$$V = \tilde{V}_0 \oplus \tilde{W}_0 = V_0 \oplus W_0, \tag{3}$$

при этом базисы подпространств V_0, W_0 и $\tilde{V}_0$, $\tilde{W}_0$ образуют строки ранее упомянутых 2-циркулянтных матриц H, G и $\tilde{H}$, $\tilde{G}$ соответственно, удовлетворяющих условию биортогональности (1). Согласно [1], для построения цепочки вложенных подпространств при использовании биортогональных вейвлет-преобразований для каждого допустимого значения j необходимо уметь строить представления

$$V_{j-1} = V_j \oplus W_j \quad \text{и} \quad \tilde{V}_{j-1} = \tilde{V}_j \oplus \tilde{W}_j,$$

такие, что подпространство W_j ортогонально $\tilde{V}_j$, тогда как подпространство $\tilde{W}_j$ ортогонально V_j.

В классической схеме биортогонального вейвлет-преобразования (рис. 2) векторы вейвлет-коэффициентов a^j и d^j вычисляются по a^{j-1} с помощью пары фильтров анализа $\tilde{h}$ и $\tilde{g}$ с последующей двоичной неполной выборкой (операция обозначена символом «$\downarrow$»), вектор вейвлет-коэффициентов a^{j-1} восстанавливается при помощи пары фильтров синтеза h и g по векторам a^j и d^j, между компонентами которых подставлены нули (операция обозначена символом «$\uparrow$»). Исходный сигнал имеет бесконечную длину, а его компоненты являются вещественными либо комплексными числами, поэтому для построения указанных представлений используются пары фильтров синтеза и анализа h, g и $\tilde{h}$, $\tilde{g}$ соответственно. Однако в рассматриваемом случае исходный сигнал имеет конечную длину и состоит из элементов поля $\mathrm{GF}(q)$, поэтому на каждом уровне разложения последовательности вейвлет-коэффициентов имеют половинную длину. Это означает, что на j-м уровне разложения необходимо использовать свои пары фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ длины $n_j = n/2^j$, удовлетворяющие условию биортогональности, $h^0 = h$, $g^0 = g$ и $\tilde{h}^0 = \tilde{h}$, $\tilde{g}^0 = \tilde{g}$.



Вычисление a^j и d^j по a^{j-1} Восстановление a^{j-1} по a^j и d^j

Рис. 2. Схема быстрого биортогонального вейвлет-преобразования

Для того чтобы описать процедуры вычисления вейвлет-коэффициентов и восстановления сигнала на j-м уровне вейвлет-разложения, парам фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ поставим в соответствие 2-циркулянтные матрицы H_j, G_j и $\tilde{H}_j$, $\tilde{G}_j$ размера $n_j/2 \times n_j$, являющиеся аналогами матриц H, G и $\tilde{H}$, $\tilde{G}$ для j-го уровня разложения. Тогда векторы вейвлет-коэффициентов a^j и d^j вычисляются по a^{j-1} при помощи соотношений

$$a^j = \tilde{H}_j a^{j-1}, \quad d^j = \tilde{G}_j a^{j-1}, \tag{4}$$

вектор вейвлет-коэффициентов a^{j-1} восстанавливается по a^j и d^j по формуле

$$a^{j-1} = H_j^{\mathrm{T}} a^j + G_j^{\mathrm{T}} d^j. \tag{5}$$

Записанное с использованием введённых матриц условие биортогональности для пар фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ имеет вид

$$\begin{aligned} &\tilde{H}_j H_j^{\mathrm{T}} = I_{n_j/2}, \;\; \tilde{G}_j G_j^{\mathrm{T}} = I_{n_j/2}, \\ &\tilde{H}_j G_j^{\mathrm{T}} = O_{n_j/2}, \;\; \tilde{G}_j H_j^{\mathrm{T}} = O_{n_j/2}. \end{aligned} \tag{6}$$

Условие точного восстановления для j-го уровня разложения записывается как

$$\begin{bmatrix} H_j^{\mathrm{T}} & G_j^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \tilde{H}_j \\ \tilde{G}_j \end{bmatrix} = I_{n_j}. \tag{7}$$

Воспользовавшись результатами работы [8], получаем, что фильтры анализа и синтеза для j-го уровня вейвлет-разложения удовлетворяют условию биортогональности (6) тогда и только тогда, когда они удовлетворяют условию точного восстановления (7). Стоит также отметить, что из условия (7) следует обратимость матриц

$$\begin{bmatrix} H_j^{\mathrm{T}} & G_j^{\mathrm{T}} \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} \tilde{H}_j \\ \tilde{G}_j \end{bmatrix},$$

из чего, в свою очередь, следует линейная независимость их столбцов и строк соответственно, а также то, что ранг матриц H_j, G_j и $\tilde{H}_j$, $\tilde{G}_j$ равен $n_j/2$.

Лемма 1. При помощи пар фильтров анализа и синтеза для уровней вейвлет-разложения с 0 по j, удовлетворяющих условию биортогональности (6), могут быть построены представления подпространств V_{j-1} и $\tilde{V}_{j-1}$ размерности n_j в виде прямой суммы подпространств V_j, W_j и $\tilde{V}_j$, $\tilde{W}_j$ соответственно размерности $n_j/2$, таких, что подпространство W_j ортогонально $\tilde{V}_j$, подпространство $\tilde{W}_j$ ортогонально V_j, кроме того, $V_j \subset V_{j-1}$ и $\tilde{V}_j \subset \tilde{V}_{j-1}$.

Доказательство. Из (5) получаем, что вектор $v \in V$ представим в виде

$$v = H_0^{\mathrm{T}} a^0 + G_0^{\mathrm{T}} d^0 = v^0 + w^0,$$

где $v^0 \in V_0$, $w^0 \in W_0$. Ещё раз воспользовавшись соотношением (5), заменим в этом выражении вектор a^0 на его представление через векторы a^1 и d^1:

$$\begin{gathered} v = H_0^{\mathrm{T}} a^0 + G_0^{\mathrm{T}} d^0 = H_0^{\mathrm{T}} (H_1^{\mathrm{T}} a^1 + G_1^{\mathrm{T}} d^1) + G_0^{\mathrm{T}} d^0 = H_0^{\mathrm{T}} \cdot H_1^{\mathrm{T}} a^1 + H_0^{\mathrm{T}} \cdot G_1^{\mathrm{T}} d^1 + G_0^{\mathrm{T}} d^0 = \\ = v^1 + w^1 + w^0, \end{gathered}$$

где $v^1 \in V_1$, $w^1 \in W_1$. Применив аналогичную процедуру к векторам вейвлет-коэффициентов $a^1, \ldots, a^{j-1}$, получим соотношение

$$v = H_0^{\mathrm{T}} \cdot \ldots \cdot H_j^{\mathrm{T}} a^j + H_0^{\mathrm{T}} \cdot \ldots \cdot H_{j-1}^{\mathrm{T}} \cdot G_j^{\mathrm{T}} d^j + \ldots + G_0^{\mathrm{T}} d^0 = v^j + w^j + \ldots + v^0,$$

где $v^j \in V_j$, $w^j \in W_j$. Примем, что базисами V_j и W_j являются строки матриц $H_j \cdot \ldots \cdot H_0$ и $G_j \cdot H_{j-1} \cdot \ldots \cdot H_0$ соответственно размера $n_j/2 \times n$.

Из соотношений (4) для векторов вейвлет-коэффициентов $a^j, \ldots, a^0$ и $d^j, \ldots, d^0$ получаем, что

$$a^j = \tilde{H}_j \cdot \ldots \cdot \tilde{H}_0 v, \quad d^j = \tilde{G}_j \cdot \tilde{H}_{j-1} \cdot \ldots \cdot \tilde{H}_0 v.$$

Примем, что базисами $\tilde{V}_j$ и $\tilde{W}_j$ являются строки матриц $\tilde{H}_j \cdot \ldots \cdot \tilde{H}_0$ и $\tilde{G}_j \cdot \tilde{H}_{j-1} \cdot \ldots \cdot \tilde{H}_0$ соответственно размера $n_j/2 \times n$.

Ранги матриц $H_j, H_{j-1} \ldots, H_0$ равны $n_j/2$, n_j, $\ldots$, n соответственно, все строки каждой из них линейно независимы, следовательно, все строки матрицы $H_j \cdot \ldots \cdot H_0$ также линейно независимы, а это значит, что её строки могут являться базисом подпространства V_j. Схожие утверждения для матриц $G_j \cdot H_{j-1} \cdot \ldots \cdot H_0$, $\tilde{H}_j \cdot \ldots \cdot \tilde{H}_0$ и $\tilde{G}_j \cdot \tilde{H}_{j-1} \cdot \ldots \cdot \tilde{H}_0$ доказываются аналогичным образом.

В силу того, что столбцы матрицы $\begin{bmatrix} H_j^{\mathrm{T}} & G_j^{\mathrm{T}} \end{bmatrix}$ линейно независимы, строки матрицы

$$\begin{bmatrix} H_j \\ G_j \end{bmatrix} \cdot H_{j-1} \cdot \ldots \cdot H_0$$

также линейно независимы. Поскольку базисами подпространств V_j и W_j являются строки матриц $H_j \cdot \ldots \cdot H_0$ и $G_j \cdot H_{j-1} \cdot \ldots \cdot H_0$ соответственно, их пересечение — нулевое подпространство, что означает, что $V_{j-1} = V_j \oplus W_j$. Поскольку базисные векторы подпространства V_j суть невырожденные линейные комбинации базисных векторов подпространства V_{j-1}, то $V_j \subset V_{j-1}$. Схожие утверждения для подпространств $\tilde{V}_{j-1}$, $\tilde{V}_j$ и $\tilde{W}_j$ доказываются аналогично.

Без ограничения общности докажем ортогональность подпространств V_j и $\tilde{W}_j$, ортогональность подпространств $\tilde{V}_j$ и W_j доказывается аналогично. Рассмотрим скалярное произведение произвольных векторов $v^j \in V_j$, $\tilde{w}^j \in \tilde{W}_j$ и покажем, что оно равно нулю. Согласно выбору базиса подпространств V_j и $\tilde{W}_j$, векторы v^j и $\tilde{w}^j$ представимы в виде

$$v^j = H_0^{\mathrm{T}} \cdot \ldots \cdot H_j^{\mathrm{T}} x, \quad \tilde{w}^j = \tilde{H}_0^{\mathrm{T}} \cdot \ldots \cdot \tilde{H}_{j-1}^{\mathrm{T}} \cdot \tilde{G}_j^{\mathrm{T}} y,$$

где x и y — векторы длины $n_j/2$, состоящие из элементов поля $\mathrm{GF}(q)$. Тогда

$$\langle \tilde{w}^j, v^j \rangle = \left\langle \tilde{H}_0^{\mathrm{T}} \cdot \ldots \cdot \tilde{H}_{j-1}^{\mathrm{T}} \cdot \tilde{G}_j^{\mathrm{T}} y, H_0^{\mathrm{T}} \cdot \ldots \cdot H_j^{\mathrm{T}} x \right\rangle = y^{\mathrm{T}} \tilde{G}_j \cdot \tilde{H}_{j-1} \cdot \ldots \cdot \tilde{H}_0 \cdot H_0^{\mathrm{T}} \cdot \ldots \cdot H_j^{\mathrm{T}} x = 0$$

в силу того, что пары фильтров анализа и синтеза каждого из уровней разложения удовлетворяют условию биортогональности (6). ■

Не освещённым остался единственный вопрос, а именно: какое количество уровней вейвлет-разложения может быть построено или, другими словами, каков диапазон допустимых значений для параметра j. Согласно (3), разложение начинается с нулевого шага, поэтому наименьшим значением для j является 0. С другой стороны, длина последовательностей вейвлет-коэффициентов на j-м шаге разложения равна $n_j/2$, поэтому максимально возможное значение j должно быть таким, что 2^{j+1} делит n, обозначим это значение через J. Таким образом, номер уровня разложения j удовлетворяет условию

$$0 \leqslant j \leqslant J. \tag{8}$$

На основании сказанного приходим к следующему утверждению.

Теорема 1 (о биортогональном разбиении векторного пространства).
Существуют представления векторного пространства V

$$V = W_0 \oplus W_1 \oplus \ldots \oplus W_J \oplus V_J,$$
$$V = \tilde{W}_0 \oplus \tilde{W}_1 \oplus \ldots \oplus \tilde{W}_J \oplus \tilde{V}_J,$$

такие, что для каждого j, удовлетворяющего условию (8), подпространство V_j ортогонально $\tilde{W}_j$, а подпространство W_j ортогонально $\tilde{V}_j$.

Доказательство. Имея для каждого уровня вейвлет-разложения от 0 до J пары фильтров анализа и синтеза, удовлетворяющие условию биортогональности (6), и применяя на каждом из уровней лемму 1, в явном виде строим представления, описанные в условии теоремы. ■

Таким образом, задача построения цепочки вложенных подпространств сводится к задаче построения фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ j-го уровня вейвлет-разложения, для решения которой автором разработаны два подхода. В первом из них фильтр h^j выбирается заново для каждого уровня разложения, а фильтры g^j, $\tilde{h}^j$, $\tilde{g}^j$ достраиваются по нему при помощи методики, описанной в [10]. Второй подход основывается на использовании модифицированной версии алгоритма построения фильтров j-го уровня разложения, представленного в [3] для ортогональных вейвлет-преобразований. Суть модификации заключается в отказе от использования преобразования Фурье. Доказана корректность проделанной модификации, а также применимость модифицированного алгоритма для построения фильтров j-го уровня разложения биортогональных вейвлет-преобразований. Далее оба подхода рассмотрены более подробно.

2. Построение уровневых фильтров при помощи алгоритма Евклида

Пусть j удовлетворяет условию (8). Поставим уровневым фильтрам h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ длины n_j в соответствие многочлены

$$h^j(x) = \sum_{i=0}^{n_j-1} h_i^j x^i, \quad g^j(x) = \sum_{i=0}^{n_j-1} g_i^j x^i,$$
$$\tilde{h}^j(x) = \sum_{i=0}^{n_j-1} \tilde{h}_i^j x^i, \quad \tilde{g}^j(x) = \sum_{i=0}^{n_j-1} \tilde{g}_i^j x^i,$$

степени которых не первосходят $n_j - 1$, и представим эти многочлены в виде суммы полифазных компонент. Многочлены $h_e^j(x) = \sum\limits_{k=0}^{n_j/2-1} h_{2k}^j x^k$ и $h_o^j(x) = \sum\limits_{k=0}^{n_j/2-1} h_{2k+1}^j x^k$ называются полифазными компонентами многочлена $h^j(x)$ и содержат только его чётные и нечётные компоненты соответственно. Тогда многочлен $h^j(x)$ представим в виде суммы полифазных компонент $h_e^j(x)$ и $h_o^j(x)$:

$$h^j(x) = h_e^j(x^2) + x h_o^j(x^2).$$

Аналогичное представление существует и для многочленов $g^j(x)$, $\tilde{h}^j(x)$, $\tilde{g}^j(x)$. Степени полифазных компонент не превосходят $n_j/2 - 1$, и в дальнейшем эти компоненты рассматриваются как элементы кольца многочленов $\mathrm{GF}(q)[x]/(x^{n_j/2} - 1)$.

По аналогии с [8], при помощи полифазных компонент для пар фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ введём полифазные матрицы

$$P_j(x) = \begin{bmatrix} h_e^j(x) & g_e^j(x) \\ h_o^j(x) & g_o^j(x) \end{bmatrix} \quad \text{и} \quad \tilde{P}_j(x) = \begin{bmatrix} \tilde{h}_e^j(x) & \tilde{g}_e^j(x) \\ \tilde{h}_o^j(x) & \tilde{g}_o^j(x) \end{bmatrix}.$$

Как уже было отмечено, пары фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ должны удовлетворять условию биортогональности (6), которое эквивалентно условию точного восстановления (7). Для полифазных матриц, согласно [8], условие точного восстановления (7) перепишется в виде

$$P_j(x)\, \tilde{P}_j^{\mathrm{T}}\left(x^{n_j/2-1}\right) = I_2. \tag{9}$$

На основании результатов работы [10], пара фильтров $\tilde{h}^j$, $\tilde{g}^j$ может быть построена по паре h^j, g^j при помощи соотношений

$$\tilde{h}^j(x) = -xg(-x^{n_j-1}), \quad \tilde{g}^j(x) = xh(-x^{n_j-1}). \tag{10}$$

Таким образом, задача построения уровневых фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ сводится к задаче построения пары фильтров h^j, g^j, которая решается при помощи алгоритма построения комплементарных фильтров над конечными полями [10]. Пара фильтров называется комплементарной, если определитель соответствующей им полифазной матрицы равен единице. Согласно [10], полифазные компоненты $h_e^j(x)$ и $h_o^j(x)$ многочлена $h^j(x)$ при помощи алгоритма Евклида нахождения НОД из [13] представимы в виде

$$\begin{bmatrix} h_e^j(x) \\ h_o^j(x) \end{bmatrix} = \prod_{i=1}^{L_j} \begin{bmatrix} q_i^j(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K_j \\ 0 \end{bmatrix}, \tag{11}$$

где $q_i^j(x)$, $i = 1, \ldots, L_j$, — неполные частные; K_j — наибольший общий делитель многочленов $h_e^j(x)$ и $h_o^j(x)$. Заметим, что в силу специфики определения операции деления в кольце $\mathrm{GF}(q)[x]/(x^{n_j/2}-1)$ ответ, возвращаемый алгоритмом, не является единственно возможным. Таким образом, если K_j является обратимым в $\mathrm{GF}(q)[x]/(x^{n_j/2}-1)$, то при помощи полученного представления для фильтра h^j всегда может быть построен комплементарный ему фильтр g^j, причём их полифазная матрица $P_j(x)$ определяется как

$$P_j(x) = \begin{bmatrix} h_e^j(x) & g_e^j(x) \\ h_o^j(x) & g_o^j(x) \end{bmatrix} = \prod_{i=1}^{L_j} \begin{bmatrix} q_i^j(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K_j & 0 \\ 0 & (-1)^{L_j}(1/K_j) \end{bmatrix}, \tag{12}$$

где (-1) — элемент, обратный 1 по сложению в поле $\mathrm{GF}(q)$; $(1/K_j)$ — элемент, обратный K_j по умножению в кольце $\mathrm{GF}(q)[x]/(x^{n_j/2}-1)$. Стоит отметить, что фильтр g^j, комплементарный фильтру h^j, определяется не однозначно. В самом деле, пусть $s^j(x) \in \mathrm{GF}(q)[x]/(x^{n_j/2}-1)$ — некоторый многочлен, тогда фильтр g_s^j, которому соответствует многочлен $g_s^j(x) = g^j(x) + h^j(x)s^j(x^2) \bmod (x^{n_j}-1)$, также комплементарен фильтру h^j.

Таким образом, алгоритм, позволяющий генерировать пары уровневых фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ длины n_j, состоит из следующих шагов:

1) в качестве фильтра h^j выбрать последовательность произвольных элементов поля $\mathrm{GF}(q)$ длины n_j, такую, что наибольший общий делитель полифазных компонент $h_e^j(x)$ и $h_o^j(x)$ многочлена $h^j(x)$ является одночленом;
2) при помощи полученного алгоритмом Евклида представления (11), воспользовавшись соотношением (12), построить фильтр g^j, комплементарный h^j;
3) при помощи соотношений (10) построить пару фильтров $\tilde{h}^j$, $\tilde{g}^j$.

3. Построение уровневых фильтров при помощи модифицированного алгоритма Кайре — Гроссмана — Пура

Обратимся к описанию второго подхода к построению уровневых фильтров, который, в отличие от первого, для построения пар фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ использует исходные фильтры h, g и $\tilde{h}$, $\tilde{g}$ и основан на модифицированной версии алгоритма [3]. Пусть j удовлетворяет условию (8), проиллюстрируем шаги исходного алгоритма Кайре — Гроссмана — Пура на примере построения фильтра h^j с длиной n_j:

1) построить поле $\mathrm{GF}(q^r)$, такое, что оно содержит элемент α_0 порядка $n/2$;

2) для полифазных компонент $h_e(x)$ и $h_o(x)$ фильтра h вычислить последовательности значений

$$\left\{h_e\left(\alpha_0^{\frac{n}{n_j}i}\right)\right\}_{i=0}^{n_j/2-1} \quad \text{и} \quad \left\{h_o\left(\alpha_0^{\frac{n}{n_j}i}\right)\right\}_{i=0}^{n_j/2-1}; \tag{13}$$

3) применить обратное преобразование Фурье к полученным последовательностям и с его помощью вычислить полифазные компоненты $h_e^j(x)$ и $h_o^j(x)$ фильтра h^j;
4) восстановить фильтр h^j по его полифазным компонентам $h_e^j(x)$ и $h_o^j(x)$.

Несмотря на простоту шагов, применимость данного алгоритма ограничена, так как используемое в нём преобразование Фурье не существует для исходных фильтров произвольной длины n, поэтому основная цель проделанных модификаций заключается в отказе от преобразования Фурье.

Рассмотрим сумму $\sum\limits_{k=0}^{n/2-1} a_k\alpha_0^{\frac{n}{n_j}ik}$, где a_k, $k=0,\ldots,n/2-1$, — элементы расширения поля $\mathrm{GF}(q)$, в котором находится α_0. С учётом того, что элемент α_0^{n/n_j} имеет порядок $n_j/2$, в последовательности значений

$$\left\{\alpha_0^{\frac{n}{n_j}ik}\right\}_{k=0}^{n/2-1}$$

можно выделить повторяющиеся участки длиной $n_j/2$, что позволяет представить исходную сумму в виде двух сумм:

$$\begin{aligned}
\sum_{k=0}^{n/2-1} a_k\alpha_0^{\frac{n}{n_j}ik} &= \sum_{k=0}^{n/2-1} a_k\left(\alpha_0^{\frac{n}{n_j}k}\right)^i = a_0\left(\alpha_0^{\frac{n}{n_j}0}\right)^i + \ldots + a_{n_j/2-1}\left(\alpha_0^{\frac{n}{n_j}(n_j/2-1)}\right)^i + \\
&+a_{n_j/2}\left(\alpha_0^{\frac{n}{n_j}n_j/2}\right)^i + \ldots + a_{2n_j/2-1}\left(\alpha_0^{\frac{n}{n_j}(2n_j/2-1)}\right)^i + \ldots + \\
&+a_{n/2-n_j/2}\left(\alpha_0^{\frac{n}{n_j}(n/2-n_j/2)}\right)^i + \ldots + a_{n/2-1}\left(\alpha_0^{\frac{n}{n_j}(n/2-1)}\right)^i = \\
&= a_0\left(\alpha_0^{\frac{n}{n_j}0}\right)^i + \ldots + a_{n_j/2-1}\left(\alpha_0^{\frac{n}{n_j}(n_j/2-1)}\right)^i + \\
&+a_{n_j/2}\left(\alpha_0^{\frac{n}{n_j}0}\right)^i + \ldots + a_{2n_j/2-1}\left(\alpha_0^{\frac{n}{n_j}(n_j/2-1)}\right)^i + \ldots + \\
&+a_{n/2-n_j/2}\left(\alpha_0^{\frac{n}{n_j}0}\right)^i + \ldots + a_{n/2-1}\left(\alpha_0^{\frac{n}{n_j}(n_j/2-1)}\right)^i = \sum_{k_1=0}^{n_j/2-1}\alpha_0^{\frac{n}{n_j}ik_1}\sum_{k_2=0}^{n/n_j-1} a_{(n_j/2)k_2+k_1}.
\end{aligned}$$

Воспользовавшись результатами выкладок, получаем, что последовательности значений (13) могут быть записаны как

$$\left\{\sum_{k_1=0}^{n_j/2-1}\alpha_0^{\frac{n}{n_j}ik_1}\sum_{k_2=0}^{n/n_j-1} h_{e,(n_j/2)k_2+k_1}\right\}_{i=0}^{n_j/2-1} \quad \text{и} \quad \left\{\sum_{k_1=0}^{n_j/2-1}\alpha_0^{\frac{n}{n_j}ik_1}\sum_{k_2=0}^{n/n_j-1} h_{o,(n_j/2)k_2+k_1}\right\}_{i=0}^{n_j/2-1}$$

и являются спектрами многочленов

$$\sum_{k_1=0}^{n_j/2-1} x^{k_1}\sum_{k_2=0}^{n/n_j-1} h_{e,(n_j/2)k_2+k_1} \quad \text{и} \quad \sum_{k_1=0}^{n_j/2-1} x^{k_1}\sum_{k_2=0}^{n/n_j-1} h_{o,(n_j/2)k_2+k_1}$$

соответственно, степени которых не первосходят $n_j/2-1$; через $h_{e,k}$ и $h_{o,k}$ обозначены коэффициенты многочленов $h_e(x)$ и $h_o(x)$ при степени k. Это означает, что многочлены $h_e^j(x)$ и $h_o^j(x)$, соответствующие спектрам (13), могут быть получены в виде

$$h_e^j(x) = \sum_{k_1=0}^{n_j/2-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{e,(n_j/2)k_2+k_1} = \sum_{k_1=0}^{n_j/2-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{2((n_j/2)k_2+k_1)},$$
$$h_o^j(x) = \sum_{k_1=0}^{n_j/2-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{o,(n_j/2)k_2+k_1} = \sum_{k_1=0}^{n_j/2-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{2((n_j/2)k_2+k_1)+1}.$$

Восстановим многочлен $h^j(x)$ по его полифазным компонентам $h_e^j(x)$ и $h_o^j(x)$:

$$h^j(x) = \sum_{k_1=0}^{n_j-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{n_jk_2+k_1}. \tag{14}$$

В самом деле,

$$h^j(x) = h_e^j(x^2) + xh_o^j(x^2) = \sum_{k_1=0}^{n_j/2-1} x^{2k_1} \sum_{k_2=0}^{n/n_j-1} h_{2((n_j/2)k_2+k_1)} +$$
$$+x \sum_{k_1=0}^{n_j/2-1} x^{2k_1} \sum_{k_2=0}^{n/n_j-1} h_{2((n_j/2)k_2+k_1)+1} =$$
$$= \sum_{k_1=0}^{n_j/2-1} x^{2k_1} \sum_{k_2=0}^{n/n_j-1} h_{n_jk_2+2k_1} + \sum_{k_1=0}^{n_j/2-1} x^{2k_1+1} \sum_{k_2=0}^{n/n_j-1} h_{n_jk_2+2k_1+1} = \sum_{k_1=0}^{n_j-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{n_jk_2+k_1}.$$

Построение уровневых фильтров g^j, $\tilde{h}^j$, $\tilde{g}^j$ описывается аналогичными соотношениями. Отметим, что в (14) для построения уровневого фильтра h^j используется фильтр h, однако его можно переписать таким образом, чтобы для построения h^j использовался фильтр h^{j-1}:

$$h^j(x) = \sum_{k=0}^{n_j-1} \left(h_k^{j-1} + h_{n_j+k}^{j-1}\right) x^k. \tag{15}$$

В самом деле,

$$\sum_{k=0}^{n_j-1} \left(h_k^{j-1} + h_{n_j+k}^{j-1}\right) x^k = \sum_{k_1=0}^{n_j-1} x^{k_1} \left(h_{k_1}^{j-1} + h_{n_j+k_1}^{j-1}\right) =$$
$$= \sum_{k_1=0}^{n_j-1} x^{k_1} \left(\sum_{k_2=0}^{(n/n_{j-1})-1} h_{n_{j-1}k_2+k_1} + \sum_{k_2=0}^{(n/n_{j-1})-1} h_{n_{j-1}k_2+n_j+k_1} \right) =$$
$$= \sum_{k_1=0}^{n_j-1} x^{k_1} \left(\sum_{k_2=0}^{(n/(2n_j))-1} h_{2n_jk_2+k_1} + \sum_{k_2=0}^{(n/(2n_j))-1} h_{2n_jk_2+n_j+k_1} \right) =$$
$$= \sum_{k_1=0}^{n_j-1} x^{k_1} \sum_{k_2=0}^{(n/(2n_j))-1} \left(h_{2n_jk_2+k_1} + h_{n_j(2k_2+1)+k_1}\right) = \sum_{k_1=0}^{n_j-1} x^{k_1} \sum_{k_2=0}^{n/n_j-1} h_{n_jk_2+k_1}.$$

Докажем две леммы, обосновывающих корректность использования описанного подхода для построения уровневых фильтров биортогонального вейвлет-преобразования.

Лемма 2. Если пары фильтров h^{j-1}, g^{j-1} и $\tilde{h}^{j-1}$, $\tilde{g}^{j-1}$ являются комплементарными, то полученные из них при помощи соотношения (15) пары h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ также являются комплементарными.

Доказательство. Без ограничения общности докажем утверждение только для пары фильтров h^j, g^j, доказательство для пары $\tilde{h}^j$, $\tilde{g}^j$ аналогично. В связи с использованием соотношения (15) представим полифазные компоненты фильтров h^{j-1}, g^{j-1} в виде

$$\begin{aligned} h_e^{j-1}(x) &= h_{e,0}^{j-1}(x) + x^{n_j/2} h_{e,1}^{j-1}(x), \quad h_o^{j-1}(x) = h_{o,0}^{j-1}(x) + x^{n_j/2} h_{o,1}^{j-1}(x), \\ g_e^{j-1}(x) &= g_{e,0}^{j-1}(x) + x^{n_j/2} g_{e,1}^{j-1}(x), \quad g_o^{j-1}(x) = g_{o,0}^{j-1}(x) + x^{n_j/2} g_{o,1}^{j-1}(x). \end{aligned} \tag{16}$$

По условию фильтры h^{j-1} и g^{j-1} являются комплементарными, поэтому, с учётом введённых обозначений, условие комплементарности для них запишется как

$$\begin{gathered} \det P_{j-1}(x) = h_e^{j-1}(x) g_o^{j-1}(x) - h_o^{j-1}(x) g_e^{j-1}(x) = \\ = \left(h_{e,0}^{j-1}(x) + x^{n_j/2} h_{e,1}^{j-1}(x)\right)\left(g_{o,0}^{j-1}(x) + x^{n_j/2} g_{o,1}^{j-1}(x)\right) - \\ - \left(h_{o,0}^{j-1}(x) + x^{n_j/2} h_{o,1}^{j-1}(x)\right)\left(g_{e,0}^{j-1}(x) + x^{n_j/2} g_{e,1}^{j-1}(x)\right) = \\ = \left(h_{e,0}^{j-1}(x) g_{o,0}^{j-1}(x) - h_{o,0}^{j-1}(x) g_{e,0}^{j-1}(x)\right) + x^{n_j/2}\left(h_{e,1}^{j-1}(x) g_{o,0}^{j-1}(x) + h_{e,0}^{j-1}(x) g_{o,1}^{j-1}(x) - \right. \\ \left. - h_{o,1}^{j-1}(x) g_{e,0}^{j-1}(x) - h_{o,0}^{j-1}(x) g_{e,1}^{j-1}(x)\right) + x^{n_j}\left(h_{e,1}^{j-1}(x) g_{o,1}^{j-1}(x) - h_{o,1}^{j-1}(x) g_{e,1}^{j-1}(x)\right) = \\ = \left(h_{e,0}^{j-1}(x) g_{o,0}^{j-1}(x) + h_{e,1}^{j-1}(x) g_{o,1}^{j-1}(x) - h_{o,0}^{j-1}(x) g_{e,0}^{j-1}(x) - h_{o,1}^{j-1}(x) g_{e,1}^{j-1}(x)\right) + \\ + x^{n_j/2}\left(h_{e,1}^{j-1}(x) g_{o,0}^{j-1}(x) + h_{e,0}^{j-1}(x) g_{o,1}^{j-1}(x) - h_{o,1}^{j-1}(x) g_{e,0}^{j-1}(x) - \right. \\ \left. - h_{o,0}^{j-1}(x) g_{e,1}^{j-1}(x)\right) \bmod (x^{n_j} - 1) = p_0(x) + x^{n_j/2} p_1(x) \bmod (x^{n_j} - 1) = 1 \bmod (x^{n_j} - 1). \end{gathered}$$

Теперь проверим выполнимость условия комплементарности для пары фильтров h^j, g^j:

$$\begin{gathered} \det P_j(x) = h_e^j(x) g_o^j(x) - h_o^j(x) g_e^j(x) = \\ = \left(h_{e,0}^{j-1}(x) + h_{e,1}^{j-1}(x)\right)\left(g_{o,0}^{j-1}(x) + g_{o,1}^{j-1}(x)\right) - \left(h_{o,0}^{j-1}(x) + h_{o,1}^{j-1}(x)\right)\left(g_{e,0}^{j-1}(x) + g_{e,1}^{j-1}(x)\right) = \\ = p_0(x) + p_1(x) = p_0(x) + x^{n_j/2} p_1(x) \bmod \left(x^{n_j/2} - 1\right) = 1 \bmod \left(x^{n_j/2} - 1\right). \end{gathered}$$

Лемма 2 доказана. ■

Замечание 1. Пары фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$, полученные из комплементарных пар h, g и $\tilde{h}$, $\tilde{g}$ соответственно при помощи соотношения (14), являются комплементарными для любого j, удовлетворяющего условию (8).

Лемма 3. Если пары фильтров h^{j-1}, g^{j-1} и $\tilde{h}^{j-1}$, $\tilde{g}^{j-1}$ удовлетворяют условию точного восстановления (9), то полученные из них при помощи соотношения (15) пары h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$ также ему удовлетворяют.

Доказательство. Пары фильтров h^{j-1}, g^{j-1} и $\tilde{h}^{j-1}$, $\tilde{g}^{j-1}$, по условию леммы, удовлетворяют условию точного восстановления, которое для них записывается в виде

$$\begin{gathered} P_{j-1}(x)\, \tilde{P}_{j-1}^{\mathrm{T}}\left(x^{(n_{j-1}/2)-1}\right) = \begin{bmatrix} h_e^{j-1}(x) & g_e^{j-1}(x) \\ h_o^{j-1}(x) & g_o^{j-1}(x) \end{bmatrix} \begin{bmatrix} \tilde{h}_e^{j-1}(x^{n_j-1}) & \tilde{h}_o^{j-1}(x^{n_j-1}) \\ \tilde{g}_e^{j-1}(x^{n_j-1}) & \tilde{g}_o^{j-1}(x^{n_j-1}) \end{bmatrix} = \\ = \begin{bmatrix} h_e^{j-1}(x)\tilde{h}_e^{j-1}(x^{n_j-1}) + g_e^{j-1}(x)\tilde{g}_e^{j-1}(x^{n_j-1}) & h_e^{j-1}(x)\tilde{h}_o^{j-1}(x^{n_j-1}) + g_e^{j-1}(x)\tilde{g}_o^{j-1}(x^{n_j-1}) \\ h_o^{j-1}(x)\tilde{h}_e^{j-1}(x^{n_j-1}) + g_o^{j-1}(x)\tilde{g}_e^{j-1}(x^{n_j-1}) & h_o^{j-1}(x)\tilde{h}_o^{j-1}(x^{n_j-1}) + g_o^{j-1}(x)\tilde{g}_o^{j-1}(x^{n_j-1}) \end{bmatrix} = I_2. \end{gathered}$$

Выполнимость условия точного восстановления для пар фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$, в свою очередь, означает выполнимость равенства

$$\begin{gathered} P_j(x)\, \tilde{P}_j^{\mathrm{T}}\left(x^{(n_j/2)-1}\right) = \begin{bmatrix} h_e^j(x) & g_e^j(x) \\ h_o^j(x) & g_o^j(x) \end{bmatrix} \begin{bmatrix} \tilde{h}_e^j\left(x^{(n_j/2)-1}\right) & \tilde{h}_o^j\left(x^{(n_j/2)-1}\right) \\ \tilde{g}_e^j\left(x^{(n_j/2)-1}\right) & \tilde{g}_o^j\left(x^{(n_j/2)-1}\right) \end{bmatrix} = \\ = \begin{bmatrix} h_e^j(x)\tilde{h}_e^j\left(x^{(n_j/2)-1}\right) + g_e^j(x)\tilde{g}_e^j\left(x^{(n_j/2)-1}\right) & h_e^j(x)\tilde{h}_o^j\left(x^{(n_j/2)-1}\right) + g_e^j(x)\tilde{g}_o^j\left(x^{(n_j/2)-1}\right) \\ h_o^j(x)\tilde{h}_e^j\left(x^{(n_j/2)-1}\right) + g_o^j(x)\tilde{g}_e^j\left(x^{(n_j/2)-1}\right) & h_o^j(x)\tilde{h}_o^j\left(x^{(n_j/2)-1}\right) + g_o^j(x)\tilde{g}_o^j\left(x^{(n_j/2)-1}\right) \end{bmatrix} = I_2. \end{gathered}$$

Без ограничения общности докажем, что из условия

$$h_e^{j-1}(x)\tilde{h}_e^{j-1}(x^{n_j-1}) + g_e^{j-1}(x)\tilde{g}_e^{j-1}(x^{n_j-1}) = 1 \bmod (x^{n_j}-1)$$

следует равенство

$$h_e^{j}(x)\tilde{h}_e^{j}\left(x^{(n_j/2)-1}\right) + g_e^{j}(x)\tilde{g}_e^{j}\left(x^{(n_j/2)-1}\right) = 1 \bmod \left(x^{n_j/2}-1\right),$$

три аналогичных утверждения доказываются схожим образом.

По аналогии с (16), представим полифазные компоненты $\tilde{h}_e^{j-1}(x)$ и $\tilde{g}_e^{j-1}(x)$ в виде

$$\tilde{h}_e^{j-1}(x) = \tilde{h}_{e,0}^{j-1}(x) + x^{n_j/2}\tilde{h}_{e,1}^{j-1}(x), \quad \tilde{g}_e^{j-1}(x) = \tilde{g}_{e,0}^{j-1}(x) + x^{n_j/2}\tilde{g}_{e,1}^{j-1}(x).$$

Рассмотрим многочлен $a(x) = \sum\limits_{k=0}^{n_j-1} a_k x^k$ степени n_j-1 над полем $\mathrm{GF}(q)$, представимый как $a(x) = a_0(x) + x^{n_j/2}a_1(x)$, где $a_0(x) = \sum\limits_{k=0}^{(n_j/2)-1} a_k x^k$ и $a_1(x) = \sum\limits_{k=0}^{(n_j/2)-1} a_{(n_j/2)+k}x^k$ — многочлены над полем $\mathrm{GF}(q)$, степени которых не превосходят $n_j/2-1$. Тогда

$$a_0\left(x^{(n_j/2)-1}\right) \bmod \left(x^{n_j/2}-1\right) = a_0 + \sum_{k=1}^{(n_j/2)-1} a_k x^{(n_j/2)(k-1)+(n_j/2)-k} \bmod \left(x^{n_j/2}-1\right) =$$
$$= a_0 + \sum_{k=1}^{(n_j/2)-1} a_k x^{(n_j/2)-k} = a_0 + a_{0,1}^*(x),$$
$$a_1\left(x^{(n_j/2)-1}\right) \bmod \left(x^{n_j/2}-1\right) = a_{n_j/2} + \sum_{k=1}^{(n_j/2)-1} a_{n_j/2+k} x^{(n_j/2)(k-1)+(n_j/2)-k} \bmod \left(x^{n_j/2}-1\right) =$$
$$= a_{n_j/2} + \sum_{k=1}^{(n_j/2)-1} a_{n_j/2+k} x^{(n_j/2)-k} = a_{n_j/2} + a_{1,1}^*(x),$$

где $a_{0,1}^*(x)$ и $a_{1,1}^*(x)$ — многочлены над полем $\mathrm{GF}(q)$, свободный член которых равен нулю, а степени не превосходят $n_j/2-1$. В свою очередь, по аналогии получим

$$a(x^{n_j-1}) \bmod (x^{n_j}-1) = a_0 + \sum_{k=1}^{n_j-1} a_k x^{n_j-k} =$$
$$= a_0 + \sum_{k=1}^{(n_j/2)-1} a_k x^{n_j-k} + a_{n_j/2}x^{n_j/2} + \sum_{k=(n_j/2)+1}^{n_j-1} a_k x^{n_j-k} =$$
$$= a_0 + x^{n_j/2}\sum_{k=1}^{(n_j/2)-1} a_k x^{(n_j/2)-k} + a_{n_j/2}x^{n_j/2} + \sum_{k=1}^{(n_j/2)-1} a_{n_j/2+k} x^{(n_j/2)-k} =$$
$$= a_0 + a_{1,1}^*(x) + x^{n_j/2}\left(a_{n_j/2} + a_{0,1}^*(x)\right).$$

Таким образом, имеем

$$\begin{aligned}
&\tilde{h}_e^{j-1}(x^{n_j-1}) \bmod (x^{n_j}-1) = \tilde{h'}_{e,0}(x) + x^{n_j/2}\tilde{h'}_{e,1}(x),\\
&\tilde{h}_e^{j}\left(x^{(n_j/2)-1}\right) \bmod \left(x^{n_j/2}-1\right) = \tilde{h'}_{e,0}(x) + \tilde{h'}_{e,1}(x),\\
&\tilde{g}_e^{j-1}(x^{n_j-1}) \bmod (x^{n_j}-1) = \tilde{g'}_{e,0}(x) + x^{n_j/2}\tilde{g'}_{e,1}(x),\\
&\tilde{g}_e^{j}\left(x^{(n_j/2)-1}\right) \bmod \left(x^{n_j/2}-1\right) = \tilde{g'}_{e,0}(x) + \tilde{g'}_{e,1}(x),
\end{aligned} \tag{17}$$

степени многочленов $\tilde{h'}_{e,0}(x)$, $\tilde{h'}_{e,1}(x)$, $\tilde{g'}_{e,0}(x)$ и $\tilde{g'}_{e,1}(x)$ не превосходят $n_j/2-1$.

На основании соотношений (16) и (17), осуществив выкладки, аналогичные проделанным при доказательстве леммы 2, получаем

$$h_e^{j-1}(x)\tilde{h}_e^{j-1}(x^{n_j-1}) + g_e^{j-1}(x)\tilde{g}_e^{j-1}(x^{n_j-1}) \bmod (x^{n_j}-1) =$$
$$= p_0(x) + x^{n_j/2}p_1(x) \bmod (x^{n_j}-1) = 1 \bmod (x^{n_j}-1)$$

и, в свою очередь,

$$h_e^j(x)\tilde{h}_e^j\left(x^{(n_j/2)-1}\right) + g_e^j(x)\tilde{g}_e^j\left(x^{(n_j/2)-1}\right) \bmod \left(x^{n_j/2}-1\right) = p_0(x) + p_1(x) \bmod \left(x^{n_j/2}-1\right) =$$
$$= p_0(x) + x^{n_j/2}p_1(x) \bmod \left(x^{n_j/2}-1\right) = 1 \bmod \left(x^{n_j/2}-1\right).$$

Лемма 3 доказана. ■

Замечание 2. Пары фильтров h^j, g^j и $\tilde{h}^j$, $\tilde{g}^j$, полученные из удовлетворяющих условию точного восстановления (9) пар h, g и $\tilde{h}$, $\tilde{g}$ при помощи соотношения (14), также удовлетворяют условию точного восстановления для любого j, подпадающего под ограничение (8).

Таким образом, описанный подход позволяет генерировать фильтры j-го уровня разложения как по исходным фильтрам при помощи соотношения (14), так и по фильтрам $(j-1)$-го уровня разложения при помощи соотношения (15).

4. Схема построения биортогональных вейвлет-кодов, использующих несколько уровней вейвлет-разложения

Изложенные подходы к построению уровневых фильтров дают возможность построить на их основе новое семейство биортогональных вейвлет-кодов, использующих несколько уровней вейвлет-разложения. Основная идея, лежащая в основе нового семейства кодов, заключается в применении на j-м уровне разложения представленной на рис. 1 процедуры кодирования собщений, использующей уровневые фильтры h^j и g^j. При этом на вход подаётся результат кодирования, полученный на $(j+1)$-м уровне разложения. Схему построения биортогональных вейвлет-кодов, использующих несколько уровней вейвлет-разложения, будем называть пирамидальной. Рассмотрим её подробнее.

Обозначим v^j — информационное слово, кодируемое на j-м уровне вейвлет-разложения; c^j — результат кодирования. Тогда, согласно описанному принципу, $v^{j-1} = c^j$, $v^J = v$, $c = c^0$. Кодирование на j-м уровне разложения осуществляется при помощи матрицы

$$A_j = H_j^{\mathrm{T}} + \lambda_j G_j^{\mathrm{T}}\Pi_j, \tag{18}$$

где Π_j — циркулянтная матрица размера $n_j/2 \times n_j/2$, первая строка которой имеет вид $(0,0,0,\ldots,1)$; λ_j — произвольный ненулевой элемент поля $\mathrm{GF}(q)$. Тогда порождающая матрица кода имеет вид

$$A = \prod_{j=0}^{J}\left(H_j^{\mathrm{T}} + \lambda_j G_j^{\mathrm{T}}\Pi_j\right).$$

Проверочная матрица кода находится стандартным способом [9].

Лемма 4. Циклический сдвиг кодового слова c на 2^{J+1} позиций вправо также является кодовым словом.

Доказательство. Покажем, что $\hat{c}^j = A_j\hat{v}^j$, где $\hat{v}^j$ является циклическим сдвигом v^j на Δ позиций вправо, является циклическим сдвигом $c^j = A_j v^j$ на 2Δ позиций

вправо. В самом деле, $c^j = A_j v^j = \sum\limits_{i=0}^{n_j/2} v_i^j \mathrm{col}_i A_j$, где v_i^j — i-я компонента вектора v^j; $\mathrm{col}_i A_j$ — i-й столбец матрицы A_j. С другой стороны,

$$\hat{c}^j = A_j \hat{v}_j = \sum_{i=0}^{n_j/2} \hat{v}_i^j \mathrm{col}_i A_j = \sum_{i=0}^{n_j/2} v^j_{(i+\Delta) \bmod (n_j/2)} \mathrm{col}_i A_j = \sum_{i=0}^{n_j/2} v_i^j \mathrm{col}_{(i+\Delta) \bmod (n_j/2)} A_j,$$

где $\hat{v}_i^j$ — i-я компонента вектора $\hat{v}^j$. По построению A_j^{T} является 2-циркулянтной матрицей, следовательно, столбец матрицы A_j с номером $(i+\Delta) \bmod (n_j/2)$ является циклическим сдвигом i-го столбца на 2Δ позиций вправо; значит, $\hat{c}_j$ является циклическим сдвигом c_j на 2Δ позиций вправо.

Пусть информационное слово $\hat{v}$ является циклическим сдвигом информационного слова v на одну позицию вправо. Тогда, согласно доказанному выше, $\hat{c}^J$ — циклический сдвиг c^J на 2 позиции вправо, $\hat{c}^{J-1}$ — циклический сдвиг c^{J-1} на 4 позиции вправо, и так далее. Кодовое слово $\hat{c}$ является циклическим сдвигом кодового слова c на 2^{J+1} позиций вправо. ■

Это означает, что, согласно [9], представленные коды также относятся к подклассу квазициклических кодов, но уже с циклическим сдвигом кодовых слов на 2^{J+1} позиций.

Таким образом, описана пирамидальная схема построения биортогональных вейвлет-кодов над полем $\mathrm{GF}(q)$, использующих несколько уровней вейвлет-разложения и имеющих длину кодовых и информационных слов n и $n/2^{J+1}$ соответственно и скорость кодирования $1/2^{J+1}$. Её ключевым преимуществом является возможность варьировать скорость кодирования путём изменения числа используемых уровней вейвлет-разложения, в то время как в представленных в предшествующих работах схемах скорость кодирования всегда равна $1/2$. Для иллюстрации схемы построения кода и непосредственно процесса кодирования рассмотрим несколько примеров. При анализе свойств кодов используется относительное кодовое расстояние, определяемое как отношение кодового расстояния к длине кодовых слов:

$$\delta = \frac{d}{n}.$$

Пример 1. Построим биортогональный вейвлет-код над полем $\mathrm{GF}(2)$ с длиной кодовых и информационных слов $n = 16$ и $k = 4$ соответственно, использующий два уровня вейвлет-разложения. Фильтр h зададим при помощи многочлена

$$h(x) = 1 + x + x^3 + x^4 + x^{13}.$$

Воспользовавшись алгоритмом п. 2, построим фильтры для нулевого и первого уровней вейвлет-разложения. Фильтр h^0 зададим при помощи многочлена $h^0(x) = h(x)$, полифазные компоненты которого определяются как

$$h_e^0(x) = 1 + x^2 \quad \text{и} \quad h_o^0(x) = 1 + x + x^6.$$

При помощи алгоритма Евклида находим представление полифазных компонент $h_e^0(x)$ и $h_o^0(x)$ в виде (11):

$$\begin{bmatrix} h_e^0(x) \\ h_o^0(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 + x^2 + x^4 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Отсюда при помощи соотношения (12) получаем

$$\begin{bmatrix} h_e^0(x) & g_e^0(x) \\ h_o^0(x) & g_o^0(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1+x^2+x^4 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

что позволяет задать фильтр g^0:

$$g^0(x) = x + x^2 + x^3 + x^7 + x^{11}.$$

В соответствии с (10) находим пару фильтров $\tilde{h}^0$ и $\tilde{g}^0$, она определяется многочленами

$$\tilde{h}^0(x) = 1 + x^6 + x^{10} + x^{14} + x^{15},$$
$$\tilde{g}^0(x) = 1 + x + x^4 + x^{13} + x^{14}.$$

Фильтр h^1 зададим при помощи многочлена $h^1(x) = h_o^0(x)$, полифазные компоненты которого суть

$$h_e^1(x) = 1 + x^3 \quad \text{и} \quad h_o^1(x) = 1.$$

При помощи алгоритма Евклида находим представление полифазных компонент $h_e^1(x)$ и $h_o^1(x)$ в виде (11):

$$\begin{bmatrix} h_e^1(x) \\ h_o^1(x) \end{bmatrix} = \begin{bmatrix} 1+x^3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Отсюда по соотношению (12) получаем

$$\begin{bmatrix} h_e^1(x) & g_e^1(x) \\ h_o^1(x) & g_o^1(x) \end{bmatrix} = \begin{bmatrix} 1+x^3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

что позволяет задать фильтр g^1 многочленом

$$g^1(x) = 1.$$

Воспользовавшись соотношениями (10), находим пару фильтров $\tilde{h}^1$ и $\tilde{g}^1$, она задаётся многочленами

$$\tilde{h}^1(x) = x,$$
$$\tilde{g}^1(x) = 1 + x + x^3.$$

Теперь, когда найдены фильтры для каждого из уровней разложения, по (18) вычислим матрицы, при помощи которых осуществляется кодирование:
– матрица нулевого уровня вейвлет-разложения:

$$A_0 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^{\mathrm{T}};$$

– матрица первого уровня:

$$A_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}^{\mathrm{T}}.$$

Порождающая матрица кода имеет следующий вид:

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}^{\mathrm{T}}.$$

Экспериментально установлено, что кодовое расстояние биортогональных вейвлет-кодов с порождающими матрицами A_1 и A равны 4 и 8 соответственно, относительное кодовое расстояние при этом осталось неизменным. Кодовому слову $(1, 0, 0, 0, 1, 0, 1, 1)$ веса 4 соответствует информационное слово $(0, 0, 0, 1)$, кодовому слову $(1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1)$ веса 8 также соответствует информационное слово $(0, 0, 0, 1)$.

Пример 2. Построим биортогональный вейвлет-код над полем GF(3) с длиной кодовых и информационных слов $n = 12$ и $k = 3$ соответственно, использующий два уровня вейвлет-разложения. Параметры λ_0 и λ_1 примем равными единице, фильтр h зададим при помощи многочлена

$$h(x) = 1 + x^7.$$

По алгоритму п. 2 построим фильтры для нулевого уровня разложения; фильтры для первого уровня построим при помощи алгоритма из п. 3. Фильтр h^0 зададим многочленом $h^0(x) = h(x)$, полифазные компоненты которого определяются как

$$h_e^0(x) = 1 \quad \text{и} \quad h_o^0(x) = x^3.$$

По алгоритму Евклида находим представление полифазных компонент $h_e^0(x)$ и $h_o^0(x)$ в виде (11)

$$\begin{bmatrix} h_e^0(x) \\ h_o^0(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x^3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Отсюда по (12) получаем

$$\begin{bmatrix} h_e^0(x) & g_e^0(x) \\ h_o^0(x) & g_o^0(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x^3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

что позволяет задать фильтр g^0 многочленом

$$g^0(x) = x.$$

Воспользовавшись соотношениями (10), находим пару фильтров $\tilde{h}^0$ и $\tilde{g}^0$:

$$\tilde{h}^0(x) = 1,$$
$$\tilde{g}^0(x) = x + 2x^6.$$

По (15) вычисляем фильтры для первого уровня разложения, они задаются такими многочленами:

$$h^1(x) = 1 + x,$$
$$g^1(x) = x,$$
$$\tilde{h}^1(x) = 1,$$
$$\tilde{g}^1(x) = 2 + x.$$

Теперь, когда найдены фильтры для каждого из уровней разложения, в соответствии с (18) вычислим матрицы, при помощи которых осуществляется кодирование:
– матрица нулевого уровня вейвлет-разложения:

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}};$$

– матрица первого уровня:

$$A_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}^{\mathrm{T}}.$$

Порождающая матрица кода имеет вид

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 2 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}.$$

Экспериментально установлено, что кодовое расстояние биортогональных вейвлет-кодов с порождающими матрицами A_1 и A равны 3 и 7 соответственно, относительное кодовое расстояние при этом выросло. Кодовому слову $(0, 1, 0, 0, 1, 1)$ веса 3 соответствует информационное слово $(0, 0, 1)$, кодовому слову $(1, 1, 0, 0, 2, 1, 0, 0, 2, 0, 1, 1)$ веса 7 также соответствует информационное слово $(1, 2, 2)$.

Заключение

В работе доказана теорема о существовании двух представлений векторного пространства в виде прямых сумм подпространств, связанных условиями биортогональности, и описаны два подхода к построению уровневых фильтров биортогонального вейвлет-преобразования. Первый подход основан на ранее использовавшемся для построения комплементарных фильтров алгоритме Евклида. Второй подход основан на модификации алгоритма [3], цель которой заключалась в отказе от применения преобразования Фурье. Доказана корректность подхода.

Теорема доказана конструктивным образом, что позволило описать пирамидальную схему построения биортогональных вейвлет-кодов, использующих несколько уровней вейвлет-разложения и имеющих скорость кодирования 2^{-L}, где L — количество уровней. Построены коды $W[16, 4, 8]$ над полем $\mathrm{GF}(2)$ и $W[12, 3, 7]$ над полем $\mathrm{GF}(3)$, численные эксперименты с которыми показали, что при определённом выборе исходных фильтров использование при кодировании нескольких уровней вейвлет-разложения приводит к росту относительного кодового расстояния.

ЛИТЕРАТУРА

1. *Mallat S.* Wavelet Tour of Signal Processing. 2nd ed. Boston: Academic Press, 1999. 799 p.
2. *Воробьев В. И., Грибунин В. Г.* Теория и практика вейвлет-преобразования. СПб.: ВУС, 1999. 206 с.
3. *Caire G., Grossman R. L., and Poor H. V.* Wavelet transforms associated with finite cyclic groups // IEEE Trans. Inform. Theory. 1993. V. 39. No. 4. P. 1157–1166.

4. *Fekri F., Mersereau R. M., and Schafer R. W.* Theory of wavelet transform over finite fields // Proc. IEEE Intern. Conf. Acoustics, Speech, and Signal Processing. 1999. V. 3. P. 1213–1216.
5. *Fekri F., McLaughlin S. W., Mersereau R. M., and Schafer R. W.* Double circulant self-dual codes using finite-field wavelet transforms // Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Berlin: Springer, 1999. P. 355–363.
6. *Fekri F., McLaughlin S. W., Mersereau R. M., and Schafer R. W.* Error control coding using finite-field wavelet transforms // Atlanta: Center for Signal Image Processing, 1999. P. 1–13.
7. *Fekri F., Mersereau R. M., and Schafer R. W.* Theory of paraunitary filter banks over fields of characteristic two // IEEE Trans. Inform. Theory. 2002. V. 48. No. 11. P. 2964–2979.
8. *Черников Д. В.* Помехоустойчивое кодирование с использованием биортогональных наборов фильтров точного восстановления // Труды конференции «Информационные технологии и системы». Светлогорск, 2013. С. 507–512.
9. *Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А.* Теория кодов, исправляющих ошибки. М.: Связь, 1974. 744 с.
10. *Черников Д. В.* Помехоустойчивое кодирование с использованием биортогональных наборов фильтров // Сибирские электрон. матем. известия. 2015. Т. 12. С. 704–713.
11. *Соловьев А. А., Черников Д. В.* Биортогональные вейвлет-коды с заданным кодовым расстоянием // Дискретная математика. 2017. Т. 29. № 2 С. 96–108.
12. *Соловьев А. А., Черников Д. В.* Биортогональные вейвлет-коды в полях характеристики два // Челяб. физ.-мат. журн. 2017. Т. 2. № 1. С. 66–79.
13. *Doubechies I. and Sweldens W.* Factoring wavelet transforms into lifting steps // J. Fourier Analysis Appl. 1998. V. 4. No. 3. P. 247–269.

REFERENCES

1. *Mallat S.* Wavelet Tour of Signal Processing, 2nd ed. Boston, Academic Press, 1999. 799 p.
2. *Vorobyov V. I. and Gribunin V. G.* Teoriya i praktika vejvlet-preobrazovaniya [Theory and Practice of Wavelet Transform]. S-Petersburg, VUS Publ., 1999. 206 p. (in Russian)
3. *Caire G., Grossman R. L., and Poor H. V.* Wavelet transforms associated with finite cyclic groups. IEEE Trans. Inform. Theory, 1993. vol. 39, no. 4, pp. 1157–1166.
4. *Fekri F., Mersereau R. M., and Schafer R. W.* Theory of wavelet transform over finite fields. Proc. IEEE Intern. Conf. Acoustics, Speech, and Signal Processing, 1999, vol. 3, pp. 1213–1216.
5. *Fekri F., McLaughlin S. W., Mersereau R. M., and Schafer R. W.* Double circulant self-dual codes using finite-field wavelet transforms. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Berlin, Springer, 1999, pp. 355–363.
6. *Fekri F., McLaughlin S. W., Mersereau R. M., and Schafer R. W.* Error control coding using finite-field wavelet transforms. Atlanta, Center for Signal Image Processing, 1999, pp. 1–13.
7. *Fekri F., Mersereau R. M., and Schafer R. W.* Theory of paraunitary filter banks over fields of characteristic two. IEEE Trans. Inform. Theory, 2002. vol. 48, no. 11, pp. 2964–2979.
8. *Chernikov D. V.* Pomekhoustojchivoe kodirovanie s ispol'zovaniem biortogonal'nyh naborov fil'trov tochnogo vosstanovleniya [Error-correcting codes using biorthogonal perfect reconstruction filter banks]. Proc. Conf. " Information Technologies and Systems", Svetlogorsk, 2013, pp. 507–512. (in Russian)
9. *MacWilliams F. G. and Sloane N. J. A.* The Theory of Error Correcting Codes. North-Holland, 1977. 796 p.
10. *Chernikov D. V.* Pomekhoustoychivoye kodirovaniye s ispol'zovaniyem biortogonal'nykh naborov fil'trov [Error-correcting codes using biorthogonal filter banks]. Siberian Electronic Math. Rep., 2015, vol. 12, pp. 704–713. (in Russian)

11. *Soloviev A. A. and Chernikov D. V.* Biorthogonal wavelet codes with prescribed code distance. Discrete Math. Appl., 2018, vol. 28, no. 3 pp. 179–188.
12. *Soloviev A. A. and Chernikov D. V.* Biortogonal'nyye veyvlet kody v polyakh kharakteristiki dva [Biorthogonal wavelet codes in the fields of characteristic two]. Chelyabinsk Physics Math. J., 2017, vol. 2, no. 1, pp. 66–79. (in Russian)
13. *Doubechies I. and Sweldens W.* Factoring wavelet transforms into lifting steps. J. Fourier Analysis and Appl., 1998, vol. 4, no. 3, pp. 247–269.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ИЗОМОРФИЗМА КОНЕЧНЫХ ПОЛУГРУПП[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

Изучается генерическая сложность проблемы изоморфизма конечных полугрупп. В этой проблеме по любым двум полугруппам одинакового порядка, заданным таблицами умножения, требуется определить, являются ли они изоморфными. В. Н. Земляченко, Н. М. Корнеенко и Р. И. Тышкевич в 1982 г. доказали, что к этой проблеме полиномиально сводится проблема изоморфизма конечных графов — известная алгоритмическая проблема, которая активно изучается с 1970-х годов и для которой до сих пор неизвестно полиномиальных алгоритмов. Таким образом, проблема изоморфизма конечных полугрупп с вычислительной точки зрения не проще проблемы изоморфизма графов. Предлагается генерический полиномиальный алгоритм для проблемы изоморфизма конечных полугрупп. В его основе лежит характеризация почти всех конечных полугрупп как 3-нильпотентных полугрупп специального вида, установленная Д. Клейтманом, Б. Ротшильдом и Дж. Спенсером, а также полиномиальный алгоритм Боллобаша, решающий проблему изоморфизма для почти всех сильно разреженных графов.

Ключевые слова: *генерическая сложность, конечные полугруппы, проблема изоморфизма.*

DOI 10.17223/20710410/51/6

ON GENERIC COMPLEXITY OF THE ISOMORPHISM PROBLEM FOR FINITE SEMIGROUPS

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

E-mail: alexander.rybalov@gmail.com

Generic-case approach to algorithmic problems was suggested by A. Miasnikov, V. Kapovich, P. Schupp, and V. Shpilrain in 2003. This approach studies behavior of an algorithm on typical (almost all) inputs and ignores the rest of inputs. In this paper, we study the generic complexity of the isomorphism problem for finite semigroups. In this problem, for any two semigroups of the same order, given by their multiplication tables, it is required to determine whether they are isomorphic. V. Zemlyachenko, N. Korneenko, and R. Tyshkevich in 1982 proved that the graph

[1]Исследование поддержано Программой фундаментальных научных исследований СО РАН I.1.1.4, проект № 0314-2019-0004.

isomorphism problem polynomially reduces to this problem. The graph isomorphism problem is a well-known algorithmic problem that has been actively studied since the 1970s, and for which polynomial algorithms are still unknown. So from a computational point of view the studied problem is no simpler than the graph isomorphism problem. We present a generic polynomial algorithm for the isomorphism problem of finite semigroups. It is based on the characterization of almost all finite semigroups as 3-nilpotent semigroups of a special form, established by D. Kleitman, B. Rothschild, and J. Spencer, as well as the Bollobas polynomial algorithm, which solves the isomorphism problem for almost all strongly sparse graphs.

Keywords: *generic complexity, finite semigroups, isomorphism problem.*

Введение

Понятие изоморфизма является одним из важнейших понятий в современной математике. Изоморфные объекты имеют одинаковые математические свойства и одинаковую математическую структуру, что позволяет абстрагироваться от конкретных представителей этих объектов, однако это также порождает проблему изоморфизма: по двум конкретным представлениям определить, являются ли они изоморфными. Наиболее известная алгоритмическая проблема такого рода — проблема изоморфизма конечных графов. Несмотря на то, что эта проблема находится в центре внимания специалистов с 1970-х годов, до сих пор для неё не найдено полиномиальных алгоритмов. В то же время не доказана её NP-полнота. Таким образом, она может занимать промежуточное положение между проблемами из класса P и NP-полными проблемами. Проблема изоморфизма возникает для многих других конечных алгебраических объектов: групп, полугрупп, колец, полей, алгебр и т. д. Например, для конечных полей эта проблема решается тривиально: известно, что любые два конечных поля одинакового порядка изоморфны. Для конечных полугрупп ситуация гораздо сложнее. Простой алгоритм, который перебирает всевозможные биекции между полугруппами и проверяет, являются ли эти биекции изоморфизмами, работает за экспоненциальное время. Существуют ли полиномиальные алгоритмы решения этой проблемы, неизвестно. В [1] доказано, что к этой проблеме полиномиально сводится проблема изоморфизма конечных графов. Таким образом, проблема изоморфизма конечных полугрупп с вычислительной точки зрения не проще проблемы изоморфизма конечных графов.

В рамках генерического подхода, введённого в [2], алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют так называемое генерическое множество. Понятие «почти все» формализуется введением естественной меры на множестве входных данных. С точки зрения практики алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов.

В [3] предложен первый полиномиальный генерический алгоритм для проблемы изоморфизма графов, который по каждому графу из пары пытается построить *каноническую разметку* вершин — каждой вершине сопоставляет некоторое натуральное число, её уникальный код. Если это удалось сделать для обоих графов, то алгоритм сравнивает полученные два набора канонических разметок: если они совпадают, то графы изоморфны. Более того, эти метки показывают, как задать сам изоморфизм — здесь он будет единственным. Алгоритм Бабаи — Эрдеша — Селкова [3] может не найти каноническую разметку для некоторых графов, но доля (вероятность) таких графов среди всех графов размера n ограничена $O(1/\sqrt[7]{n})$. В дальнейшем было предложено много эффективных алгоритмов для решения проблемы изоморфизма графов в раз-

личных моделях случайных графов. Б. Боллобаш в [4] предложил полиномиальный алгоритм, решающий эту проблему для почти всех сильно разреженных графов, в которых для графов с n вершинами вероятность $p(n)$ того, что две вершины соединены ребром, удовлетворяет условию $\dfrac{C_1 \ln n}{n} \leqslant p(n) \leqslant \dfrac{C_2}{n^{11/12}}$, где $C_1 > 1$, $0 < C_2 < 1$ — любые константы. Так же как и алгоритм Бабаи — Эрдеша — Селкова, алгоритм Боллобаша пытается сначала построить канонические разметки вершин обоих графов, по которым затем ищет изоморфизм (единственный) между графами, если он существует. Вероятность того, что каноническая разметка не будет найдена для одного графа, оценивается числом n^{-C}, где $C > 0$ — некоторая константа.

В данной работе предлагается генерический полиномиальный алгоритм для проблемы изоморфизма конечных полугрупп. В его основе лежит характеризация почти всех конечных полугрупп как 3-нильпотентных полугрупп специального вида, установленная в [5], а также упомянутый полиномиальный алгоритм Боллобаша для почти всех сильно разреженных графов из [4].

1. Генерические алгоритмы

Пусть I — некоторое множество входов. Для подмножества $S \subseteq I$ определим *последовательность относительных плотностей*

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n — множество входов размера n; $S_n = S \cap I_n$. Заметим, что $\rho_n(S)$ — это вероятность попасть в S при случайной и равновероятной генерации входов из I_n.

Асимптотической плотностью множества S назовём верхний предел

$$\rho(S) = \overline{\lim_{n\to\infty}} \rho_n(S).$$

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 $\{x \in I : \mathcal{A}(x) \neq ?\}$ является генерическим.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если

$$\forall x \in I\ \big((\mathcal{A}(x) = y \in J) \Rightarrow (f(x) = y)\big).$$

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). Множество $S \subseteq I$ называется *генерически разрешимым за полиномиальное время*, если существует генерический полиномиальный алгоритм, вычисляющий его характеристическую функцию.

2. Конечные полугруппы

Для определённости будем рассматривать конечные полугруппы с элементами из множеств $\{1, 2, \ldots, n\}$, $n \in \mathbb{N}$. *Таблицей умножения* конечной полугруппы S порядка n

называется таблица $n \times n$, в которой на месте (i, j) стоит результат произведения элементов i и j.

Полугруппы S_1 и S_2 называются *изоморфными*, если существует биекция $\varphi : S_1 \to \to S_2$, такая, что для любых элементов $a, b \in S_1$ имеет место $\varphi(ab) = \varphi(a)\varphi(b)$. Биекция φ называется *изоморфизмом*. *Проблема изоморфизма конечных полугрупп* состоит в следующем. Даны две конечные полугруппы S_1 и S_2 одинакового порядка, заданные таблицами умножения. Определить, являются ли они изоморфными.

Будем говорить, что конечная полугруппа S с основным множеством $\{1, 2, \ldots, n\}$ имеет вид $H(A, \psi, z)$, где $A \subset \{1, 2, \ldots, n\}$, $\psi : A \times A \to \bar{A}$ и $z \in \bar{A}$ (через $\bar{A}$ обозначено дополнение к A), если операция умножения в S определена следующим образом:

$$xy = \begin{cases} \psi(x, y), & \text{если } x, y \in A, \\ z & \text{иначе.} \end{cases}$$

Легко проверяется, что определённая таким образом операция умножения обладает свойством ассоциативности. Элемент z играет роль нуля. Любая полугруппа вида $H(A, \psi, z)$ является 3-нильпотентной. В [5] доказано, что почти все конечные полугруппы являются 3-нильпотентными, более того, имеют вид $H(A, \psi, z)$. Точнее, пусть $\mathcal{S}$ — множество всех конечных полугрупп, $\mathcal{NS}$ — 3-нильпотентные полугруппы вида $H(A, \psi, z)$. Тогда множество $\mathcal{NS}$ является генерическим в $\mathcal{S}$. Доказано также, что почти все полугруппы достаточно большого порядка n имеют вид $H(A, \psi, z)$, где $|A|$ принадлежит интервалу $I_n = (t_0 - 3, t_0 + 3)$, $t_0 = n - n/(2 \ln n)$. Таким образом, если $\mathcal{NS}_n$ — 3-нильпотентные полугруппы вида $H(A, \psi, z)$ порядка n, а $\mathcal{NS}(k)_n$ — полугруппы в $\mathcal{NS}_n$ вида $H(A, \psi, z)$, где $|A| = k$, то множество

$$\bigcup_{n=1}^{\infty} \bigcup_{k \in I_n} \mathcal{NS}(k)_n$$

является генерическим в $\mathcal{S}$.

Заметим, что представление конечной полугруппы в виде $H(A, \psi, z)$ может быть неоднозначным. Это происходит, если $A_1 \subset A_2$ и $\psi(a, b) = z$, когда $a \in A_2 \setminus A_1$ или $b \in A_2 \setminus A_1$. Будем называть представление полугруппы $S = H(A, \psi, z)$ *минимальным*, если не существует множества A', такого, что $S = H(A', \psi', z')$ и $|A'| < |A|$. Обозначим через $\mathcal{NS}'$ множество всех полугрупп из $\mathcal{NS}$ с минимальным представлением, а через $\mathcal{NS}'(k)_n$ — множество всех полугрупп из $\mathcal{NS}(k)_n$ с минимальным представлением. В [5] доказано, что $\mathcal{NS}'$ также является генерическим в $\mathcal{S}$; более того, для любого достаточно большого n и любого $k < n$ имеет место

$$|\mathcal{NS}(k)_n| = (1 + o(1))|\mathcal{NS}'(k)_n|.$$

Отсюда следует, что множество

$$\bigcup_{n=1}^{\infty} \bigcup_{k \in I_n} \mathcal{NS}'(k)_n$$

также является генерическим в $\mathcal{S}$ и

$$\lim_{n \to \infty,\ k \in I_n} \frac{|\mathcal{NS}(k)_n|}{|\mathcal{NS}'(k)_n|} = 1.$$

Лемма 1. Существует генерический полиномиальный алгоритм $\mathcal{A}$, который по произвольной полугруппе S, заданной таблицей умножения, получает её минимальное представление в виде $H(A, \psi, z)$ (если это возможно).

Доказательство. Полиномиальный алгоритм $\mathcal{A}$ работает на полугруппе S следующим образом:

1) Ищет по таблице умножения нуль: такой элемент z, что $za = z$ и $az = z$ для любого $a \in S$.
2) Если нуля нет, выдаёт ответ «?».
3) Ищет все элементы $B = \{b_1, \ldots, b_m\}$, такие, что $ab_i = z$ и $b_i a = z$ для любого $a \in S$. Это гарантирует минимальность представления.
4) Полагает $A = S \setminus (B \cup \{z\})$ и проверяет, верно ли, что для любых $a_1, a_2 \in A$ имеет место $a_1 a_2 \in B$.
5) Если проверка на шаге 4 пройдена, выдаёт множество A, часть таблицы умножения для множества A как задание функции ψ и элемент z как нуль.
6) Если проверка на шаге 4 не пройдена, то выдаёт ответ «?».

Генеричность этого алгоритма следует из результатов Д. Клейтмана, Б. Ротшильда, Дж. Спенсера [5]. ■

Лемма 2. Пусть $S_1 = H(A_1, \psi_1, z_1)$ и $S_2 = H(A_2, \psi_2, z_2)$ — изоморфные полугруппы с минимальными представлениями и φ — изоморфизм между ними. Тогда

1) $\varphi(z_1) = z_2$;
2) ограничение φ на A_1 есть биекция τ между A_1 и A_2, откуда $|A_1| = |A_2|$;
3) ограничение φ на $S_1 \setminus (A_1 \cup \{z_1\})$ есть биекция π между $S_1 \setminus (A_1 \cup \{z_1\})$ и $S_2 \setminus (A_2 \cup \{z_2\})$;
4) биекция π может быть найдена за полиномиальное время по биекции τ.

Доказательство.

1) Для любого $s \in S_1$ имеет место $sz_1 = z_1$, откуда $\varphi(z_1) = \varphi(sz_1) = \varphi(s)\varphi(z_1)$. Так как $\varphi(s)$ при этом пробегает всю полугруппу S_2, элемент $\varphi(z_1)$ является нулем в S_2, то есть $\varphi(z_1) = z_2$.

2) Легко видеть, что для минимального представления $S = H(A, \psi, z)$ имеют место следующие утверждения:

а) $\forall a \in A\ \exists b \in A\ (ab \neq z$ или $ba \neq z)$;

б) $\forall b \notin A\ \forall c \in S\ (bc = cb = z)$.

Из этого следует, что $\varphi(a) \in A_2$ для любого $a \in A_1$, т. е. ограничение φ на A_1 есть биекция τ между A_1 и A_2.

3) Следует из п. 2.

4) Пусть известна биекция τ, являющаяся ограничением некоторого изоморфизма φ между S_1 и S_2 на множество A_1. Обозначим $B_1 = S_1 \setminus (A_1 \cup \{z_1\})$ и $B_2 = S_2 \setminus (A_2 \cup \{z_2\})$. Нужно найти биекцию π между B_1 и B_2, которая является ограничением φ на B_1.

Пусть $\mathrm{Img}(\psi_1) \subseteq S_1 \setminus A_1$ есть образ функции ψ_1. Тогда для любого $b \in B_1$, такого, что $b \in \mathrm{Img}(\psi_1)$, найдутся $a_1, a_2 \in A_1$, такие, что $a_1 a_2 = b$. Поэтому

$$\pi(b) = \varphi(a_1 a_2) = \varphi(a_1)\varphi(a_2) = \tau(a_1)\tau(a_2) \in B_2.$$

Таких возможных пар не более, чем квадратичное число от порядка полугрупп, поэтому такой поиск выполняется за полиномиальное время.

Остаётся заметить, что в качестве ограничения искомой биекции π между $B_1 \setminus \mathrm{Img}(\psi_1)$ и $B_2 \setminus \pi(\mathrm{Img}(\psi_1))$ можно выбрать произвольную биекцию между этими множествами. ■

3. Конечные полугруппы и графы

Граф G — это пара (V, E), где V — множество вершин; $E \subseteq V \times V$ — множество рёбер графа G. Таким образом, будем рассматривать ориентированные графы с петлями. Под размером графа $G = (V, E)$ будем понимать число вершин $|V|$. Два графа $G = (V_1, E_1)$ и $H = (V_2, E_2)$ называются *изоморфными*, если $|V_1| = |V_2|$, $|E_1| = |E_2|$ и существует взаимно однозначная функция $f : V_1 \to V_2$, такая, что $(u, v) \in E_1$ тогда и только тогда, когда $(f(u), f(v)) \in E_2$. *Проблема изоморфизма графов* состоит в следующем: заданы два графа G_1 и G_2 одинакового размера, требуется определить, являются ли они изоморфными.

Для любой полугруппы $S = H(A, \psi, z)$ определим граф $G(S)$ следующим образом. Вершины графа $G(S)$ — это множество A. Для вершин $a_1, a_2 \in A$ в графе $G(S)$ есть ребро (a_1, a_2) тогда и только тогда, когда $a_1 a_2 = z$.

Лемма 3. Пусть полугруппы $S_1 = H(A_1, \psi_1, z_1)$ и $S_2 = H(A_2, \psi_2, z_2)$ с минимальными представлениями изоморфны. Тогда изоморфны и графы $G(S_1)$ и $G(S_2)$.

Доказательство. Обозначим через φ изоморфизм между полугруппами S_1 и S_2. По лемме 2 $|A_1| = |A_2|$ и ограничение φ на A_1 является биекцией. Докажем, что ограничение φ на A_1 является изоморфизмом между графами $G(S_1)$ и $G(S_2)$. Действительно, так как $\varphi(z_1) = z_2$, то для любых $a_i, a_j \in A_1$ в графе $G(S_1)$ есть ребро $(a_i, a_j) \Leftrightarrow$ $\Leftrightarrow a_i a_j = z_1 \Leftrightarrow \varphi(a_i a_j) = z_2 \Leftrightarrow \varphi(a_i)\varphi(a_j) = z_2 \Leftrightarrow$ есть ребро $(\varphi(a_i), \varphi(a_j))$ в графе $G(S_2)$. ■

Пусть $\mathcal{G}$ — множество графов, для которых алгоритм Боллобаша не находит канонической разметки. Определим множество

$$\mathcal{R}(k)_n = \{S \in \mathcal{NS}(k)_n : G(S) \in \mathcal{G}\}.$$

Лемма 4. Существуют константы $C_1, C_2 > 0$, такие, что для любого достаточно большого n и любого $k \in (t_0 - 3, t_0 + 3)$, где $t_0 = n - \dfrac{n}{2 \ln n}$, имеет место

$$\rho_n(\mathcal{R}(k)_n) = \frac{|\mathcal{R}(k)_n|}{|\mathcal{NS}(k)_n|} < \frac{C_1}{n^{C_2}}.$$

Доказательство. Заметим, что

$$\mathcal{NS}(k)_n = \bigcup_{A \subset \{1,\ldots,n\},\, |A| = k} \mathcal{NS}(k, A)_n,$$

где $\mathcal{NS}(k, A)_n$ — множество всех полугрупп вида $H(A, \psi, z)$, A — k-элементное подмножество $\{1, \ldots, n\}$. Легко видеть, что $|\mathcal{NS}(k, A_1)_n| = |\mathcal{NS}(k, A_2)_n|$ для любых k-элементных подмножеств A_1 и A_2 множества $\{1, \ldots, n\}$. Число таких $\mathcal{NS}(k, A)_n$ равно C_n^k. Определим для любого k-элементного подмножества $A \subset \{1, \ldots, n\}$

$$\mathcal{R}(k, A)_n = \mathcal{R}(k)_n \cap \mathcal{NS}(k, A)_n.$$

Заметим, что

$$|\mathcal{R}(k, A_1)_n| = |\mathcal{R}(k, A_2)_n|$$

для всех k-элементных подмножеств A_1 и A_2 множества $\{1,\dots,n\}$, так как любая биекция между A_1 и A_2 задаёт биекцию между сравниваемыми множествами. Число таких множеств равно C_n^k. Поэтому для любого фиксированного подмножества A имеем $\rho_n(\mathcal{R}(k,A)_n) = \rho_n(\mathcal{R}(k)_n)$.

Зафиксируем k-элементное подмножество $A \subset \{1,\dots,n\}$ и отметим, что $\rho_n(\mathcal{R}(k,A)_n)$ есть вероятность того, что для случайно и равновероятно выбранной из $\mathcal{NS}(k,A)_n$ полугруппы S алгоритм Боллобаша не выдаёт каноническую разметку для графа $G(S)$. Докажем, что вероятность быть соединёнными ребром для вершин случайного графа $G(S)$ не зависит от этих вершин, и оценим эту вероятность.

Полугруппа S имеет вид $H(A,\psi,z)$, где ψ — случайно и равновероятно выбранная функция из $A\times A$ в $\bar{A}$; z — случайно и равновероятно выбранный элемент из $\bar{A}$. Пусть $A=\{a_1,\dots,a_k\}$ и $S\setminus A=\{b_1,b_2,\dots,b_{n-k}\}$. Вершинами графа $G(S)$ является множество A. Вершины a_i и a_j соединены ребром, если $\psi(a_i,a_j)=z$. Так как функция ψ выбирается случайно и равновероятно, вероятность этого для любых i,j равна $p=1/(n-k)$. Так как

$$n-\frac{n}{1{,}99\ln n}<n-\frac{n}{2\ln n}-3<k<n-\frac{n}{2\ln n}+3<n-\frac{n}{2{,}01\ln n},$$

то

$$\frac{1{,}99\ln n}{n}<p<\frac{2{,}01\ln n}{n}.$$

Учитывая, что функция $f(x)=\ln x/x$ при $x>e$ убывает и что для достаточно больших n имеет место $n<1{,}01\,k$, получаем

$$\frac{1{,}99\ln(1{,}01\,k)}{1{,}01\,k}<\frac{1{,}99\ln n}{n}<p<\frac{2{,}01\ln n}{n}<\frac{2{,}01\ln k}{k}<\frac{0{,}5}{k^{11/12}}.$$

Отсюда

$$\frac{1{,}97\ln k}{k}<p<\frac{0{,}5}{k^{11/12}}.$$

Поэтому вероятность того, что алгоритм Боллобаша не выдаёт каноническую разметку для графа $G(S)$, для достаточно больших k оценивается, согласно [4], величиной $1/k^C$ с некоторой универсальной константой $C>0$. Следовательно,

$$\rho_n(\mathcal{R}(k)_n)=\rho_n(\mathcal{R}(k,A)_n)<\frac{1}{k^C}<\frac{C_1}{n^C}.$$

Последнее неравенство верно в силу того, что $0{,}99\,n<k$ при достаточно больших n. ■

4. Основной результат

Теорема 1. Проблема изоморфизма конечных полугрупп генерически разрешима за полиномиальное время.

Доказательство. Пусть S_1 и S_2 — две полугруппы порядка n, заданные таблицами умножения. Полиномиальный генерический алгоритм $\mathcal{B}$, решающий проблему изоморфизма, работает на входе (S_1,S_2) следующим образом:

1) Запустить полиномиальный генерический алгоритм $\mathcal{A}$ из леммы 1 на S_1 и S_2.
2) Если $\mathcal{A}(S_1)=?$ или $\mathcal{A}(S_2)=?$, то выдать ответ «?».
3) Теперь имеются минимальные представления полугрупп $S_1=H(A_1,\psi_1,z_1)$ и $S_2=H(A_2,\psi_2,z_2)$.

4) Если $|A_1| \neq |A_2|$, то выдать ответ «НЕТ». Корректность этого ответа следует из леммы 2.
5) Пусть $k = |A_1|$. Если $k \notin \left\{n - \dfrac{n}{2\ln n} - 3, \ldots, n - \dfrac{n}{2\ln n} + 3\right\}$, то выдать ответ «?».
6) Построить графы $G(S_1)$ и $G(S_2)$.
7) Запустить алгоритм Боллобаша для проверки изоморфизма графов $G(S_1)$ и $G(S_2)$.
8) Если проверка закончилась неудачей, выдать ответ «?».
9) Если алгоритм Боллобаша выдал «НЕТ», выдать «НЕТ». Корректность ответа в данном случае следует из леммы 3.
10) Если алгоритм Боллобаша выдал «ДА» и выдал изоморфизм φ между графами $G(S_1)$ и $G(S_2)$, то проверить, является ли изоморфизмом между полугруппами S_1 и S_2 отображение φ', определённое следующим образом:

$$\varphi'(a) = \begin{cases} \varphi(a), & \text{если } a \in A_1, \\ z_2, & \text{если } a = z_1, \\ \pi(a), & \text{если } a \in S_1 \setminus (A_1 \cup \{z_1\}). \end{cases}$$

Здесь π — биекция между $S_1 \setminus (A_1 \cup \{z_1\})$ и $S_2 \setminus (A_2 \cup \{z_2\})$, построенная по ограничению φ на A_1 из п. 4 леммы 2. Если является, то выдать ответ «ДА», иначе — «НЕТ». Корректность ответа также следует из леммы 3.

Докажем генеричность алгоритма $\mathcal{B}$. Алгоритм может выдать ответ «?» на шагах 2, 5 и 8. Пренебрежимость множества входов, на которых алгоритм выдаёт ответ «?» на шагах 2 и 5, следует из леммы 1 и результатов Клейтмана — Ротшильда — Спенсера. На шаге 8 алгоритм выдаёт ответ «?», если алгоритм Боллобаша не смог найти каноническую разметку для графа $G(S_1)$ или $G(S_2)$. Напомним, что через $\mathcal{G}$ обозначается множество графов, для которых алгоритм Боллобаша не выдаёт каноническую разметку. Нужно доказать, что

$$\lim_{n\to\infty,\ k\in I_n} \frac{|\mathcal{R}'(k)_n|}{|\mathcal{NS}'(k)_n|} = 0,$$

где $I_n = \left\{n - \dfrac{n}{2\ln n} - 3, \ldots, n - \dfrac{n}{2\ln n} + 3\right\}$ и $\mathcal{R}'(k)_n = \{S \in \mathcal{NS}'(k)_n : G(S) \in \mathcal{G}\}$. Так как $\mathcal{R}'(k)_n \subset \mathcal{R}(k)_n$ и, согласно [5], имеет место

$$\lim_{n\to\infty,\ k\in I_n} \frac{|\mathcal{NS}(k)_n|}{|\mathcal{NS}'(k)_n|} = 1,$$

то достаточно доказать, что

$$\lim_{n\to\infty,\ k\in I_n} \frac{|\mathcal{R}(k)_n|}{|\mathcal{NS}(k)_n|} = 0.$$

Это следует из леммы 4, по которой для всех $k \in I_n$ выполняется неравенство

$$\frac{|\mathcal{R}(k)_n|}{|\mathcal{NS}(k)_n|} < \frac{C_1}{n_2^C}.$$

Теорема 1 доказана. ■

Автор выражает благодарность рецензенту за полезные замечания и предложения по улучшению текста статьи.

ЛИТЕРАТУРА

1. *Земляченко В. Н., Корнеенко Н. М., Тышкевич Р. И.* Проблема изоморфизма графов // Записки научных семинаров ЛОМИ. 1982. Т. 118. С. 83–158.
2. *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.
3. *Babai L., Erdos P, and Selkow S.* Random graph isomorphism // SIAM J. Computing. 1980. V. 9. No. 3. P. 628–635.
4. *Bollobas B.* Distinguishing of vertices of random graphs // Ann. Discrete Math. 1982. V. 13. P. 33–50.
5. *Kleitman D. J., Rothschild B. R., and Spencer J. H.* The number of semigroups of order n // Proc. Amer. Math. Soc. 1976. V. 55. No. 1. P. 227–232.

REFERENCES

1. *Zemlyachenko V. N., Korneenko N. M., and Tyshkevich R. I.* The graph isomorphism problem [Problema izomorfizma grafov]. Zapiski Nauchnyh Seminarov LOMI, 1982, vol. 118, pp. 83–158. (in Russian)
2. *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.
3. *Babai L., Erdos P, and Selkow S.* Random graph isomorphism. SIAM J. Computing, 1980, vol. 9, no. 3, pp. 628–635.
4. *Bollobas B.* Distinguishing of vertices of random graphs. Ann. Discrete Math., 1982, vol. 13, pp. 33–50.
5. *Kleitman D. J., Rothschild B. R., and Spencer J. H.* The number of semigroups of order n. Proc. Amer. Math. Soc., 1976, vol. 55, no. 1, pp. 227–232.

СВЕДЕНИЯ ОБ АВТОРАХ

ГАМАЮНОВ Денис Юрьевич — кандидат физико-математических наук, старший научный сотрудник факультета ВМК МГУ имени М. В. Ломоносова, г. Москва.
E-mail: **gamajun@seclab.cs.msu.su**

КИРШАНОВА Елена Алексеевна — Ph.D., доцент ИФМНиИТ, и. о. зав. лабораторией «Мат. методы защиты и обработки информации», БФУ им. И.Канта, г. Калининград. E-mail: **ekirshanova@kantiana.ru**

ЛИТИЧЕВСКИЙ Дмитрий Владимирович — аспирант Челябинского государственного университета, г. Челябинск. E-mail: **litichevskiydv@gmail.com**

МАЛЫГИНА Екатерина Сергеевна — кандидат физико-математических наук, доцент ИФМНиИТ, младший научный сотрудник лаборатории «Мат. методы защиты и обработки информации», БФУ им. И.Канта, г. Калининград.
E-mail: **emalygina@kantiana.ru**

МАРТЫНЕНКОВ Игорь Владимирович — Астраханский государственный технический университет, г. Астрахань. E-mail: **mivpost@yandex.ru**

НОВОСЕЛОВ Семен Александрович — старший преподаватель ИФМНиИТ, младший научный сотрудник лаборатории «Мат. методы защиты и обработки информации», БФУ им. И.Канта, г. Калининград. E-mail: **snovoselov@kantiana.ru**

ОЛЕФИРЕНКО Денис Олегович — аспирант ИФМНиИТ БФУ им. И.Канта, г. Калининград. E-mail: **dolefirenko@kantiana.ru**

ПОПКОВ Кирилл Андреевич — кандидат физико-математических наук, научный сотрудник Института прикладной математики им. М. В. Келдыша РАН, г. Москва.
E-mail: **kirill-formulist@mail.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**

ШЛЯХТИНА Елена Анатольевна — студентка факультета ВМК МГУ имени М. В. Ломоносова, г. Москва. E-mail: **ea.shlyakhtina@gmail.com**

Журнал «Прикладная дискретная математика» входит в перечень ВАК рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата и доктора наук по специальностям 01.01.06 — «Математическая логика, алгебра и теория чисел», 01.01.09 — «Дискретная математика и математическая кибернетика», 05.13.11 — «Математическое и программное обеспечение вычислительных машин,комплексов и компьютерных сетей», 05.13.17 — «Теоретические основы информатики», 05.13.19 — «Методы и системы защиты информации, информационная безопасность», а также в перечень журналов, рекомендованных ФУМО ВО ИБ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал индексируется в базах данных Web of Science (Emerging Sources Citation Index (ESCI) и Russian Science Citation Index (RSCI)), Scopus, MathSciNet и Zentralblatt MATH.

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте journals.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также правила подготовки рукописей статей для публикации в журнале.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*