Учреждение Российской академии наук
Санкт-Петербургское отделение Математического
института им. В. А. Стеклова РАН

На правах рукописи

Сергей Игоревич Николенко

Новые конструкции криптографических примитивов, основанные на полугруппах, группах и линейной алгебре

(01.01.06 — математическая логика, алгебра и теория чисел)

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук,
ведущий научный сотрудник ПОМИ РАН
Эдуард Алексеевич Гирш

Санкт-Петербург
— 2008 —

Оглавление

1 Введение **4**

1.1 Криптография как раздел информатики 4

1.2 Модели вычислений 8

1.3 Основные определения современной криптографии . . . 14

2 Новые конструкции полных односторонних функций **20**

2.1 Введение 20

2.2 Задача достижимости с распределением для полусистем Туэ 24

2.3 Задача Поста 26

2.4 Полная односторонняя функция, основанная на задаче поиска замощения 28

2.5 Полная односторонняя функция, основанная на полусистемах Туэ 32

2.6 Полная односторонняя функция на базе задачи Поста . 36

2.7 Полные односторонние функции и DistNP-трудные комбинаторные задачи 39

2.8 Полные односторонние функции в большей общности . . 42

3 Новые конструкции криптографических примитивов, доказуемо надёжных в слабом смысле **46**

3.1 Введение 46

3.2 Определения 48

3.3 Матрицы сложных функций 52

3.4 Исключение гейтов 57

3.5 Две конструкции 67

3.6 Семейство функций с секретом, надёжных в слабом смысле 69
3.7 Функция с секретом с экспоненциальной гарантией надёжности 72

4 Новые алгебраические конструкции криптографических примитивов 75
4.1 Алгебраическая криптография 75
4.2 Ослабленные результаты современной криптографии . . 79
4.3 Доказуемый взлом 81
4.4 Определения 83
4.5 Криптосистемы, основанные на инвариантах групп, и их доказуемый взлом 85
4.6 Дерево групп 90
4.7 Листья дерева 95
4.8 Атаки на криптосистемы, основанные на инвариантах . . 99
4.9 Схема согласования ключа Аншель-Аншеля-Голдфельда и её устойчивость против взлома с доказательством . . . 102

Литература 105

Глава 1

Введение

1.1 Криптография как раздел информатики

Классическая криптография, понимаемая как способ передать сообщение так, чтобы противник не сумел его расшифровать, применялась, наверное, с тех самых пор, как люди впервые задумались о том, как передавать друг другу сообщения. Различного рода шифры, в том числе весьма изящные, применялись ещё в античности. Древние греки (точнее, спартанцы) использовали так называемые *скитале́ы*: цилиндры, на которые наматывались узкие полоски пергамента. Затем текст писали на полоске поперёк цилиндра; получался перестановочный шифр, для декодирования которого нужно было снова намотать пергамент на цилиндр такой же толщины [78]. Юлий Цезарь считается изобретателем *шифра Цезаря*, в котором каждый символ алфавита заменяется на другой, отстоящий от него в алфавите на некоторое фиксированное число позиций (смещение). Краткое руководство по использованию шифров для обмена любовными посланиями содержит даже «Камасутра».

Довольно давно появились и первые работы, направленные на взлом шифров. Разумеется, если кто-то что-то от кого-то скрывает, значит, это может иметь ценность, и эту ценность порой можно было добыть успешной дешифровкой. Такие, сугубо прикладные работы, конечно, велись с самого появления шифров. Однако появлялись и теоретические исследования. Например, и шифр Цезаря, и большинство других кодов и шифров, использовавшихся в античности и средние века, не были устойчивы против метода *частотного анализа*, при котором наиболее вероятная расшифровка того или иного сим-

вола вычисляется исходя из частоты встречаемости этого символа в закодированном тексте (при известных частотах появления букв алфавита в среднестатистическом тексте на данном языке). Этот метод, судя по всему, впервые рассмотрели Абу Юсуф аль-Кинди и другие арабские учёные (для арабского же языка, разумеется) в IX-X вв. нашей эры [5]. В качестве источников по истории классической криптографии можно порекомендовать [14, 74, 117].

Современная криптография, которую рассматривают как раздел информатики, — очень молодая наука. Она настолько молода, что даже если начинать изложение её истории раньше её «официального» появления, всё равно за рамки XX века выйти не получится. К началу века относится первый серьёзный теоретический успех: конструкция абсолютно стойкого шифра, так называемой «схемы одноразовых блокнотов» (one-time pad), разработанной Гильбертом Вернамом и Клодом Моборном [129]. В этой схеме с закрытым ключом в качестве шифра транслируется побитовая сумма (XOR) кодируемого сообщения и случайной битовой строки. При условии надёжной передачи секретного ключа (это, конечно, самое уязвимое место) схему одноразовых блокнотов взломать невозможно [113]; это первый код с таким свойством.

Появление современной криптографии было в значительной степени мотивировано военными нуждами: во время Второй мировой войны надёжная и защищённая связь была крайне ценна. Значительная часть того, что происходило в Блетчли-парке и аналогичных учреждениях, относилась, конечно, ко взлому и разработке конкретных шифров. Но шла и теоретическая работа. Первым математически полным и строгим изложением теории кодирования следует, видимо, считать послевоенные работы Клода Шеннона по теории информации [112, 113]. Шеннон заложил математическую базу для криптографических ис-

следований, однако в течение почти тридцати последующих лет исследования в этом направлении либо не проводились вовсе, либо были засекречены.

Прорыв, который мы склонны считать настоящим началом криптографии как раздела информатики, произошёл в середине 1970-х гг. Во-первых, в 1975 году появился первый настоящий криптографический стандарт — DES, Data Encryption Standard — разработанный в IBM и предлагаемый банкам и другим финансовым организациям для обмена секретными данными [43]. DES обладал весьма высокой надёжностью, его криптоанализ не выявил серьёзных дефектов, и взломать DES стало возможным только тогда, когда вычислительных мощностей оказалось достаточно для пусть улучшенных, но всё же атак «грубой силой» [21, 34, 40].

А главное — в 1976 году появилась работа Уитфилда Диффи и Мартина Хеллмана «New directions in cryptography» [39], в которой была впервые предложена конструкция криптографического примитива (в данном случае — протокола согласования ключа), который полагался не на надёжность передачи некоторого секретного ключа или алгоритма дешифровки, как все его предшественники, а на вычислительную сложность решения некоторой задачи, в данном случае — дискретного логарифма (вскоре появился и соответствующий патент, включающий в число авторов оказавшего большое влияние на становление криптографии Ральфа Меркле [68]). В протоколе Диффи-Хеллмана участники (их традиционно называют Алиса и Боб) сначала договариваются (открыто) о том, по какому простому модулю p проводить вычисления и какой первообразный корень g по модулю p использовать. Затем Алиса секретно выбирает натуральное число a и открыто пересылает $g^a \bmod p$. Боб секретно выбирает натуральное число b и открыто пересылает $g^b \bmod p$. В результате у участников

протокола образуется общий секрет $g^{ab} \bmod p$, который они могут использовать, например, в криптографических протоколах с секретным ключом. А противнику, для того чтобы получить этот секрет, нужно по g, g^a и g^b восстановить g^{ab}; эта задача считается вычислительно сложной.

Иными словами, задача *построения* общего ключа является в этой конструкции достаточно простой, а вот задача *взлома*, восстановления этого ключа на основе общедоступной информации, хотя и является алгоритмически разрешимой, имеет (предположительно) высокую сложность. Именно на разницу между вычислительной сложностью кодирования и декодирования и полагается пользователь такой криптографической конструкции.

Эта работа была вскоре продолжена, и появились новые криптографические примитивы, основанные на вычислительно сложных задачах. В 1978 году Ривест, Шамир и Эйдельман предложили криптосистему с открытым ключом RSA [107], которая до сих пор остаётся одним из наиболее успешных примеров таких криптосистем. В криптосистеме с открытым ключом Боб должен передать Алисе сообщение, закодировав его при помощи своего публичного ключа так, чтобы Алиса смогла его дешифровать при помощи своего секретного ключа. Пару (секретный ключ, публичный ключ) генерирует Алиса перед началом обмена сообщениями. В протоколе RSA ключи порождаются следующим образом: Алиса выбирает два простых числа p и q, вычисляет $n = pq$ и $\phi(n) = (p-1)(q-1)$, а затем выбирает число $1 < e < \phi(n)$, взаимно простое с $\phi(n)$, и вычисляет такое d, что $de \equiv 1 \pmod{\phi(n)}$. Затем Алиса передаёт в качестве публичного ключа пару чисел (n, e). Боб, желая передать число $m < n$, кодирует его как $c = m^e \bmod n$ и открыто пересылает (чтобы избежать атак, работающих в частных случаях, Боб должен передавать в качестве m не

своё сообщение, а его версию, модифицированную при помощи одной из так называемых padding schemes). Алиса теперь может расшифровать это сообщение, вычислив его как $m = c^d \bmod n$, так как, по малой теореме Ферма,

$$c^d = (m^e)^d = m^{ed} \equiv m \pmod{n}.$$

Задача взломщика в этой системе была бы простой, если бы число n можно было эффективно разложить на множители; стоит отметить, что обратное неизвестно: взаимоотношения между факторизацией и задачей взлома RSA до конца ещё не установлены [27, 29, 106]. Уже были разработаны две успешные атаки на RSA, причём обе в том или ином смысле использовали принцип «одним махом перебрать все возможные делители»; одна из них проводится в так называемой unit-cost модели, где разрешено за один шаг делать арифметические операции над числами произвольной длины [110], а другая работает на квантовых компьютерах [116]; однако в классической модели успешно раскладывать числа на множители или решать задачу RSA пока не умеет никто.

Основными источниками по базовым определениям и конструкциям современной криптографии можно считать [50–53, 77, 89].

1.2 Модели вычислений

Прежде чем давать определения, следует определить модель вычислений, в которой мы будем работать. Долгое время модель вычислений была единственной и естественной, основанной на работах Тьюринга и Поста [99, 101, 127], которые построили базовые конструкции, эквивалентные друг другу и предположительно реализующие все возможные алгоритмы. Это предположение получило известность как *тезис Чёрча-Тьюринга*; тезис, выдвинутый самим Чёрчем, касался общерекурсивных функций и функций, вычислимых при по-

мощи λ-исчисления Чёрча [31, 33, 127]. Поэтому классическая теория сложности изучает алгоритмы, реализуемые на машине Тьюринга [12, 49, 61, 71, 96, 119, 162].

Однако в последние годы значительно возрос интерес к другой модели вычислений, *квантовым вычислениям*, которые используют схемы, построенные из непрерывных унитарных отображений, производящихся над квантовыми системами [19, 41, 47, 73, 86, 94, 141, 149]. Эта модель не предоставляет принципиально новых вычислимых функций, но вполне возможно, что в ней некоторые задачи можно решить быстрее, чем в классической модели. В частности, популярность квантовой модели вычислений началась с *алгоритма Шора*, которым можно решить задачу разложения числа на простые множители и задачу дискретного логарифма [116]. Таким образом, и криптосистема RSA, и протокол согласования ключа Диффи-Хеллмана перестают быть надёжными, если противник может использовать квантовые компьютеры. Конечно, квантовые компьютеры не всемогущи — например, NP-трудные задачи вряд ли можно будет решить на квантовых компьютерах за полиномиальное время [16]. Но наиболее популярные задачи классической криптографии, такие, как RSA, квантово решаются достаточно эффективно. Таким образом, оказалось, что для построения криптографических примитивов в квантовой модели вычислений требуются другие, более устойчивые конструкции. Это привело к развитию так называемой *квантовой криптографии*, которая использует квантовые эффекты для согласования ключа и построения других примитивов [17, 18].

В частности, именно квантовые вычисления и алгоритм Шора стали одной из главных мотиваций для исследования некоммутативных криптографических примитивов [8–10, 15, 93, 95, 102, 148]. В то время как алгоритм Шора решает задачу факторизации и дискретного

логарифма в абелевых группах [79, 149], о некоммутативных примитивах ничего подобного не известно. Новые конструкции таких примитивов, описанные в главе 4, являются одним из основных результатов настоящей работы.

В главе 4 мы будем исследовать некоммутативные схемы, но касаться квантовой криптографии как таковой не будем. Поэтому здесь мы зафиксируем классическую модель вычислений и приведём лишь базовые классические определения. В определении машины Тьюринга мы следуем классическим источникам [71, 162].

Определение 1.1 *Машина Тьюринга* — это упорядоченная семёрка $M = \langle Q, \Gamma, B, \Sigma, \pi, s, H \rangle$, где

- Q — конечное множество *состояний* машины Тьюринга;
- Γ — конечный алфавит *символов ленты*;
- $B \in \Gamma$ — пустой символ (единственный символ, который может встречаться на ленте бесконечное число раз);
- $\Sigma \subseteq \Gamma \setminus \{B\}$ — алфавит символов, подаваемых на вход;
- $\pi : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ — *функция перехода*, описывающая работу машины Тьюринга: L обозначает сдвиг головки влево, R — вправо (существуют также модификации определения, позволяющие головке оставаться на месте);
- $s \in Q$ — *начальное состояние* машины Тьюринга;
- $H \subseteq Q$ — множество *конечных состояний*.

Нас будут интересовать только машины с единственным конечным состоянием $H = \{h\}$. Неформально говоря, машина Тьюринга состоит из бесконечной в обе стороны ленты и головки, расположенной над одной из ячеек этой ленты. На ленте в начальном состоянии s записан вход, являющийся строкой над Σ. Машина производит переходы из одного состояния в другое в соответствии с функцией π, на вход которой подаётся текущее состояние $q \in Q$ и символ $\alpha \in \Gamma$, который

на данном шаге находится под головкой. Головка двигается влево или вправо в соответствии с третьей компонентой результата функции π.

Машина Тьюринга M *вычисляет функцию* $f : \Sigma^* \to (\Gamma^* \setminus \{B\})^*$, если при запуске M со входом $x \in \Sigma^*$ она заканчивает работу (переходит в состояние h), оставляя на ленте строку $f(x)$. В дальнейшем мы для простоты не будем делать различий между алфавитами Σ и $\Gamma\setminus\{B\}$. Обозначим через $t_M(x)$, где $t_M : \Sigma^* \to \mathbb{N}$, количество шагов, которое требуется машине Тьюринга M, чтобы, получив на вход x, перейти в конечное состояние h. Машина Тьюринга M *работает в течение времени* $T : \mathbb{N} \to \mathbb{N}$, если

$$\forall n \in \mathbb{N} \max_{x:|x|=n} t_M(x) \leq T(n),$$

где $|x|$ — длина строки x.

Нам также потребуются недетерминированные и вероятностные машины Тьюринга. Мы не будем давать отдельных определений, а будем считать, что недетерминированная машина Тьюринга — это обычная детерминированная машина Тьюринга с двумя лентами, на которых записаны две части входа: собственно вход и «подсказка»; недетерминированная машина Тьюринга M *вычисляет функцию* $f : \Sigma^* \to \Sigma^*$, если для каждого входа x существует такая «подсказка» $y \in \Sigma^*$, что $M(x, y) = f(x)$. Вероятностная машина Тьюринга — это то же самое, что недетерминированная, изменяется только интерпретация: символы подсказки теперь интерпретируются как случайные символы машины; будем говорить, что вероятностная машина Тьюринга *вычисляет функцию* f на входе x с вероятностью p, если

$$\Pr_{y \in U_m} [M(x, y) = f(x)] = p,$$

где m — количество случайных символов из Σ, U_m — равномерное распределение на строках длины m из алфавита Σ, а $y \in U_m$ означает «y взято по распределению U_m». В дальнейшем обычно $\Sigma = \{0, 1\}$.

В дальнейшем мы будем пользоваться тезисом Чёрча и использовать термины «алгоритм» и «машина Тьюринга» как синонимы; например, «полиномиальный алгоритм» — это алгоритм, реализуемый полиномиальной машиной Тьюринга, т.е. машиной, работающей в течение времени $T(n) = c_1 n^{c_2}$ для некоторых констант c_1 и c_2.

Другой важной для данного исследования моделью вычислений является *схемная сложность* [109, 125, 130, 137, 142, 159]. В схемной модели сложность функции определяется как размер минимальной схемы, которая реализует эту функцию. Схемы состоят из *гейтов*, в качестве которых могут выступать различные булевские функции. Первые работы по схемной сложности относятся, разумеется, к тому же времени, когда Клод Шеннон впервые записал определение булевских схем и начал развивать теорию, позволяющую строить вычисления на основе пропозициональной логики Буля [111, 114, 115].

Дадим точное определение схемы. Прежде всего обозначим через $\mathbb{B}_{n,m}$ множество всех 2^{m2^n} функций $f : \mathbb{B}^n \to \mathbb{B}^m$, где $\mathbb{B} = \{0, 1\}$ — поле из двух элементов.

Определение 1.2 Пусть Ω — некоторое множество булевских функций $f : \mathbb{B}^m \to \mathbb{B}$ (m может быть разным для разных f). Тогда *Ω-схема* — это ациклический направленный граф с метками, состоящий из вершин двух типов:

— вершин входящей степени 0 (вершин, в которые не входят рёбра), маркированных одной из переменных $x_1, \ldots, x_n$,

— и вершин, маркированных одной из функций $f \in \Omega$, в которые входит столько рёбер, какова арность этой функции.

Вершины первого типа называются *входами*, вершины второго типа — *гейтами*[1]. *Размер* схемы — это количество гейтов в ней.

[1] В русскоязычной литературе встречается термин «*вентиль*», и лет сорок назад он был общеупотребителен; но мы, пожалуй, не рискнём

Каждый гейт Ω-схемы вычисляет некоторую булевскую функцию. Соответственно, схемная сложность функции $f : \mathbb{B}^n \to \mathbb{B}^m$ в базисе Ω обозначается как $C_\Omega(f)$ и определяется как размер минимальной Ω-схемы, которая вычисляет функцию f (в которой есть m гейтов, вычисляющих результат применения функции f ко входным битам). Чтобы можно было без оговорок устранить унарные гейты, будем считать, что в гейте вычисляется как сам результат его функции, так и его отрицание. Наша модель вычислений — это булевские схемы с произвольными бинарными гейтами; иными словами, каждый гейт схемы маркируется одной из 16 булевских функций из $\mathbb{B}_{2,1}$. В дальнейшем через $C(f)$ будет обозначаться схемная сложность f в базисе $\mathbb{B}_{2,1}$, состоящем из всех бинарных булевских функций. Мы будем предполагать, что каждый гейт в этой схеме зависит от обоих входов, т.е. нет гейтов, маркированных константами и унарными функциями Id и $\neg$. Это не умаляет общности, потому что такие гейты легко исключить из нетривиальной схемы, не увеличивая её размер.

Стоит отметить, что схемная сложность — одна из немногих моделей вычислений, в которых возможны доказательства *конкретных*, а не асимптотических оценок сложности. Например, Л. Стокмайер в своей диссертации привёл функцию, любая реализация которой при помощи бинарной булевской схемы на входах размера ≤ 616 должна иметь не менее 10^{123} гейтов [6, 123, 125].

Основные результаты в классической схемной сложности относятся к 1980-м гг. и раньше; в них значительна заслуга отечественных учёных [24, 97, 124, 125, 153–156, 158, 160, 161, 163–165]. В последние годы центр усилий в теории схемной сложности переместился на результаты, связанные со схемами ограниченной глубины и/или ограниченным набором вычисляемых в них функций [3, 30, 48, 67, 72, 103,

---

его использовать.

121, 138, 140, 157]. Однако нам потребуются именно классические результаты, поскольку оценки, которые мы будем доказывать в главе 3, будут выполняться над базисом $\mathbb{B}_{2,1}$.

1.3 Основные определения современной криптографии

Как мы уже упоминали, современная криптография основана на разнице между вычислительной сложностью задачи кодирования и задачи декодирования. В этом разделе мы приведём формальные базовые определения криптографических примитивов, которые понадобятся нам в дальнейшем, и обсудим некоторые их свойства.

Начнём с определения односторонней функции. Односторонняя функция — самое естественное развитие основной идеи криптографии: это функция, которую легко вычислить, но трудно обратить. Наши определения следуют [51].

Определение 1.3 Функция $f : \{0,1\}^* \to \{0,1\}^*$ называется *сильно односторонней*, если выполняются следующие условия.

1. Существует детерминированная машина Тьюринга M со входным алфавитом $\{0,1\}$, которая за полиномиальное время вычисляет функцию f.
2. Для каждой вероятностной машины Тьюринга M' и каждого многочлена p для всякого достаточно большого n

$$\Pr\left[M'(f(x), 1^n) \in f^{-1}(f(x))\right] < \frac{1}{p(n)},$$

где вероятность берётся по случайным битам машины M' и входу x длины n (оба распределения равномерные).

Иначе говоря, функция сильно односторонняя, если никакой полиномиальный противник (никакая полиномиальная вероятностная машина Тьюринга) не может обратить её ни на какой значительной доле входов.

Определение 1.4 Функция $f : \{0,1\}^* \to \{0,1\}^*$ называется *слабо односторонней*, если выполняются следующие условия.

1. Существует детерминированная машина Тьюринга M со входным алфавитом $\{0,1\}$, которая за полиномиальное время вычисляет функцию f.
2. Существует такой многочлен p, что для каждой вероятностной машины Тьюринга M' для всякого достаточно большого n

$$\Pr\left[M'(f(x), 1^n) \notin f^{-1}(f(x))\right] > \frac{1}{p(n)},$$

где вероятность берётся по случайным битам машины M' и входу x длины n (оба распределения равномерные).

Иначе говоря, функция слабо односторонняя, если для каждого полиномиального противника существует значительная доля входов, на которой этот противник не может обратить функцию. Очевидно, всякая сильно односторонняя функция является и слабо односторонней, но не наоборот.

В качестве примера функции, которая в настоящее время считается серьёзным кандидатом на звание слабо односторонней, можно привести уже упоминавшуюся выше задачу разложения числа на простые множители. Эта задача, лежащая в основе системы RSA, привлекала значительный интерес исследователей. Долгая работа над схожей, но более слабой задачей — определения, является ли данное число простым, — в конце концов увенчалась успехом, и в 2004 году был построен полиномиальный алгоритм, решающий эту задачу [2][2]; впрочем, вероятностные алгоритмы работают ещё лучше [13, 38, 122]. А вот сама задача разложения на простые множители пока не подда-

[2]Следует отметить, что ещё в 1976 году был разработан детерминированный алгоритм, проверяющий числа на простоту; однако его корректность основана на обобщённой гипотезе Римана [90].

ётся. Наилучшие известные алгоритмы работают в течение времени $2^{O\left((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}\right)}$, где N — число, которое раскладывается на множители (напомним, что эффективное решение должно быть полиномиально не от числа N, а от длины его записи $\log N$) [98]. Таким образом, пока что считается безопасным предполагать, что функция

$$f(x, y) = x \cdot y$$

является слабо односторонней.

В дальнейшем нам потребуются некоторые свойства односторонних функций [51].

Утверждение 1.1 1. Если существует слабо односторонняя функция, то существует и сильно односторонняя функция (т.е. существование слабых и сильных односторонних функций эквивалентно).

2. Если существует хотя бы одна слабо (сильно) односторонняя функция, то существует и слабо (сильно) односторонняя функция, которую можно вычислить машиной Тьюринга, работающей не более n^2 шагов на входе длины n.

Следующим после односторонних функций шагом в конструировании криптографических примитивов являются *семейства односторонних функций* (collections of one-way functions). Дело в том, что, например, криптосистему RSA трудно (и не нужно) выразить в виде одной односторонней функции. Она на самом деле является множеством функций, по одной для каждого числа N и каждого основания e.

Определение 1.5 Семейство функций $\{f_i : D_i \to \{0,1\}^*\}_{i\in I}$ называется *сильно односторонним*, если существуют три таких полиномиальных вероятностных машины Тьюринга $\mathcal{I}$, $\mathcal{D}$ и $\mathcal{F}$, что выполняются следующие условия.

1. *Эффективная вычислимость и случайная выборка.* Результат работы $\mathcal{I}$ на входе 1^n является случайной величиной, принимающей значения во множестве I. Результат работы алгоритма $\mathcal{D}$ на входе $i \in I$ является случайной величиной, принимающей значения во множестве D_i. На входе (i, x), где $i \in I$ и $x \in D_i$, алгоритм $\mathcal{F}$ всегда выдаёт $f_i(x)$.
2. *Невозможность обращения.* Для каждой вероятностной полиномиальной машины Тьюринга M', всякого многочлена p и всех достаточно больших n

$$\Pr\left[M'(f_{I_n}(X_n), 1^n) \in f_{I_n}^{-1}(f_{I_n}(X_n))\right] < \frac{1}{p(n)},$$

где I_n — случайная величина, описывающая выход алгоритма $\mathcal{I}$ на входе 1^n, X_n — случайная величина, описывающая выход алгоритма $\mathcal{D}$ на входе 1^n, и вероятность берётся по случайным битам алгоритмов $\mathcal{I}$, $\mathcal{D}$ и M'.

Семейство функций $\{f_i : D_i \to \{0,1\}^*\}_{i \in I}$ называется *слабо односторонним*, если выполнено первое условие, а второе заменено на следующее.

2. *Слабая невозможность обращения.* Существует такой многочлен p, что для каждой вероятностной машины Тьюринга M' для всякого достаточно большого n

$$\Pr\left[M'(f_{I_n}(X_n), 1^n) \notin f_{I_n}^{-1}(f_{I_n}(X_n))\right] > \frac{1}{p(n)},$$

где вероятность берётся по случайным битам алгоритмов $\mathcal{I}$, $\mathcal{D}$ и M'.

Саму тройку алгоритмов $(\mathcal{I}, \mathcal{D}, \mathcal{F})$ мы также будем называть семейством односторонних функций.

Например, для семейства функций RSA множество индексов I состоит из пар (N, e), где N — произведение двух различных простых чисел (по практическим соображениям, p и q не должны быть

слишком близки, между ними должна быть разница не менее $2N^{1/4}$), $N = pq$, $p \neq q$, а e — некоторое число, $1 < e < (p-1)(q-1)$, взаимно простое с $(p-1)(q-1)$. Множества определения функций равны $D_{(N,e)} = \mathbb{Z}_N$. Функция $f_{(N,e)}$ определяется следующим образом:

$$f_{(N,e)}(x) = x^e \bmod N.$$

Чтобы определить алгоритм $\mathcal{I}_{RSA}$, нужно представить вероятностный полиномиальный алгоритм, который выдаёт равномерно распределённые простые числа. Такие алгоритмы существуют, но можно рассмотреть и более эффективный алгоритм, который будет просто выбирать два числа в интервале $[2^{n-1}, 2^n]$, а затем проверять каким-нибудь эффективным тестом на простоту, являются ли они простыми [13,35,38, 96,122,150]. Поскольку простые числа плотны, алгоритм будет успешно выдавать пару простых чисел достаточно часто. Алгоритм $\mathcal{D}_{RSA}$ в данном примере будет просто равномерно выбирать случайный элемент множества $\mathbb{Z}_N^* = \{1, \ldots, N\}$, а алгоритм $\mathcal{F}_{RSA}$ будет методом последовательного возведения в квадрат вычислять $f_{(N,e)}$.

Для того чтобы построить криптосистему, то есть метод передачи сообщений между двумя участниками, просто односторонних функций недостаточно. Нужно не только чтобы противник не смог расшифровать передаваемый код, но и чтобы второй участник смог это сделать. Поэтому для построения криптосистемы необходимо построить не просто семейство односторонних функций, а семейство функций с секретом (trapdoor functions), для которых существует такая дополнительная информация (секрет, trapdoor), что с её помощью обратить функцию становится просто.

Определение 1.6 Пусть $\mathcal{I}$ — вероятностный полиномиальный алгоритм, $\mathcal{D}$ и $\mathcal{F}$ — детерминированные полиномиальные алгоритмы. Обозначим через $I_1(1^n)$ и $I_2(1^n)$ первую и вторую половины выхода $\mathcal{I}(1^n)$

соответственно (пусть $n = 2k$). Тогда тройка алгоритмов $(\mathcal{I}, \mathcal{D}, \mathcal{F})$ называется *семейством перестановок с секретом*, если выполнены следующие условия.

1. *Это односторонние перестановки.* Тройка $(\mathcal{I}_1, \mathcal{D}, \mathcal{F})$ индуцирует семейство односторонних перестановок, т.е. для любой вероятностной полиномиальной машины Тьюринга M и любого многочлена p для достаточно больших n

$$\Pr\left[M(F_{I_n}(X_n), I_n) \in F_{I_n}^{-1}F_{I_n}(X_n)\right] < \frac{1}{p(n)},$$

   где I_n — случайная величина, результат работы $\mathcal{I}_1$ на входе 1^n, X_n — случайная величина, результат работы $\mathcal{D}$ на входе I_n, а вероятность берётся по случайным битам всех участвующих алгоритмов (см. [51]).
2. *Но их легко обратить с секретом.* Существует такой полиномиальный алгоритм Inv, что для каждого $(i_1, i_2) \in I_1 \times I_2 = \mathcal{I}(1^n)$ и каждого $x \in D_i$ верно, что

$$\mathrm{Inv}(i_2, \mathcal{F}(i_1, x)) = x.$$

Можно определить семейства перестановок с секретом, для которых все алгоритмы, в том числе алгоритм обращения, вероятностные, и определение даётся с учётом возможности (маловероятных) ошибок [51]. Однако для целей нашего исследования в главе 3 потребуется только определение 1.6, причём его несколько модифицированный вариант (см. определение 3.1).

Глава 2

Новые конструкции полных односторонних функций

2.1 Введение

Задачи, возникающие в теоретической информатике, естественным образом представляются в виде задач распознавания тех или иных *языков*, или множеств строк в некотором алфавите (обычно в алфавите $\{0,1\}$). Языки же, в свою очередь, делятся на *сложностные классы*. Например, класс P состоит из языков, задачу принадлежности к которым можно решить на детерминированной машине Тьюринга, заканчивающей работу за время, полиномиальное от длины её входа, а класс NP — из языков, принадлежность к которым определяется за полиномиальное время на недетерминированной машине Тьюринга.

Развитие теории сложности вычислений началось с работ, исследующих сложность отдельных задач. Однако первые же исследования показали, что методы, позволяющие исследовать сложностные классы в целом, а не рассматривать каждую задачу по отдельности, были бы значительно полезнее. Одним из первых достижений теории сложности вычислений стала *теорема Кука–Левина* [32, 151], позволившая развить теорию NP-полных задач [49, 76], а затем и полных задач для других сложностных классов, например, для сложности в среднем [26, 63, 84, 133]; см. также общие источники по теории сложности вычислений [12, 64, 96].

Полная задача того или иного сложностного класса — это такая задача, к которой сводятся все остальные (т.е. такая задача, что если научиться решать её эффективно, то можно будет эффективно решать и все остальные задачи класса). Понятия «сведе́ния» у разных классов отличаются, и даже в пределах одного контекста сведе́ния могут

разниться (например, сведения по Тьюрингу и по Карпу в классе NP). Ниже мы определим понятие полной односторонней функции, которое будет центральным объектом анализа в этой главе.

Для теоретической информатики крайне важны возможности, появляющиеся при наличии в сложностном классе полной задачи. Если полная задача есть, предмет анализа можно сместить с класса в целом (о котором обычно «в лоб» мало что можно доказать) на одну конкретную задачу. К примеру, задача выполнимости булевых формул SAT (от satisfiability problem), являясь классической NP-полной задачей, привлекает значительный интерес исследователей [60, 143]. Аналогично, для класса DistNP (класса задач из NP, снабжённых полиномиально вычислимыми распределениями), который более тесно связан с настоящей работой, была доказана полнота задачи Поста и задачи преобразования матриц [23, 62, 63, 128, 133]. Несмотря на то, что было доказано, что полные задачи существуют не для всех классов [118], полные задачи в теории сложности вычислений — один из самых полезных объектов.

Здесь стоит, правда, отметить, что не все полные задачи действительно полезны для практических и/или теоретических применений. В то время как такие полные задачи, как выполнимость или раскраска графа, допускают комбинаторные подходы, существуют и задачи, анализировать которые ничуть не проще, чем анализировать соответствующий класс сложности. Такие задачи обычно возникают в результате процедур диагонализации и требуют перечисления всех машин Тьюринга или всех задач из определённого сложностного класса.

Настоящая работа посвящена задачам криптографии. Достаточно долгое время о полных задачах в криптографии мало что было известно. В то время как «обычные» сложностные классы обзавелись полными задачами относительно скоро, между определением крипто-

системы с открытым ключом [39] и полной задачей для класса криптосистем с открытым ключом (с ограниченной ошибкой декодирования) [55, 66] прошло тридцать лет. Более того, разработанные полные криптосистемы пока что относятся к «плохой» разновидности полных задач: они требуют перечисления всех машин Тьюринга и вряд ли могут иметь дальнейшее применение, как для практической реализации, так и для теоретического анализа сложности или надёжности.

Прежде чем рассматривать криптосистемы с открытым ключом, кажется естественным задать аналогичный вопрос о более простом объекте: об односторонних функциях (как известно, криптография с открытым ключом эквивалентна существованию секретно обратимой функции, которая является частным случаем односторонней функции). Первый шаг в направлении «полезных» полных односторонних функций был сделан Л. А. Левиным; он предложил конструкцию первой известной полной односторонней функции [85] (см. также [51, 81, 152]). Полнота здесь понимается в следующем смысле.

Определение 2.1 Пусть функция $f : \{0,1\}^* \to \{0,1\}^*$ такова, что если существует некоторая функция $g : \{0,1\}^* \to \{0,1\}^*$, являющаяся сильно (слабо) односторонней функцией, то f тоже является сильно (слабо) односторонней функцией. Тогда f называется *полной* сильно (слабо) односторонней функцией.

Первая конструкция полной слабо односторонней функции, построенная Л. А. Левиным, получила название *универсальной* односторонней функции. Эта конструкция использует универсальную машину Тьюринга U для вычисления следующей функции:

$$f_{uni}(\mathrm{desc}(M), x) = (\mathrm{desc}(M), M(x)),$$

где $\mathrm{desc}(M)$ — описание машины Тьюринга M, а $M(x)$ — ответ, выдаваемый M на входе x за $\leq |x|^2$ шагов. Если среди машин M есть

односторонние функции (легко показать, что если есть хоть какие-то односторонние функции, то есть и односторонние функции, которые работают за, к примеру, квадратичное время от длины своего входа [51]), то f_{uni} будет слабо односторонней функцией.

Отметим, что такая полная односторонняя функция относится к вышеупомянутому классу «бесполезных» полных задач. Использовать эту конструкцию на практике не представляется возможным. Поэтому вполне естественно, что Л. А. Левин поставил задачу поиска «комбинаторных» полных односторонних задач, которые бы не зависели от перечисления машин Тьюринга или от подачи на вход описаний машин Тьюринга. В течение 15 лет проблема оставалась открытой, а затем была разрешена самим же Левиным [152]. В [152] используется оригинальная идея: полностью запретить недетерминированный выбор при вычислении прямой функции, оставив его только для обращения (ниже мы подробнее опишем эту конструкцию).

Также Л. А. Левин в [152] поставил задачу поиска других комбинаторных полных односторонних функций. В этой главе будет показано, как, используя схожие с [152] идеи, можно получить полную одностороннюю функцию на базе задач переписывания строк (string rewriting). В работе [132] было доказано, что эти задачи полны в среднем. Немного более сложная конструкция приведёт к успеху и для задачи Поста. Кроме того, мы обсудим общие свойства, которые должны присутствовать у комбинаторной задачи, чтобы её можно было модифицировать так, чтобы аналогичными рассуждениями получить полную одностороннюю функцию, и докажем об этом формальное утверждение. Основные результаты этой главы опубликованы в [81].

2.2 Задача достижимости с распределением для полусистем Туэ

Рассмотрим конечный алфавит $\mathcal{A}$. Упорядоченную пару строк $\langle g, h\rangle$ над алфавитом $\mathcal{A}$ мы будем называть *правилом подстановки* (rewriting rule; иногда правило подстановки также называют *продукцией*, production). Эти пары записываются как $g \to h$ и интерпретируются как правила, по которым в других строках производятся подстановки. Формально говоря, для двух строк $u, v \in \mathcal{A}^*$ будем писать $u \Rightarrow_{g\to h} v$, если $u = agb$, $v = ahb$ для некоторых строк $a, b \in \mathcal{A}^*$. Набор правил подстановки называется *полусистемой Туэ* (semi-Thue system). Для полусистемы Туэ R будем писать $u \Rightarrow_R v$, если $u \Rightarrow_{g\to h} v$ для некоторого правила подстановки $\langle g, h\rangle \in R$. Мы будем записывать $u \Rightarrow_R^* v$ в том случае, если существует такая конечная последовательность правил подстановки $\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle \in R$, что

$$u = u_0 \Rightarrow_{g_1\to h_1} u_1 \Rightarrow_{g_2\to h_2} u_2 \Rightarrow \ldots \Rightarrow_{g_m\to h_m} u_m = v$$

(то есть возьмём вместо исходного определения $\Rightarrow_R$ его транзитивно-рефлексивное замыкание). Кроме того, определим ограниченную версию $\Rightarrow_R^*$: будем записывать $u \Rightarrow_R^n v$, если существует такая конечная последовательность правил подстановки $\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle \in R$, что

$$u = u_0 \Rightarrow_{g_1\to h_1} u_1 \Rightarrow_{g_2\to h_2} u_2 \Rightarrow \ldots \Rightarrow_{g_m\to h_m} u_m = v, \text{ где } m \le n,$$

то есть v получается из u посредством правил подстановки из R не более чем за n шагов.

Теперь мы можем определить задачу достижимости с распределением для полусистем Туэ.

Задача 2.1 (достижимости с распределением для полусистем Туэ)
Вход. Полусистема Туэ $R = \{\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle\}$, две битовые строки

u и v, натуральное число n. Размер входа, таким образом, равен

$$\log n + |u| + |v| + \sum_{1}^{m} (|g_i| + |h_i|).$$

Задача. Верно ли, что $u \Rightarrow_R^n v$?

Распределение. Случайно и независимо выбрать натуральные числа n и m и битовые строки u и v. Затем случайно и независимо выбрать битовые строки $g_1, h_1, \ldots, g_m, h_m$. Числа и строки выбираются по стандартному квазиравномерному распределению, то есть распределению, плотность которого для натуральных чисел пропорциональна $\frac{1}{n^2}$, а для битовых строк — пропорциональна $\frac{2^{-|u|}}{|u|^2}$.

Мы не будем более подробно останавливаться на свойствах полусистем Туэ; они описаны, например, в [28, 126, 145]. Отметим только интересующее нас свойство: в [135] было показано, что задача достижимости с распределением полна для класса DistNP. Доказательство использовало тот факт, что в полусистеме Туэ можно промоделировать работу машины Тьюринга. Именно это соображение и привлекло внимание к исследованию полусистем Туэ — первым их применением было доказательство неразрешимости задачи достижимости, основанное на моделировании задачи остановки машины Тьюринга [101, 126].

Однако для наших дальнейших конструкций нам понадобится ещё одно понятие достижимости в полусистемах Туэ. Для полусистемы Туэ R будем записывать $u \overset{!}{\Rightarrow}_R v$, если $u = agb$ и $v = ahb$ для некоторого правила подстановки $\langle g, h \rangle \in R$ и строк $a, b \in \mathcal{A}^*$, причём не существует никакого другого правила подстановки $\langle g', h' \rangle \in R$, для которого было бы верно, что $u = a'g'b'$ и $v = a'h'b'$ для некоторых $a', b' \in \mathcal{A}^*$. Аналогично отношению достижимости $\Rightarrow_R$, мы расширяем $\overset{!}{\Rightarrow}_R$ до своего транзитивно-рефлексивного замыкания $\overset{!}{\Rightarrow}_R^*$, а также вводим ограниченную версию: $u \overset{!}{\Rightarrow}_R^n v$, если $u \overset{!}{\Rightarrow}_R^* v$, и соответствующая

цепь состоит из не более чем n шагов.

Иначе говоря, $u \overset{!}{\Rightarrow}^*_R v$, если $u \Rightarrow^*_R v$, и на каждом шаге этого вывода есть ровно одно применимое правило подстановки. Эта единственность (или, точнее говоря, этот детерминизм) крайне важна для того, чтобы в разделе 2.5 применить метод Левина.

2.3 Задача Поста

В [63] было показано, что следующая задача является полной для класса DistNP (см. также [23, Замечание 2]).

Задача 2.2 (Задача Поста)
Вход. Натуральное число m, пары битовых строк

$$\Gamma = \{\langle u_1, v_1\rangle, \ldots, \langle u_m, v_m\rangle\},$$

битовая строка x, натуральное число n. Размер входа, таким образом, равен $\log n + |x| + \sum_1^m (|u_i| + |v_i|)$.

Задача. Верно ли, что $u_{i_1} \ldots u_{i_k} = x v_{i_1} \ldots v_{i_k}$ для некоторого $k \leq n$?

Распределение. Случайно и независимо выбрать натуральные числа n и m и битовую строку x. Затем случайно и независимо выбрать битовые строки $u_1, v_1, \ldots, u_m, v_m$. Числа и строки выбираются по стандартному квазиравномерному распределению, т.е. распределению, плотность которого для натуральных чисел пропорциональна $\frac{1}{n^2}$, а для битовых строк — пропорциональна $\frac{2^{-|u|}}{|u|^2}$.

Для целей построения полных односторонних функций понадобится небольшая модификация этой задачи. Мы поставим вопрос так: верно ли в обозначениях задачи 2.2, что

$$u_{i_1} \ldots u_{i_k} y = x v_{i_1} \ldots v_{i_k}$$

для некоторой битовой строки y?

Свойства задачи Поста во многом похожи на свойства полусистем Туэ. Задача Поста также была сформулирована как комбинаторный вариант неразрешимой задачи и получила известность как одна из простых и удобных неразрешимых задач [61, 100, 108, 119]. Сформулированная нами задача также неразрешима, если удалить ограничение n; полнота ограниченной модифицированной версии для класса DistNP доказана в [63].

Продолжим аналогию с полусистемами Туэ. Функцию, основанную на модифицированной задаче Поста, разумно рассматривать как вывод при помощи некоторых правил вывода. А именно, для некоторого непустого списка

$$\Gamma = (\langle u_1, v_1\rangle, \ldots, \langle u_m, v_m\rangle)$$

будем говорить, что из битовой строки x *выводится* строка y *за один шаг*, и будем писать $x \vdash_\Gamma y$, если есть такая пара $\langle u, v\rangle \in \Gamma$, что $uy = xv$. Отношение «выводится» $\vdash_\Gamma^*$ определим как транзитивно-рефлексивное замыкание отношения «выводится за один шаг»: из битовой строки x *выводится* строка y, если есть такая последовательность пар $\langle u_1, v_1\rangle, \ldots, \langle u_m, v_m\rangle \in \Gamma$ и такая последовательность битовых строк $x = x_1, x_2 \ldots, x_m, x_{m+1} = y$, что для каждого $1 \leq i \leq m$ $u_i x_{i+1} = x_i v_i$.

Чтобы применить метод Левина, нам нужно будет избавиться от потенциального недетерминизма в конструкции задачи Поста. На этот раз описание детерминированной версии $\overset{!}{\vdash}$ отношения $\vdash$ окажется более сложным, чем в случае полусистем Туэ. Тривиального повторения той же идеи детерминированности на каждом шаге окажется недостаточно: мы не сможем в естественной записи машины Тьюринга передвинуть головку машины налево (см. раздел 2.6). Поэтому придётся ввести проверку на один шаг вперёд: если у системы есть два правила, которые можно применить, и одно из этих правил на следующем же шаге приводит к обрыву цепочки, мы детерминированно будем выби-

рать другое правило.

Формально, будем говорить, что из x выводится y за один шаг ($x \overset{!}{\vdash}_\Gamma y$), если существует не более двух таких пар $\langle p, s\rangle, \langle p', s'\rangle \in \Gamma$, что $py = xs$ и $p'y' = xs'$ для некоторых строк y, y' (где $y \neq y'$, но p может совпадать с p': два разных применения одного и того же правила всё равно приводят к недетерминизму) и, более того, к строке y' неприменимо ни одно из содержащихся в Γ правил. Как и ранее, через $x \overset{!}{\vdash}{}^*_\Gamma y$ будем обозначать транзитивно-рефлексивное замыкание этого отношения, а также введём ограниченную версию отношения: будем записывать $u \overset{!}{\vdash}{}^n_\Gamma v$, если $u \overset{!}{\vdash}_\Gamma v$ за не более чем n шагов.

2.4 Полная односторонняя функция, основанная на задаче поиска замощения

Прежде чем представлять наши собственные конструкции, напомним полную одностороннюю функцию, рассмотренную Л. А. Левиным в [152]. В этом разделе мы немного модифицируем конструкцию Левина и представим другую конструкцию моделирования машин Тьюринга, основанную на идеях из [134]. Разница с исходной конструкцией в том, что Л. А. Левин рассматривал функцию замощения, в которой углы плиток, а не их стороны, были помечены символами, и совпадать должны были все четыре сходящихся в угле символа.

В нашей конструкции *плитка* — это квадрат, каждая сторона которого помечена символом из конечного алфавита $\mathcal{A}$. Мы предполагаем, что количество доступных плиток каждого вида не ограничено. *Замощение* квадрата $n \times n$ — это множество из n^2 плиток, расположенных в виде сетки $n \times n$, причём символы на общих сторонах соседних плиток совпадают.

Нам будет удобно рассматривать задачу поиска замощения как систему преобразования строк, похожую на полусистему Туэ (и ос-

нованную на тех же полугрупповых идеях). Зафиксируем конечное множество плиток T. Будет говорить, что T *переводит* строку x алфавита $\mathcal{A}$ в строку y, где $|x| = |y|$, если существует такое замощение квадрата размера $|x| \times |x|$, что символы на нижних сторонах плиток нижней строки образуют строку x, а символы на верхних сторонах плиток верхней строки образуют строку y. Будем записывать этот факт как $x \longrightarrow_T y$.

Под *процессом замощения* будем понимать дополнение частично замощенного квадрата до полностью замощенного шаг за шагом, по одной плитке за шаг. Аналогично полусистемам Туэ, мы определим $x \overset{!}{\longrightarrow}_T y$ как $x \longrightarrow_T y$ с дополнительным ограничением: мы разрешаем добавлять новую плитку в частично замощенный квадрат только в том случае, если во множестве T есть ровно одна подходящая плитка. Заметим, что при таком подходе для каждой строки x либо существует ровно одна строка y, для которой $x \overset{!}{\longrightarrow}_T y$, либо такой строки не существует вовсе.[1]

Зафиксируем некоторое кодирование множества меток битовыми строками, которое кодирует каждую метку битовой строкой длины не более $\log|\mathcal{A}| + O(1)$. Вплоть до конца этого раздела мы будем отождествлять плитку и её код. Для задачи замощения это несущественно, но будет играть важную роль в следующем разделе.

Определение 2.2 *Функция замощения* — это функция $f : \{0,1\}^* \to$

[1] Стоит отметить, что существует и другая задача замощения, в которой вопрос ставится так: можно ли при помощи данного набора плиток полностью замостить плоскость? Эта проблема также неразрешима (по тем же причинам); она была предложена Хао Вангом в 1961 году и привела к ряду интересных и неожиданных результатов [20, 36, 37, 75, 131]. Но мы используем только «ограниченную» версию.

$\{0,1\}^*$, определённая следующим образом:

— если вход имеет вид (T, x) для конечного множества плиток T и строки x, и $x \xrightarrow{!}_T y$, то $f(T, x) = (T, y)$;

— в противном случае, $f(T, x) = (T, x)$.

Теорема 2.1 Если односторонние функции существуют, то функция замощения является слабо односторонней функцией.

Доказательство. Доказательство этой теоремы следует доказательству существования полных односторонних функций, приведённому в [51] (и впервые предложенному в [85]), практически дословно. Пусть g — сохраняющая длину (т.е. $|g(x)| = |x|$ для любого x) односторонняя функция, для которой существует реализующая её машина Тьюринга, работающая не более n^2 шагов на входах длины n (по утверждению 1.1, если существуют какие-нибудь односторонние функции, существует и такая).

Рассмотрим функцию замощения f. Докажем сначала, что всякую машину Тьюринга можно просимулировать посредством задачи замощения.

Лемма 2.1 Для всякой детерминированной машины Тьюринга M с алфавитом символов ленты $\Gamma = \{0, 1, B\}$, работающей в течение не более чем n^2 шагов на входе длины n, существует такой набор плиток T_M, что его набор меток $\mathcal{A}$ включает в себя Γ, причём $M(x) = y$, $|x| = |y|$, тогда и только тогда, когда

$$\$sxB^{n(n-1)}\# \xrightarrow{!}_{T_M} \$hyB^{n(n-1)}\#,$$

где s — начальное состояние M, h — конечное состояние, B — её пустой символ, а $\$, \# \notin \Gamma$ — дополнительные символы, маркеры начала и конца.

Доказательство. Рассмотрим машину Тьюринга M, у которой через Q_M обозначим набор состояний, через s — начальное состояние, h —

конечное состояние, π — функцию перехода, $\{0, 1, B\}$ — символы ленты. Через \$ обозначим маркер начала, а через $\#$ — маркер конца. Мы также введём новый символ для каждой пары из множества $Q_M \times \{0, 1, B\}$. Тогда искомый набор T_M будет состоять из следующих плиток.

1. Для каждого символа ленты $a \in \{0, 1, B\}$ добавим плитки

2. Для каждых $a, b, c \in \{0, 1, B\}$, $q \in Q_M \setminus \{h\}$, $p \in Q$, если

$$\pi_M(q, a) = (p, b, R),$$

то добавим плитки

3. Для каждых $a, b, c \in \{0, 1, B\}$, $q \in Q_M \setminus \{h\}$, $p \in Q_M$, если

$$\pi_M(q, a) = (p, b, L),$$

то добавим плитки

4. Наконец, для \$ и $\#$ добавим плитки

Теперь легко видеть, что каждой операции, производимой машиной Тьюринга M, теперь соответствует допустимое продолжение замощения квадрата. Таким образом, для каждого вычисления на детерминированной машине Тьюринга M на входе x, продолжающегося

не более чем n^2 шагов, можно построить замощение квадрата $|x|^2 \times |x|^2$, в котором нижние метки нижнего ряда образуют входную строку (дополненную символами B до длины n), а верхние метки верхнего ряда образуют результат вычислений (дополненный символами B до длины n). Если вычисление продолжается менее n^2 шагов, квадрат можно, тем не менее, замостить полностью, используя для верхних уровней плитки из пункта 1. □

Итак, по лемме 2.1, существует такой конечный набор плиток T_M, что

$$\$sxB^{n(n-1)}\# \longrightarrow^*_{T_M} \$hyB^{n(n-1)}\#$$

эквивалентно $g(x) = y$. Значит, с константной вероятностью, равной вероятности того, что префиксом входной строки окажется в точности описание T_M, обращение функции замощения окажется эквивалентным обращению функции g. □

2.5 Полная односторонняя функция, основанная на полусистемах Туэ

Первая новая полная односторонняя функция, которую мы рассмотрим, основана на задаче достижимости с распределением для полусистем Туэ. Чтобы построить полную одностороннюю функцию, нам нужно сначала превратить эту задачу в функцию, а затем использовать метод Левина, следя при этом за сохранением длины.

Определение 2.3 *Функция достижимости для полусистем Туэ* (ФДПТ) — это функция $f : \mathcal{A}^* \to \mathcal{A}^*$, определённая следующим образом:

— если вход имеет вид $(\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle, x)$, и в полусистеме Туэ $\Gamma = (\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle)$ верно, что $x \overset{!}{\Rightarrow}^t_\Gamma y$, $t = |x|^2 + 4|x| + 2$, в Γ нет правил подстановки, которые были бы применимы к y, и $|y| = |x|$, то $f(\Gamma, x) = (\Gamma, y)$;

— в противном случае, $f(\Gamma, x) = (\Gamma, x)$.

Теорема 2.2 Если односторонние функции существуют, то ФДПТ является слабо односторонней функцией.

Доказательство. Доказательство этой теоремы, как и Теоремы 2.1, следует идеям [51].

Во-первых, заметим, что ФДПТ легко вычислить: нужно просто рассмотреть первую часть входа как описание полусистемы Туэ (если не получается, вернуть вход) и применять её правила, пока либо найдётся неоднозначность, либо пройдут $|x|^2 + 4|x| + 2$ шага, либо входная строка превратится в другую строку такой же длины y, и никаких других правил применить уже невозможно. В первых двух случаях следует вернуть вход, а в третьем подать на выход (Γ, y).

Далее, заметим, что с константной вероятностью (для равномерного распределения эта вероятность пропорциональна $\frac{1}{|R|^2 2^{|R|}}$) описание любой фиксированной полусистемы Туэ (это просто битовая строка) появится в качестве первой части входа ФДПТ. Рассмотрим сохраняющую длину одностороннюю функцию g и машину Тьюринга M_g, которая вычисляет g за квадратичное время (они существуют по предположению теоремы и утверждению 1.1).

Для продолжения доказательства нам потребуется закодировать машину Тьюринга через полусистему Туэ. Для этого нужно достаточно эффективное кодирование (для того, чтобы распределения на входах и закодированных входах полиномиально трансформировались друг в друга). Это позволяет сделать так называемая *динамическая двоичная схема кодирования*, которая была разработана Ю. Гуревичем и использовалась в [63, 134, 135].

Утверждение 2.1 Для всякого конечного алфавита $\mathcal{A}$, для которого $|\mathcal{A}| > 2$, и каждой пары начинающихся с единиц битовых строк x и y существует *динамическая двоичная схема кодирования* $\mathcal{A}$ над

алфавитом $\{0,1\}$ со следующими свойствами.

1. Все коды (двоичные коды символов $\mathcal{A}$) имеют одну и ту же длину $l = 2\log|x| + O(1)$.
2. Обе строки x и y отличимы от каждого кода, т.е. ни один код не является подстрокой x или y.
3. Если непустой суффикс z кода u является префиксом кода v, то $z = u = v$ (т.е. всегда можно распознать, где заканчивается одно кодовое слово и начинается следующее).
4. Строки x и y можно единственным образом записать в виде конкатенации битовых строк 1, 10, 000, и 100, которые не являются префиксами ни одного из кодовых слов.

Теперь можно доказать, что полусистемы Туэ могут симулировать детерминированные машины Тьюринга.

Лемма 2.2 Для всякой детерминированной машины Тьюринга M существует такая полусистема Туэ R_M, что $M(x) = y$ тогда и только тогда, когда

$$\underline{s}x\underline{\$} \stackrel{!}{\Rightarrow}{}^{t}_{R_M} \underline{h}y\underline{\$},$$

где $t = T + 2|x| + 2|y| + 2$, T — время работы M на входе x.

Доказательство. Определим полусистему Туэ, которая будет соответствовать данной машине Тьюринга M (будем обозначать её через R_M). Правила подстановки делятся на три части:

$$R_M = R_1 \cup R_2 \cup R_3.$$

Положим $\mathcal{B} = \{1, 10, 100, 000\}$, зафиксируем динамическую двоичную схему кодирования для $\mathcal{A}$, x и y и обозначим через $\underline{w}$ код строки w в этой схеме.

R_1 состоит из следующих правил для каждой строки $u \in \mathcal{B}$:

$$\begin{aligned} \underline{su} &\to \underline{\$us_1}, \\ \underline{s_1u} &\to \underline{us_1}, \\ \underline{us_1\$} &\to \underline{s_2u\$}, \\ \underline{us_2} &\to \underline{s_2u}, \\ \underline{\$s_2} &\to \underline{\$s}, \end{aligned}$$

где s, s_1, s_2, \$ — вспомогательные символы. Эти правила нужны для того, чтобы переписать исходную строку $\underline{sx\$}$ в $\underline{\$sx\$}$. Поскольку x можно единственным образом записать как $u_1 \dots u_m$ для некоторых $u_i \in \mathcal{B}$, это преобразование можно выполнить за $2m + 1 \leq 2|x| + 1$ шагов.

R_2 состоит из правил подстановки, соответствующих инструкциям машины Тьюринга.

1. Для каждого состояния $q \in Q \setminus \{h\}$, $p \in Q$, $a, b, c \in \{0, 1, B\}$:

$$\pi_M(q, a) = (p, b, R) \quad \Rightarrow \quad qac \to bpc, qa\$ \to bpB\$ \in R_2.$$

2. Для каждого состояния $q \in Q \setminus \{h\}$, $p \in Q$, $a, b, d \in \{0, 1, B\}$ и $c \in \{0, 1, \$\}$,

$$\pi_M(q, a) = (p, b, L) \quad \Rightarrow \quad dqac \to pdbc, dqB\$ \to pdbB\$ \in R_2$$

для $a \neq B$, $c \neq \$$, или $b \neq B$.

R_1 и R_2 полностью аналогичны конструкции, приведённой в [134], где она используется для доказательства полноты в среднем задачи эквивалентности слов в полусистеме Туэ. Благодаря свойствам динамической двоичной схемы использующиеся в системе строки однозначно декодируются, а правила подстановки соотвествуют инструкциям машины Тьюринга. Третья часть конструкции [134] направлена на то, чтобы перевести результат из $\underline{\$sy\$}$, где y — результат вычисления машины Тьюринга, в запись последовательности вычислений (протокола) машины Тьюринга. Это и используется для доказательства того, что недетерминированные полусистемы Туэ DistNP-сложны.

Поэтому мы на этот раз отклонимся от [134]: нам нужен другой набор правил, потому что мы хотим в ответе получить выход машины Тьюринга, а не протокол её работы. Наша версия R_3 выглядит следующим образом:

$$\begin{aligned}
\underline{\$hu} &\to \underline{\$}u\underline{s_5}, \\
\underline{s_5u} &\to u\underline{s_5}, \\
\underline{s_5u\$} &\to u\underline{s_6\$}, \\
\underline{us_6} &\to \underline{s_6}u, \\
\underline{\$s_6} &\to \underline{h},
\end{aligned}$$

где s_5, s_6 — вспомогательные символы. Это преобразование можно выполнить за $2|y| + 1$ шагов.

Эти правила просто транслируют $\underline{y}$ обратно из закодированного вида в y, а также добавляют h в начало выхода, получая, таким образом, истинное конечное состояние исходной машины Тьюринга M. □

По лемме 2.2, существует такая полусистема Туэ R_M, что $\underline{s}x\underline{\$} \Rightarrow^{*,t}_{R_M} \underline{h}y\underline{\$}$ эквивалентно $g(x) = y$. Следовательно, на константной доле входов обращение ФДПТ эквивалентно обращению g. □

2.6 Полная односторонняя функция на базе задачи Поста

В этом разделе мы опишем одностороннюю функцию, основанную на задаче Поста, и докажем её полноту. Напомним, что в разделе 2.3 мы определили детерминированный вариант отношения выводимости не наиболее прямолинейным путём, а с возможным «забеганием вперёд» на один шаг: если есть два варианта применения правил подстановки, и один из них сразу же ведёт в тупик, можно выбрать второй и считать это детерминированным выбором.

Сначала определим собственно функцию.

Определение 2.4 *Функция преобразования Поста* (ФПП) — это

функция $f : \mathcal{A}^* \to \mathcal{A}^*$, определённая следующим образом:

— если вход имеет вид

$$(\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle, x),$$

и для системы правил подстановки

$$\Gamma = (\langle g_1, h_1\rangle, \ldots, \langle g_m, h_m\rangle) :$$

верно, что $x \xrightarrow[\Gamma]{!}{}^{n^4} y$, в Γ нет правил подстановки, которые можно было бы применить к y, и $|y| = |x|$, то $f(\Gamma, x) = (\Gamma, y)$;

— в противном случае, f $f(\Gamma, x) = (\Gamma, x)$.

Теорема 2.3 Если односторонние функции существуют, то ФПП является слабо односторонней функцией.

Доказательство. Доказательство этой теоремы аналогично доказательствам теорем 2.1 и 2.2. Предположим, что g — сохраняющая длину односторонняя функция, а M — реализующая эту функцию машина Тьюринга, которая работает за время не более n^2, где n — длина входа.

Нам нужно свести вычисления универсальной машины Тьюринга к задаче Поста; способ это сделать описан, например, в [63].

Сначала определим конструкцию системы правил подстановки Γ_M, соответствующих данной машине Тьюринга M. Как обычно, обозначим через Q множество состояний машины Тьюринга M, через s — её начальное состояние, через h — конечное состояние, через π_M — функцию перехода машины Тьюринга M, а через $0, 1, B$ — символы, которые могут появляться на ленте. Для всех символов мы используем динамическую двоичную кодовую схему, описанную в разделе 2.5. Γ_M состоит из следующих правил вывода.

1. Для каждого символа ленты u:

$$\langle \underline{u}, \underline{u}\rangle.$$

2. Для каждого состояния $q \in Q_M \backslash \{h\}$, $p \in Q$, $a, b \in \{0, 1\}$ и каждого правила перехода $\pi_M(q, a) = (p, b, R)$:

$$\langle \underline{qa}, \underline{bp} \rangle.$$

3. Для каждого состояния $q \in Q_M \setminus \{h\}$, $p \in Q$, $a \in \{0, 1\}$ и каждого правила перехода $\pi_M(q, B) = (p, a, R)$:

$$\langle \underline{qB}, \underline{bpB} \rangle.$$

4. Для каждого состояния $q \in Q_M \setminus \{h\}$, $p \in Q$, $a, b, c \in \{0, 1\}$ и каждого правила перехода $\pi_M(q, a) = (p, b, L)$:

$$\langle \underline{cqa}, \underline{pcb} \rangle.$$

5. Для каждого состояния $q \in Q_M \setminus \{h\}$, $p \in Q$, $a \in \{0, 1\}$ и каждого правила перехода $\pi_M(q, B) = (p, a, L)$:

$$\langle \underline{cqB}, \underline{pcbB} \rangle.$$

Лемма 2.3 Для детерминированной машины Тьюринга M с временем работы, не превышающем n^2, и соответствующего этой машине Тьюринга набора правил подстановки Γ_M

$$M(x) = y \text{ тогда и только тогда, когда } \underline{sxB} \vdash_{\Gamma_M}^{*, n^4} \underline{hyB}.$$

Доказательство. Состояние машины Тьюринга M после t шагов вычислений представляется в виде строки xqy, где q — текущее состояние M, x — состояние ленты перед головкой машины, а y — состояние ленты справа от головки до первого пустого символа. Симуляция одного шага работы машины M из состояния xqy состоит из не более чем $|x|$ применений правила 1, одного применения одного из правил 2–5, а затем не более чем $|y| - 1$ применений правила 1.

Заметим, что правило 1 применимо не только на финальном этапе симуляции, но и тогда, когда нужно передвинуть головку машины

Тьюринга налево. Именно для этого нам потребовалось модифицировать детерминированный вариант задачи Поста. Если M — детерминированная машина Тьюринга, то после такого «неправильного» применения правила 1 мы попадём в ситуацию, когда в Γ_M нет ни одного применимого правила. Поэтому модифицированное вариант с задачей симуляции ДТМ справится успешно. □

По лемме 2.3, существует такая конечная система правил подстановки Γ_M, что $\underline{sxB} \vdash^{*,n^4}_{R_M} \underline{hyB}$ эквивалентно $g(x) = y$. Следовательно, на константной доле входов обращение функции ФПП эквивалентно обращению функции g. □

Замечание 2.1. Отметим небольшое изменение в распределениях на входах и выходах: ФПП получает на вход $\underline{x}$ и выдаёт $\underline{y}$, в то время как моделируемая машина Тьюринга g получает x и выдаёт y. Такие «мелкие детали» в теории сложности в среднем зачастую делают утверждения некорректными. К счастью, распределения на x и $\underline{x}$ могут быть трансформированы друг в друга полиномиальным алгоритмом, и поэтому ФПП даже с модифицированным входом остаётся слабо односторонней функцией (более подробно эта ситуация рассмотрена в [50]).

2.7 Полные односторонние функции и DistNP-трудные комбинаторные задачи

Предложенные в настоящей работе конструкции полных односторонних функций очень похожи и друг на друга, и на исходную конструкцию полной односторонней функции Левина. Естественно, возникает вопрос: в каких ещё комбинаторных задачах можно найти такие же полные односторонние функции?

Все рассматривавшиеся функции были основаны на комбинаторных проблемах, из которых также получаются и DistNP-трудные за-

дачи [63, 133]. Однако конструкция полных односторонних функций получается не вполне прямолинейной — нужен ещё метод Левина. Основная суть, главная идея построения полной односторонней функции, которой метод Левина и подчинён, заключается в том, чтобы функция оставалась сохраняющей длину и легко вычислимой. Очевидные труднообратимые функции делятся на два типа.

1. *Легко вычислимые, но не сохраняющие длину.* Для каждой DistNP-трудной задачи можно без труда построить функцию f, которая отображает *протоколы* решения задачи в результат. Эту функцию будет трудно обратить; в том числе её будет трудно обратить в среднем. Но поскольку она не сохраняет длину (протокол гораздо длиннее условия), не получится транслировать равномерное распределение на *выходах* функции f в разумное распределение на её *входах*. Можно задать вопрос о том, как определить «разумно равномерное» распределение на *корректных* замощениях квадрата (т.е. таких замощениях, в которых все символы на соседних сторонах плиток совпадают), из которого получалось бы «разумно равномерное» распределение на верхних строках этих замощений. Нам представляется, что такое распределение либо невозможно построить, либо для этого требуется принципиально новая техника.
2. *Сохраняющие длину, но трудно вычислимые.* Рассмотрим некоторую DistNP-трудную задачу и рассмотрим функцию, которая отображает её вход в её выход (например, нижнюю строку замощённого квадрата в верхнюю строку). Эту функцию трудно обратить, и она сохраняет длину, то есть с распределениями проблем не будет. Но её при этом и вычислить трудно: чтобы её вычислить, нужно решить задачу замощения; поэтому на роль односторонней функции она тоже не подходит.

Вслед за Л. А. Левиным мы справляемся с этими ограничениями, рассматривая *детерминированную версию* DistNP-трудной задачи. На этот раз задача замощения имеет нетривиальный ответ только в том случае, если всегда есть *ровно одна* подходящая плитка. Получается функция второго из вышеописанных типов, но при этом её легко вычислить. Аналогично, в разделе 2.5 мы требовали, чтобы на каждом шаге было ровно одно применимое правило подстановки (для этого мы ввели $\Rightarrow^*$). А в разделе 2.6 прямолинейная детерминированная аналогия оказалась недостаточно выразительной, и нам пришлось слегка обобщить понятие «детерминированного выбора», разрешив перебрать фиксированное число вариантов.

В итоге получается, что у комбинаторной задачи должны выполняться два свойства, чтобы из неё получалась полная односторонняя функция.

1. У задачи должна быть детерминированная версия, в которой вычисление результата можно получить за полиномиальное время.
2. Эта детерминированная версия должна быть достаточно выразительной, чтобы симулировать детерминированную машину Тьюринга. Например, полностью детерминированная задача соответствия Поста (без перебора вариантов) отвечает первому условию, но является недостаточно выразительной (стандартное вложение может промоделировать только машины Тьюринга, в которых головка движется всё время вправо).

Эти условия кажутся настолько строгими, что их хочется формализовать в виде математического утверждения. Такую формализацию мы опишем в следующем разделе.

2.8 Полные односторонние функции в большей общности

Прежде всего нужно определить, что такое комбинаторная задача. Заметим, что все наши примеры сводились к конечному множеству некоторых правил, по которым одна конфигурация задачи переходила в другую (например, под действием правила подстановки одна строка переводилась в другую строку полусистемой Туэ). Эта интуиция и ляжет в основу общего определения.

Определение 2.5 *Комбинаторная задача* S — это кортеж

$$S = \langle X, f_1, \ldots, f_m \rangle,$$

где X — некоторое множество (множество конфигураций), а $f_i : X \to X$ — частично заданные функции.

Определение следует понимать так: если конфигурация $x \in X$ лежит в области определения функции f_i, это значит, что правилом f_i её можно перевести в $f_i(x)$.

Пример 2.1. Для задачи соответствия Поста, заданной парами строк

$$(u_1, v_1), \ldots, (u_m, v_m),$$

$X = \{0,1\}^* \times \{0,1\}^*$, а функция $f_i : \{0,1\}^* \to \{0,1\}^*$ для каждого $1 \leq i \leq m$ переводит пару строк (a, b) в пару строк $(u_i a, b v_i)$.

Теперь определим, что такое «комбинаторная задача симулирует детерминированную машину Тьюринга». Это в предложенных определениях тоже сделать несложно. Через $t_M : \Gamma^* \times Q \times \mathbb{N} \to \Gamma^*$ (от слова tape) обозначим функцию, отражающую преобразование, которое машина Тьюринга делает с лентой. Строка $t_M(\gamma, q, pos)$ совпадает с γ везде, кроме позиции с индексом pos; там появляется вторая компонента результата функции перехода $\pi(q, \gamma_{pos})$.

Определение 2.6 Семейство комбинаторных задач $\mathcal{S}$ *симулирует детерминированные машины Тьюринга*, если для всякой машины

Тьюринга

$$M = \langle Q, \Gamma, B, \Sigma, \pi, s, H \rangle$$

существует такая комбинаторная задача $S = \langle X, f_1, \ldots, f_m \rangle \in \mathcal{S}$ и такое отображение $\sigma : Q \times \Gamma^* \to X$, что

$$\pi(q, \alpha) = (q', \alpha', \mathrm{move}), \text{ где } \mathrm{move} \in \{L, R\},$$

тогда и только тогда, когда для каждой строки $\gamma \in \Gamma^*$ и каждой такой позиции $\mathrm{pos} \in \mathbb{N}$, для которой $\gamma_{\mathrm{pos}} = \alpha$, существует единственное i, $1 \leq i \leq m$, для которого $\sigma(q, \gamma) \in \mathrm{dom} f_i$, и

$$f_i(\sigma(q, \gamma)) = \sigma(q', t_M(\gamma, q, \mathrm{pos})).$$

Проще говоря, конфигурации комбинаторной задачи должны суметь закодировать (в определении кодирование происходит посредством отображения σ) каждое состояние машины Тьюринга плюс записанное в текущий момент на ленте слово, причём то, куда функции f_i могут перевести эту конфигурацию, должно быть согласовано с функцией перехода машины Тьюринга.

Теперь введём по аналогии с рассматривавшимися в предыдущих разделах отношениями отношение выводимости на комбинаторной задаче $S = \langle X, f_1, \ldots, f_m \rangle$. Для двух конфигураций $x, y \in X$ будем говорить, что y *выводится из* x *за один шаг* и записывать $x \Rightarrow_S y$, если существует такой индекс $i \in 1..m$, что $x \in \mathrm{dom} f_i$, и $f_i(x) = y$. Введём также его «детерминированный вариант» $\Rightarrow^*_S$: $x \Rightarrow^*_S y$ за один шаг, если существует ровно один такой индекс $i \in 1..m$, что $x \in \mathrm{dom} f_i$, и $f_i(x) = y$. Как и ранее, замкнём отношения $\Rightarrow_S$ и $\Rightarrow^*_S$ по транзитивности. Будем также записывать $x \Rightarrow^{*,n}_S y$, если $x \Rightarrow^*_S y$ не более чем за n шагов.

Чтобы построить полную одностороннюю функцию, теперь достаточно только позаботиться о том, чтобы все эти процедуры с комби-

наторными задачами можно было проделать за полиномиальное время.

Определение 2.7 Семейство комбинаторных задач $\mathcal{S}$ называется *эффективно кодируемым*, если существует такое отображение $\rho : \mathcal{S} \to \{0,1\}^*$, такой полином p и такая машина Тьюринга $M_{\mathcal{S}}$, что для всякой задачи $S = \langle X, f_1, \ldots, f_m \rangle \in \mathcal{S}$ длина кода $|\rho(S)| \leq p(m)$, а машина $M_{\mathcal{S}}$ для всякого $i \in 1..m$ и состояния $x \in \mathrm{dom} f_i$ за полиномиальное число шагов вычисляет функцию

$$\rho(x) \mapsto \rho(f_i(x)).$$

Например, для полусистем Туэ эффективным кодированием будет просто кодирование левых и правых частей правил подстановки. Машина M должна будет прочесть строчку, найти вхождение нужной левой части и произвести подстановку.

Теперь можно определить функцию, которая окажется полной односторонней.

Определение 2.8 Для некоторого эффективно кодируемого посредством функции кодирования $\rho_{\mathcal{S}}$ семейства комбинаторных задач $\mathcal{S}$, симулирующего детерминированные машины Тьюринга, его *функция симуляции машины Тьюринга* (ФСМТ) $f : \{0,1\}^* \to \{0,1\}^*$ определяется следующим образом:

— если вход представляет собой пару $(\rho_{\mathcal{S}}(S), x)$ для некоторой $S \in \mathcal{S}$, то:

- если $x \Rightarrow_S^{*,n^2} y$ для некоторого y, причём для любого $i \in 1..m$ $y \notin \mathrm{dom} f_i$, то $f(\rho_{\mathcal{S}}(S), x) = (\rho_{\mathcal{S}}(S), y)$;
- в противном случае, f действует тождественно;

— в противном случае, f действует тождественно.

Теорема 2.4 Если односторонние функции существуют, то ФСМТ каждого эффективно кодируемого семейства комбинаторных задач $\mathcal{S}$,

симулирующего детерминированные машины Тьюринга, является односторонней функцией (точнее говоря, слабо односторонней).

Доказательство. С константной вероятностью (для равномерного распределения эта вероятность пропорциональна $\frac{1}{|p(S)|^2 2^{|p(S)|}}$) эффективное кодирование любой из комбинаторных задач $S \in \mathcal{S}$ появится в качестве первой части входа. Рассмотрим сохраняющую длину одностороннюю функцию g и машину Тьюринга M_g, которая вычисляет g за квадратичное время (они существуют по предположению теоремы и предложению 1.1). Тогда, по определению 2.6, существует комбинаторная задача $S_g \in \mathcal{S}$, симулирующая работу машины Тьюринга M_g. Следовательно, на константной доле входов обращение ФСМТ эквивалентно обращению g. □

Глава 3

Новые конструкции криптографических примитивов, доказуемо надёжных в слабом смысле

3.1 Введение

В современной криптографии, как мы уже упоминали, практически нет доказуемо надёжных конструкций с открытым ключом. Начиная с первого протокола согласования ключа Диффи-Хеллмана [39] и первой криптосистемы с открытым ключом RSA [107], не была доказана надёжность ни одного криптографического протокола с открытым ключом (разумеется, как мы уже упоминали, существуют надёжные протоколы с секретным ключом, например одноразовые блокноты [113, 129]). Конечно, безусловное доказательство надёжности было бы трудно получить, ведь из него неизбежно следовало бы, что $P \neq NP$. Но нет и доказательств условных, в естественных структурных предположениях. Недавно появившиеся разработки основанных на решётках криптосистем, связывающих криптографическую надёжность со сложностью в худшем случае, в действительности имеют дело с задачами, NP-трудность которых не доказана и, более того, маловероятна [4, 42, 104, 105].

Известны полные конструкции криптографических примитивов: как односторонних функций (см. главу 2), так и криптосистем с открытым ключом [55, 66]. Однако они тоже не позволяют связать криптографическую надёжность с ключевыми предположениями традиционной структурной теории сложности; более того, асимптотическая природа имеющихся утверждений о полноте не позволяет утверждать, что ту или иную криптографическую конструкцию трудно взломать для ключей какой-либо фиксированной длины. Кажется, что класси-

ческой современной криптографии ещё очень далеко до каких-либо *доказуемо* надёжных конструкций.

Но если мы не можем доказать, что для взлома криптографических протоколов противнику потребуется работать более чем полиномиально дольше, чем честным участникам, может быть, у нас получится доказать *хоть какую-нибудь* разницу между честным декодированием и взломом? В 1992 году Ален Хильтген [69] сконструировал функцию, для которой сумел доказать некоторую разницу между сложностью её вычисления и обращения: функцию Хильтгена почти *вдвое* (в $2 - o(1)$ раз) труднее обратить, чем вычислить. Его пример является линейной функцией над $GF(2)$ с матрицей, у которой мало ненулевых элементов, в то время как в обратной к ней матрице ненулевых элементов много. Сложность обращения при этом следует из простого наблюдения Ламаньи и Сэвиджа [82, 109]: каждый бит выхода нетривиально зависит от многих переменных, и эти биты все разные, следовательно, можно доказать нижнюю оценку на сложность вычисления их всех вместе взятых. Модель вычислений в данном случае была наиболее общей из возможных: количество гейтов в булевой схеме, которая использует произвольные бинарные булевские гейты. Мы уже отмечали, что в этой модели значительно бóльшего ожидать и не приходится: наилучшие нижние оценки схемной сложности в этой модели линейны от количества входов, да и константы там совсем небольшие: так, лучшая нижняя оценка для булевской функции $\mathbb{B}^n \to \mathbb{B}$ составляет всего лишь $3n - o(n)$ [24, 137].

Основной результат этой главы — конструкция другого криптографического примитива — семейства функций с секретом — с безусловным доказательством (столь же слабой) надёжности [144].

Для того чтобы получить этот результат, нам придётся доказать нижнюю оценку на схемную сложность функций определённого

вида; для этого мы будем использовать метод *исключения гейтов* (gate elimination), на котором основаны практически все существующие нижние оценки в схемной сложности [24, 97, 124].

Наша конструкция состоит из двух частей, двух функций, соединённых в одну прямой суммой. Для первой из этих функций задача противника (следуя традиции, мы будем называть его «Чарли») сложнее, чем задача шифрующего сообщение отправителя («Боба»), а для второй функции противнику приходится хуже, чем дешифрующему получателю («Алисе»). Мы докажем, что если одна часть сообщения закодирована первым способом, а другая — вторым, то задача Чарли будет сложнее, чем любая из задач Алисы и Боба. Формально говоря, сложности дешифровки для противника будет по крайней мере в $\frac{25}{22}$ раза больше, чем сложности честной шифровки, дешифровки и генерации ключей.

Глава организована следующим образом. В разделе 3.2 мы приведём основные определения понятий, используемых в этой главе. В разделе 3.3 будут установлены некоторые комбинаторные свойства матриц, которые будут представлять собой кандидаты на звание сложных функций. В разделе 3.4 мы представим базовый метод исключения гейтов, который будет позже использован для доказательства нижних оценок, а также применяем его в контексте функций, односторонних в слабом смысле. Наконец, в разделах 3.5 и 3.6 мы представим конструкцию семейства функций с секретом, доказуемо надёжных в слабом смысле.

3.2 Определения

В этой главе мы используем модель вычислений, описанную в разделе 1.3, а именно схемную сложность. Мы хотим, чтобы размер *схем*, взламывающих наше семейство функций с секретом, оказался больше

размера схем, которые выполняют кодирование.

А. Хильтген ввёл для каждой *обратимой* функции n переменных $f \in \mathbb{B}_{n,m}$ понятие *меры сложности в слабом смысле* (measure of feeble one-wayness)

$$M_F(f) = \frac{C(f^{-1})}{C(f)}$$

(напомним, что $C(f)$ — размер минимальной схемы с бинарными гейтами, реализующей f). Результаты Хильтгена заключались в том, что он нашёл последовательности функций $\{f_n\}_{n=1}^{\infty}$ с нетривиальными (т.е. большими единицы) константами

$$\lim_{n\to\infty} \inf M_F(f_n),$$

которые Хильтген называет *порядком необратимости* (order of one-wayness). Его результаты мы подробнее обсудим в разделе 3.4.

В 1.3 мы ввели определение и кратко обсудили понятие семейства функций с секретом. Здесь нам потребуется ввести новое, чуть более детальное определение, потому что в контексте сложности в слабом смысле важна каждая константа. Поскольку определение 3.1 ничего не утверждает о собственно сложности и необратимости, мы назвали эту конструкцию *кандидатом в функции с секретом*.

Определение 3.1 Зафиксируем функции $pi, ti, m, c : \mathbb{N} \to \mathbb{N}$. *Семейство кандидатов в функции с секретом* представляет собой последовательность троек $\mathcal{C} = \{(Key_n, Eval_n, Inv_n)\}_{n=1}^{\infty}$, где:

— $\{Key_n\}_{n=1}^{\infty}$ — это семейство схем порождения ключей

$$Key_n : \mathbb{B}^n \to \mathbb{B}^{pi(n)} \times \mathbb{B}^{ti(n)},$$

— $\{Eval_n\}_{n=1}^{\infty}$ — это семейство вычисляющих функцию схем

$$Eval_n : \mathbb{B}^{pi(n)} \times \mathbb{B}^{m(n)} \to \mathbb{B}^{c(n)}, \text{ а}$$

— $\{Inv_n\}_{n=1}^{\infty}$ — это семейство обращающих функцию схем

$$Inv_n : \mathbb{B}^{ti(n)} \times \mathbb{B}^{c(n)} \to \mathbb{B}^{m(n)},$$

причём для каждого n, каждого $s \in \mathbb{B}^n$ (начального числа генератора) и каждого $m \in \mathbb{B}^{m(n)}$ (сообщения)

$$\mathrm{Inv}_n(\mathrm{Key}_{n,2}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) = m,$$

где $\mathrm{Key}_{n,1}(s)$ и $\mathrm{Key}_{n,2}(s)$ — первые $\mathrm{pi}(n)$ бит («публичная информация», public information) и последние $\mathrm{ti}(n)$ бит («секрет», trapdoor information) выхода схемы $\mathrm{Key}_n(s)$, соответственно.

Сейчас мы менее формально объясним смысл определения 3.1. Число n — это параметр надёжности функции с секретом, длина начального числа генератора случайных чисел. Длину входа функции с секретом мы обозначили через $m(n)$, через $c(n)$ — длину её выхода, а через $\mathrm{pi}(n)$ и $\mathrm{ti}(n)$ — длину публичной информации и секрета соответственно. Мы называем такое семейство функций «кандидатом», потому что в определении 3.1 ничего не говорится о надёжности, а только вводятся обозначения для размерностей и устанавливается корректность обращения. Во всех конструкциях, встречающихся в этой главе, $m(n) = c(n)$ и $\mathrm{pi}(n) = \mathrm{ti}(n)$, но мы сочли разумным дать максимально общее определение.

Чтобы понять, насколько функция надёжна, нужно ввести понятие взлома функции. Неформально говоря, противник должен обратить функцию, не зная секрета. Мы вводим успешный взлом как обращение с вероятностью более $1/2$ (позже мы рассмотрим и противников, имеющих меньшие вероятности успеха).

Через $C_{1/2}(f)$ мы будем обозначать минимальный размер схемы, которая вычисляет функцию $f \in \mathbb{B}_{n,m}$ на более чем половине её входов (длины n). Очевидно, $C_{1/2}(f) \leq C(f)$ для всех функций f.

Определение 3.2 Схема N *успешно обращает* семейство кандидатов в функции с секретом $\{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ на входах длины n, если для равномерного распределения U, взятого по $s \in \mathbb{B}^n$ и $m \in \mathbb{B}^{m(n)}$ (по

начальным числам генератора и входам),

$$\Pr_{(s,m)\in U}[N(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) = m] > \frac{1}{2}.$$

Замечание 3.2. На самом деле мы в дальнейшем докажем оценки на противников даже с более слабыми условиями обращения: мы докажем, что ни одна схема не сможет успешно обратить построенное нами семейство функций *ни для какого начальноо числа генератора* s, то есть в худшем случае по s, а не в среднем.

Для того чтобы семейство функций с секретом было надёжным, в контексте надёжности в слабом смысле достаточно, чтобы размер обращающих схем был хоть немного (в константное число раз) больше, чем размер схем, участвующие в его вычислении.

Определение 3.3 Будем говорить, что семейство кандидатов в функции с секретом $\mathcal{C} = \{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ имеет *порядок надёжности* $k \in \mathbb{R}$, если для каждой такой последовательности схем $\{N_n\}_{n=1}^{\infty}$, в которой N_n успешно обращает $\mathcal{C}$ на входах длины n,

$$\liminf_{n\to\infty} \min\left\{\frac{C(N_n)}{C(\mathrm{Key}_n)}, \frac{C(N_n)}{C(\mathrm{Eval}_n)}, \frac{C(N_n)}{C(\mathrm{Inv}_n)}\right\} \geq k.$$

Иначе говоря,

$$\liminf_{n\to\infty} \min\left\{\frac{C_{1/2}(f_{pi(n)+c(n)})}{C(\mathrm{Key}_n)}, \frac{C_{1/2}(f_{pi(n)+c(n)})}{C(\mathrm{Eval}_n)}, \frac{C_{1/2}(f_{pi(n)+c(n)})}{C(\mathrm{Inv}_n)}\right\} \geq k,$$

где функция $f_{pi(n)+c(n)} \in \mathbb{B}_{pi(n)+c(n),m(n)}$ отображает

$$(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) \mapsto m.$$

Пример 3.1. Приведём простейшие замечания-примеры. Если секретного ключа нет вовсе ($pi(n) = 0$), то каждое такое семейство кандидатов в функции с секретом $\{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ имеет порядок надёжности 1, т.к. последовательность схем $\{\mathrm{Inv}_n\}_{n=1}^{\infty}$ успешно его обращает. Если $\{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ реализуют функцию с секретом в обычном криптографическом, не в слабом смысле, то $k = \infty$.

Более того, $k = \infty$, просто если оценки на размер противника более чем линейны, например, если противнику требуется $O(n \log n)$ гейтов. К сожалению, даже таких оценок на размер схем из произвольных бинарных гейтов, реализующих конкретные булевские функции, пока не известно.

Замечание 3.3. Можно было бы рассмотреть порождение ключей как отдельный процесс и исключить его сложность из определения порядка надёжности. Однако мы доказываем наши результаты для этого определения, и они от этого становятся только сильнее.

Замечание 3.4. Следует явно отметить, что мы говорим исключительно об *единовременной* надёжности. Противник может использовать меньшее количество гейтов, обращая семейство кандидатов в функции с секретом второй раз для той же пары ключей: например, он может вычислить секретную информацию и успешно использовать её вторично. Таким образом, в наших конструкциях требуется для каждого нового входа выбирать новую пару ключей (фактически, новое начальное число генератора).

В следующих разделах мы разработаем конструкцию функции с секретом, надёжной в слабом смысле, т.е. семейства кандидатов в функцию с секретом, у которого будет нетривиальный порядок надёжности.

3.3 Матрицы сложных функций

Все наши конструкции основаны на линейной функции $f : \mathbb{B}^n \to \mathbb{B}^n$, для которой А. Хильтген доказал, что она является односторонней в слабом смысле с порядком надёжности $3/2$ [69]. Мы ограничимся случаем, когда $n \equiv 0 \bmod 4$, по причинам, которые будут изложены чуть ниже. Заметим, что от этого случая мы сможем без проблем перейти к произвольному n, не потеряв в надёжности в смысле определения 3.3:

для $n \not\equiv 0 \bmod 4$ можно просто рассмотреть схему, длина входа которой представляет собой наименьшее делящееся на 4 число, превосходящее n.

В дальнейшем все вычисления производятся над $\mathbb{F}_2$. Введём стандартные матричные обозначения:

- $\boldsymbol{e}_k$ — единичная матрица $k \times k$;
- $\boldsymbol{0}_k$ — нулевая матрица $k \times k$;
- $\boldsymbol{e}_{ij}$ — матрица, единственный ненулевой элемент которой находится на позиции (i, j);
- $\boldsymbol{e}_{i*}$ — матрица, в которой i-я строка состоит из единиц, а остальные из нулей;
- $\boldsymbol{e}_{*j}$ — матрица, в которой j-й столбец состоит из единиц, а остальные из нулей;
- $\boldsymbol{1}_k$ — матрица $k \times k$, состоящая из единиц;
- $\boldsymbol{u}_k$ — верхняя треугольная матрица $k \times k$ ($u_{ij} = 1 \Leftrightarrow i < j$);
- $\boldsymbol{l}_k$ — нижняя треугольная матрица $k \times k$ ($l_{ij} = 1 \Leftrightarrow i > j$);
- $\boldsymbol{m}_\pi$ — матрица перестановки π ($m_{ij} = 1 \Leftrightarrow j = \pi(i)$).

Через $\boldsymbol{e}$, $\boldsymbol{0}$, $\boldsymbol{1}$, $\boldsymbol{u}$ и $\boldsymbol{l}$ без индексов будем обозначать соответствующие матрицы размерности $\frac{n}{2} \times \frac{n}{2}$. Положим также $\sigma = (1\ 2\ \ldots\ n)$.

В этих обозначениях матрица функции Хильтгена f выглядит так:

$$A = \boldsymbol{e}_n + \boldsymbol{m}_\sigma + \boldsymbol{e}_{n,\frac{n}{2}+1}.$$

Лемма 3.1 Пусть $n = 4k$ для некоторого $k \in \mathbb{N}$. Тогда верны следующие равенства:

$$\begin{aligned} A^{-1} &= \boldsymbol{1}_n + \begin{pmatrix} \boldsymbol{e} + \boldsymbol{u} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{l} \end{pmatrix}, \\ A^{-2} &= \boldsymbol{1}_n \boldsymbol{u}_n + \boldsymbol{u}_n \boldsymbol{1}_n + \begin{pmatrix} \boldsymbol{e} + \boldsymbol{u}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{l}^2 \end{pmatrix}. \end{aligned}$$

Доказательство. 1. Заметим, что

$$m_\sigma \begin{pmatrix} e+u & 0 \\ 0 & l \end{pmatrix} = \begin{pmatrix} u & 0 \\ e_{\frac{n}{2}*} & e+l+e_{\frac{n}{2}*} \end{pmatrix} = \begin{pmatrix} u & 0 \\ 0 & e+l \end{pmatrix} + e_{n*},$$

$$e_{n,\frac{n}{2}+1} 1 = e_{n*}, \text{ и}$$

$$e_{n,\frac{n}{2}+1} \begin{pmatrix} e+u & 0 \\ 0 & l \end{pmatrix} = 0,$$

т.к. в матрице $\begin{pmatrix} e+u & 0 \\ 0 & l \end{pmatrix}$ строка с номером $\frac{n}{2}+1$ нулевая. Кроме того, для любой матрицы x над $\mathbb{F}^2$

$$x + x = 0,$$

для любой перестановки $\pi \in S_k$

$$m_\pi 1_k = 1_k m_\pi = 1_k,$$

а при $n \equiv 0 \pmod 4$ верно, что $1^2 = 0$. Теперь видно, что

$$(e_n + m_\sigma + e_{n,\frac{n}{2}+1}) \left(1_n + \begin{pmatrix} e+u & 0 \\ 0 & l \end{pmatrix}\right) =$$

$$= 1_n + \begin{pmatrix} e+u & 0 \\ 0 & l \end{pmatrix} + 1_n + \begin{pmatrix} u & 0 \\ 0 & e+l \end{pmatrix} + e_{n*} + e_{n*} + 1 = e_n.$$

Произведение в обратном порядке вычисляется аналогично.

2. Заметим, что при $n \equiv 0 \bmod 4$

$$(u+l)1 = 1 = 1(u+l),$$

и, следовательно, $\mathfrak{l}\mathfrak{1} = \mathfrak{1} + \mathfrak{u}\mathfrak{1}$, $\mathfrak{1}\mathfrak{l} = \mathfrak{1} + \mathfrak{1}\mathfrak{u}$. Теперь видно, что

$$\begin{aligned}
&\left(\mathfrak{1}_n + \begin{pmatrix} \mathfrak{e}+\mathfrak{u} & 0 \\ 0 & \mathfrak{l} \end{pmatrix}\right)^2 = \\
&= \mathfrak{1}^2 + \begin{pmatrix} \mathfrak{1}+\mathfrak{1}\mathfrak{u} & \mathfrak{1}\mathfrak{l} \\ \mathfrak{1}+\mathfrak{1}\mathfrak{u} & \mathfrak{1}\mathfrak{l} \end{pmatrix} + \begin{pmatrix} \mathfrak{u}\mathfrak{1}+\mathfrak{1} & \mathfrak{u}\mathfrak{1}+\mathfrak{1} \\ \mathfrak{l}\mathfrak{1} & \mathfrak{l}\mathfrak{1} \end{pmatrix} + \begin{pmatrix} (\mathfrak{e}+\mathfrak{u})^2 & 0 \\ 0 & \mathfrak{l}^2 \end{pmatrix} = \\
&= \begin{pmatrix} \mathfrak{1}\mathfrak{u}+\mathfrak{u}\mathfrak{1} & \mathfrak{1}\mathfrak{u}+\mathfrak{u}\mathfrak{1} \\ \mathfrak{1}\mathfrak{u}+\mathfrak{u}\mathfrak{1} & \mathfrak{1}\mathfrak{u}+\mathfrak{u}\mathfrak{1} \end{pmatrix} + \begin{pmatrix} \mathfrak{e}+\mathfrak{u}^2 & 0 \\ 0 & \mathfrak{l}^2 \end{pmatrix} = \\
&= \mathfrak{1}_n\mathfrak{u}_n + \mathfrak{u}_n\mathfrak{1}_n + \begin{pmatrix} \mathfrak{e}+\mathfrak{u}^2 & 0 \\ 0 & \mathfrak{l}^2 \end{pmatrix}.
\end{aligned}$$

□

Нас также интересует матрица $\mathcal{A}$ размером $n \times 2n$, состоящая из A^{-2} и A^{-1}, расположенных друг за другом:

$$\mathcal{A} = \begin{pmatrix} A^{-2} & A^{-1} \end{pmatrix}.$$

Нам потребуются следующие свойства A^{-1} и $\mathcal{A}$.

Лемма 3.2 Пусть $n = 4k$ для некоторого $k \in \mathbb{N}$. Тогда:

1. Все столбцы $\mathcal{A}$ (и, следовательно, A^{-1}) различны.
2. В каждой строке матрицы A^{-1} (соответственно, $\mathcal{A}$) есть по меньшей мере $\frac{n}{2}$ (соответственно, $\frac{5n}{4}$) ненулевых элементов.
3. После удаления всех, кроме двух (соответственно, всех, кроме пяти) столбцов матрицы A^{-1} (соответственно, $\mathcal{A}$) в ней останется по крайней мере одна строка с двумя ненулевыми элементами.

Доказательство. Начнём доказательство с того, что проинтерпретируем результаты Леммы 3.1. В каждой строке A стоят по две единицы (на диагонали и справа от неё), кроме последней строки, в которой единиц три: в $(n, 1)$, $(n, \frac{n}{2} + 1)$ и (n, n). В каждой строке A^{-1} есть по меньшей мере $\frac{n}{2}$ ненулевых элементов (единиц), а $(\frac{n}{2} + 1)$-я строка и вовсе нулей не содержит.

В матрице A^{-2} тоже много единиц: матрица $(\mathbf{1}_n\mathbf{u}_n + \mathbf{u}_n\mathbf{1}_n)$ представляет собой матрицу $n \times n$, заполненную нулями и единицами в шахматном порядке, т.к.

$$(\mathbf{1}_n\mathbf{u}_n)_{ij} = 1 \quad \Leftrightarrow \quad j \equiv 1 \pmod 2,$$
$$(\mathbf{u}_n\mathbf{1}_n)_{ij} = 1 \quad \Leftrightarrow \quad i \equiv 0 \pmod 2.$$

Более того,

$$(\boldsymbol{e} + \mathbf{u}^2)_{ij} = 1 \quad \Leftrightarrow \quad j > i \quad \text{и} \quad i + j \equiv 0 \pmod 2,$$
$$(\mathbf{l}^2)_{ij} = 1 \quad \Leftrightarrow \quad i > j \quad \text{и} \quad i + j \equiv 0 \pmod 2,$$

поэтому в A^{-2} есть два треугольных блока, заполненных единицами: для $1 \leq i \leq j \leq \frac{n}{2}$ и для $\frac{n}{2} + 1 < j < i \leq n$. Таким образом, у матрицы A^{-2} каждая строка содержит о меньшей мере $\frac{n}{2}$ единиц; более того, её треугольные блоки, состоящие из единиц, совпадают с треугольными блоками матрицы A^{-1}, состоящими из нулей, а остальное пространство покрыто нулями и единицами в шахматном порядке.

Первое утверждение очевидно.

Строка матрицы A^{-1} с номером i содержит $\frac{n}{2} + i$ ненулевых элементов для $i \leq \frac{n}{2}$ и $\frac{n}{2} + n - i$ ненулевых элементов для $i \geq \frac{n}{2}$. Таким образом, второе утверждение выполняется для матрицы A^{-1}. В то же самое время, i-я строка A^{-2} содержит по меньшей мере $\frac{3n}{4} - \frac{i}{2}$ ненулевых элементов для $i \leq \frac{n}{2}$ и по меньшей мере $\frac{n}{2} + \frac{1}{2}(i - \frac{n}{2} - 1)$ ненулевых элементов для $i > \frac{n}{2}$. Поэтому i-я строка A^{-2} содержит по меньшей мере

$$\frac{n}{2} + i + \frac{3n}{4} - \frac{i}{2} = \frac{5n}{4} + \frac{i}{2}$$

ненулевых элементов для $i \leq \frac{n}{2}$ и по меньшей мере

$$\frac{n}{2} + n - i + \frac{n}{2} + \frac{1}{2}(i - \frac{n}{2} - 1) = \frac{7n}{4} - \frac{1}{2}(i - 1) \geq \frac{5n}{4}$$

ненулевых элементов для $i > \frac{n}{2}$.

Докажем теперь третий пункт леммы. Поскольку в A^{-1} есть строка, целиком состоящая из единиц, придётся удалить все столбцы этой матрицы, кроме одного, чтобы в строке осталась только одна единица. То же самое верно для левой части матрицы A^{-2} (обратите внимание на её первую строку), а также для правой части матрицы A^{-2}, за исключением её последнего столбца (обратите внимание на её последнюю строку). □

3.4 Исключение гейтов

В этом разделе мы сначала кратко напомним методы доказательства оценок, использованные в работах Хильтгена, а затем введём метод исключения гейтов как основной метод доказательства нижних оценок в этой главе. Хильтген доказал все свои оценки при помощи следующего весьма простого наблюдения, впервые сделанного Ламаньей и Сэвиджем (мы приводим доказательство, потому что его трудно найти в литературе).

Утверждение 3.1 ([82, 109]; [69, Теоремы 3 и 4])

1. Предположим, что функция $f : \mathbb{B}^n \to \mathbb{B}$ нетривиально зависит от каждой из n своих переменных, то есть для любого i, $1 \leq i \leq n$, найдутся такие значения $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{B}$, что

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

   Тогда $C(f) \geq n - 1$.
2. Пусть $f = (f^{(1)}, \ldots, f^{(m)}) : \mathbb{B}^n \to \mathbb{B}^m$, где $f^{(k)}$ — это k-я компонента функции f. Если все m функций–компонент $f^{(i)}$ попарно различны, и для каждой из них $C(f^{(i)}) \geq c \geq 1$, то $C(f) \geq c + m - 1$.

Доказательство. 1. Рассмотрим минимальную схему размера s, вычисляющую f. Поскольку f зависит (здесь и далее мы бу-

дем говорить «зависит», имея в виду «нетривиально зависит») от всех n своих переменных, у каждого из гейтов–входов должно быть по крайней мере одно исходящее ребро. Поскольку схема минимальна, у каждого из оставшихся гейтов в ней, кроме, возможно, выхода, тоже должно быть по крайней мере одно исходящее ребро. Поэтому в схеме должно быть по крайней мере $s + n - 1$ рёбер. С другой стороны, в схеме из s бинарных гейтов не может быть более чем $2s$ рёбер. Поэтому $2s \geq s + n - 1$.

2. Рассмотрим схему, вычисляющую f. Отметим, что в ней есть по меньшей мере $c - 1$ гейтов, не вычисляющих никакую функцию схемной сложности c или более (в качестве таких гейтов можно рассмотреть первые $c - 1$ гейтов в каком-либо топологическом порядке). Но для вычисления любой функции–компоненты $f^{(i)}$ требуется добавить по меньшей мере ещё один гейт, причём потребуется добавлять по одному гейту для каждой компоненты, ведь один новый гейт добавляет только одну новую функцию. Так и получается необходимая оценка в $c + m - 1$ гейтов.

□

Оценки Хильтгена доказаны следующим образом: сначала доказывается оценка на сложность вычисления одного бита выхода (например, поскольку в каждой строке A есть по крайней мере $\frac{n}{2}$ единиц, то минимальная сложность каждой компоненты $A\vec{y}$ составляет $\frac{n}{2}-1$), а из них получал нижние оценки на сложность вычисления всей функции, т.е. обращения A^{-1} (например, сложность вычисления $A\vec{y}$ составляет $\frac{n}{2} + n - 2 = \frac{3n}{2} - 2$).

Кроме того, в криптографии обычно желательно доказать не только оценки в худшем случае, но также тот факт, что противник неспособен обратить функцию на значительной доле её входов. Первым шагом к доказательству этого утверждения может стать следую-

щий факт.

Утверждение 3.2 Для каждой из матриц A, A^2, $\mathcal{A}$ любая схема, использующая менее чем минимально необходимое для точного вычисления соответствующей функции количество гейтов, обращает функцию на не более чем половине всех входов.

В работах Хильтгена этот факт доказывался при помощи следующего очень простого наблюдения (которое там даже явно не сформулировано).

Лемма 3.3 Рассмотрим функцию $f = \bigoplus_{i=1}^{n} x_i$. Для каждой функции g, которая нетривиально зависит только от m из этих переменных при $m < n$,

$$\Pr_{x_1,\ldots,x_n}[f(x_1,\ldots,x_n) = g(x_{i_1},\ldots,x_{i_m})] = \frac{1}{2}.$$

Доказательство. Поскольку $m < n$, существует такой номер $j \in 1..n$, что g не зависит от x_j. Это значит, что для каждого набора значений всех остальных переменных для одного из значений x_j результат вычисления g совпадает с результатом f, а для другого значения не совпадает. Это значит, что f отличается от g в точности на половине своих входов. □

Такого рассуждения вполне достаточно для необратимости в слабом смысле по Хильтгену для квадратной матрицы. A^{-1}: сначала мы применяем первую часть предложения 3.1 и получаем, что сложность каждого выхода по меньшей мере $\frac{n}{2} - 1$, а затем вторая часть предложения 3.1 даёт нам желаемую оценку в $\frac{3n}{2} - 1$ гейтов. Более того, если у схемы гейтов меньше, чем требуется, один из её выходов неизбежно должен будет зависеть от меньшего числа входных переменных, чем нужно. А это, по лемме 3.3, немедленно даёт нам долю ошибки в $\frac{1}{2}$.

Однако в этой главе мы пользуемся также и прямоугольными матрицами, и оказывается, что аналогичное простое рассуждение уже не так замечательно работает в этом случае. Поэтому нам придётся

использовать другой способ доказательства нижних оценок, а именно *исключение гейтов*, которое до сих пор было использовано во всех доказательствах нижних оценок в классической теории схемной сложности [137].

Основная идея этого метода — индуктивное рассуждение следующего вида. Рассмотрим функцию f и подставим некоторое значение c в некоторую переменную x, получив таким образом схему, вычисляющую функцию $f|_{x=c}$. Эту схему можно упростить, потому что гейты, в которые входила эта переменная, теперь либо стали унарными (а отрицание можно удалить из схемы, перенеся его роль на следующие гейты), либо и вовсе превратились в константы (и тогда можно будет удалять и их потомков). Важный случай здесь тот, когда гейт нелинеен, например гейт, вычисляющий конъюнкцию AND или дизъюнкцию OR. В этом случае всегда возможно выбрать такое значение для входа, что гейт станет константным. Этот процесс можно продолжать индуктивно, пока можно выбирать подходящую переменную, входящую в достаточное количество гейтов. Очевидно, количество исключённых в течение этого процесса гейтов представляет собой нижнюю оценку на схемную сложность f.

В нашей работе исключение гейтов использовано в одной из своих простейших форм. Вот главная лемма, которую мы позже применим к интересующим нас матрицам.

Лемма 3.4 Пусть $t, u \geq 1$. Предположим, что $\chi : \mathbb{B}^{\nu(n)} \to \mathbb{B}^n$ — это линейная функция с матрицей X над полем $GF(2)$. Предположим также, что:

- все столбцы X различны;
- в каждой строке X есть по меньшей мере u ненулевых элементов;
- после удаления любых t столбцов X в матрице всё ещё останется строка, содержащая по крайней мере два ненулевых элемента.

Тогда $C(\chi) \geq u + t$ и, более того, $C_{1/2}(\chi) \geq u + t$.

Доказательство. Рассмотрим схему, вычисляющую χ на более чем половине её входов. Зафиксируем некоторый топологический порядок на гейтах схемы (топологический порядок — это линейный порядок, в котором всякий гейт-потомок меньше гейта-предка). Мы будем обозначать функцию, которую схема вычисляет *на самом деле*, через h (h не обязательно линейна, но должна совпадать с χ на более чем половине входов).

Рассмотрим наибольший гейт g по зафиксированному порядку. Поскольку g — наибольший, входящие в него рёбра должны идти из входов схемы. Обозначим их через x и y. По лемме 3.3, ни один из этих входов не может одновременно являться выходом схемы, ведь даже после удаления первых $u - 1$ столбцов из матрицы X каждая строка будет содержать по меньшей мере две единицы, то есть каждый выход будет зависеть по меньшей мере от двух переменных.

Возможны следующие случаи.

1. Одна из переменных, входящих в g, например x, входит также в какой-нибудь другой гейт. В этом случае, подставив любое значение вместо x, мы сможем исключить два гейта.
2. g — гейт, вычисляющий нелинейную функцию, и ни x, ни y не подаются на вход другим гейтам. Если у g есть потомки, то, подставив в качестве x подходящую константу, мы сможем исключить и сам гейт g, и его потомков, то есть по меньшей мере два гейта.

   Если же у g нет потомков, это значит, что g является выходом; рассмотрим этот выход h_i. Если χ_i зависит от какой-либо другой переменной z, мы можем убедиться, что h_i отличается от χ_i по крайней мере на половине входов, меняя значения z. Для этого рассмотрим два различных значения переменной z, 0 и 1, и за-

фиксируем значения всех остальных переменных. Для одного из значений z функция h будет непременно давать неверный ответ, потому что

$$(h_i)_{z=0} = (h_i)_{z=1}$$

(h_i не зависит от z), но

$$(\chi_i)_{z=0,x=a,y=b} \neq (\chi_i)_{z=1,x=a,y=b},$$

потому что $\chi_i = z \oplus \ldots$. А если $\chi_i = x$, $\chi_i = y$, $\chi_i = x \oplus y$ или $\chi_i = x \oplus y \oplus 1$, мы можем заменить g на линейный гейт (а то и его отсутствие), который будет точно вычислять χ_i; при такой замене в схеме больше ничего не изменится, поскольку у g нет детей. Следовательно, мы попадаем в условия случая 3.

3. g вычисляет линейную функцию, и ни x, ни y не подаются на вход другим гейтам. Докажем, что этот случай невозможен.

   В этом случае h не зависит от x или y по отдельности, а зависит только от $x \oplus y$. Мы покажем, что в этом случае h отличается от χ по крайней мере на половине своих входов. Во-первых, заметим, что поскольку χ зависит от x и y по отдельности, существует выход χ_i, который зависит только от одной из этих переменных, например x (других возможностей быть не может, поскольку χ линейна). Тогда $\chi_i = x \oplus \bigoplus_{z \in Z} z$, где $y \notin Z$. Возможны два случая: либо выход h_i функции h зависит от $x \oplus y$, либо он не зависит ни от x, ни от y.

   В первом случае рассмотрим два различных означивания переменной y, 0 и 1, и зафиксируем значения всех остальных переменных. Для одного из означиваний y h будет непременно давать неверный ответ, потому что

$$h_i(\ldots, y = 0, x = a, \ldots) \neq h_i(\ldots, y = 1, x = a, \ldots)$$

(когда мы меняем значение y, мы тем самым меняем $x \oplus y$), но

$$\chi_i(\ldots, y = 0, x = a, \ldots) = \chi_i(\ldots, y = 1, x = a, \ldots),$$

потому что χ_i вовсе не зависит от y. Во втором случае рассуждение аналогично, но менять нужно значение x: h не изменит значение выхода, а χ изменит.

Итак, мы доказали, что если все строки матрицы A содержат больше одного ненулевого элемента, то существует переменная, подстановка значения в которую приводит к исключению двух гейтов. Будем называть такие переменные «хорошими».

Индуктивное применение этого утверждения с учётом третьего пункта леммы 3.2 позволяет исключить по крайней мере $2(u - 1)$ гейтов (заметим, что подстановка переменной эквивалентна удалению столбца из матрицы).

Что произойдёт, когда мы удалим все «хорошие» переменные? Мы продолжаем тот же процесс исключения гейтов[1]. Этот процесс можно продолжать до тех пор, пока в матрице присутствует хотя бы одна строка с двумя ненулевыми элементами: если хотя бы одна такая строка есть, значит, схема должна быть нетривиальной, и, значит, верхний гейт существует. Правда, теперь возможен ещё и случай (и в точном вычислении функции χ, и в её приближении), когда одна из входных переменных сразу же подаётся на выход всей функции, и мы, таким образом, сможем исключить за один шаг только один гейт. Однако исключение одного гейта после подстановки переменной нам гарантировано в любом случае. Таким образом, мы суммарно исклю-

[1] В доказательстве этой леммы его можно было бы и не продолжать, а сразу получить результат из утверждения 3.1, но мы позже используем это доказательство ещё раз в лемме 3.5.

чим по меньшей мере

$$2(u-1)+((t+1)-(u-1))=u+t$$

гейтов. □

В своих конструкциях мы будем использовать и блочно-диагональные матрицы. Конечно, «интуитивно ясно», что совместное вычисление двух функций, у которых ни входы, ни выходы не пересекаются, должно быть не проще, чем их вычисление по отдельности, а нижняя оценка должна получаться как сумма соответствующих нижних оценок. Но, как ни странно, этот факт в общем случае неверен, как показано, например, в [137, Section 10.2]. Поэтому мы рассмотрим только частный случай, для интересующих нас линейных функций.

Лемма 3.5 Пусть линейная функция ζ определяется блочно-диагональной матрицей

$$\zeta\left(\vec{x}^{(1)},\vec{x}^{(2)},\ldots,\vec{x}^{(m)}\right)=\begin{pmatrix}X_1 & 0 & \ldots & 0\\ 0 & X_2 & \ldots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \ldots & X_m\end{pmatrix}\begin{pmatrix}\vec{x}^{(1)}\\ \vec{x}^{(2)}\\ \vdots\\ \vec{x}^{(m)}\end{pmatrix},$$

и матрицы X_j удовлетворяют условиям леммы 3.4 с параметрами u_j и t_j, соответственно (матрицы могут быть прямоугольными и иметь разный размер). Тогда

$$C(\zeta)\geq\sum_{j=1}^{m}(u_j+t_j)$$

и, более того,

$$C_{1/2}(\zeta)\geq\sum_{j=1}^{m}(u_j+t_j).$$

Доказательство. Доказательство почти полностью повторяет доказательство леммы 3.4. Заметим, что когда мы подставляем переменную из $x^{(i)}$, она ничего не меняет в матрице X_j для $j\neq i$. Таким образом, мы подставляем «хорошие» переменные (которые позволяют

исключить два гейта) до тех пор, пока они есть, а затем подставляем «плохие» переменные (которые исключают только один гейт) *отдельно для каждой матрицы.*

Если одна из матриц тривиализуется, то есть в ней не остаётся строк с двумя ненулевыми элементами (это может произойти после того, как мы подставим в неё $u_i - 1$ «хороших», а затем $t_i - u_i + 2$ «плохих» переменных), мы просто подставим все остальные переменные, соответствующие этой части блочно-диагональной матрицы (не исключая ни одного гейта — матрица своё уже отработала) и забудем о ней.

Осталось рассмотреть случай, в котором один из входов в верхний гейт происходит из $x^{(i)}$, а другой — из $x^{(j)}$. Все три случая из доказательства леммы 3.4 проходят и в этой ситуации без изменений: в первом случае мы подставляем значение в «хорошую» переменную, во втором случае нелинейный гейт по-прежнему позволяет удалить два гейта, а третьего случае не может быть по тем же соображениям.

Таким образом, удаление всех столбцов из матрицы X_i приведёт к исключению по меньшей мере

$$2(u_i - 1) + (t_i - u_i + 2) = t_i + u_i$$

гейтов, и суммарно мы получим оценку

$$C_{1/2}(\zeta) \geq \sum_{j=1}^{m} (u_j + t_j).$$

□

Сформулируем теперь прямые следствия этих лемм и комбинаторные леммы о конкретных интересующих нас матрицах.

Лемма 3.6 Пусть $n, n' \equiv 0 \pmod 4$,

$$\alpha(\vec{x}) = A^{-1}\vec{x}, \quad \alpha_2(\vec{x}) = \begin{pmatrix} A^{-1} & A^{-2} \end{pmatrix} \vec{x}, \quad \alpha_*(\vec{x}) = \begin{pmatrix} A^{-1} & A^{-2} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \vec{x},$$

где A_*^{-1} обозначает матрицу с той же структурой, что и A^{-1}, но с размерностью n' вместо n. Тогда

$$C(\alpha) \geq \frac{3n}{2} - 2, \qquad C(\alpha_2) \geq \frac{13n}{4} - 5, \qquad C(\alpha_*) \geq \frac{3n'}{2} + \frac{13n}{4} - 7,$$

и всякая схема, в которой меньше указанного количества гейтов, не вычисляет соответствующую функцию более чем на половине её входов.

Доказательство. Следует из леммы 3.4 и леммы 3.5 подстановкой доказанных в лемме 3.2 оценок на $u(n)$ и $t(n)$ (в частности, $t = n - 2$ для матрицы A^{-1}, а для матрицы $\mathcal{A}$ оценка составляет $t = 2n - 5$). □

С другой стороны, верны следующие верхние оценки.

Лемма 3.7 Следующие утверждения верны.

1. Существует схема размера $\frac{3n}{2} - 1$, вычисляющая линейную функцию $\phi : \mathbb{B}^n \to \mathbb{B}^n$ с матрицей A^{-1}.
2. Существует схема размера $\frac{7n}{2}$, вычисляющая линейную функцию $\phi : \mathbb{B}^{2n} \to \mathbb{B}^n$ с матрицей $\begin{pmatrix} A^{-1} & A \end{pmatrix}$.
3. Существует схема размера $\frac{5n}{2} - 1$, вычисляющая линейную функцию $\phi : \mathbb{B}^{2n} \to \mathbb{B}^n$ с матрицей $\begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix}$.

Доказательство. 1. Сначала построим сумму $\bigoplus_{i=1}^{n/2} x_i$ (на это потребуется $\frac{n}{2} - 1$ гейтов). Затем, последовательно добавляя входы x_i, $i = \frac{n}{2}..n$, мы сможем вычислить выходы y_i, $i = \frac{n}{2}..n$, а в результате получится сумма всех входов $\bigoplus_{i=1}^{n} x_i$ (на это уйдёт ещё $\frac{n}{2}$ гейтов). Наконец, первые $\frac{n}{2}$ выходов получатся последовательным «вычитанием» первых $\frac{n}{2}$ входов из суммы всех входов (ещё $\frac{n}{2}$ гейтов).

2. Мы только что выяснили, что для того чтобы реализовать левую часть этой матрицы, нужны $\frac{3n}{2} - 1$ гейтов. Затем мы просто прибавим к каждому выходу по отдельности по два бита, которые нужно добавить из правой части (в последней строке — три

бита); это займёт ещё $2n + 1$ гейтов.

3. Заметим, что в данном случае

$$\phi(a, b) = \begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = A^{-1}(a \oplus b)$$

для любых $a, b \in \mathbb{B}^n$. Поэтому мы сначала сложим $a \oplus b$ (что займёт n гейтов), а затем реализуем A^{-1} (что займёт ещё $\frac{3n}{2} - 1$ гейтов).

□

3.5 Две конструкции

В этом разделе мы рассмотрим две различные конструкции кандидатов в функции с секретом. Ни одна из них сама по себе не даёт функцию с секретом, надёжную в слабом смысле; но в следующем разделе мы их совместим, и комбинация будет работать как надо.

В первой конструкции обращение с секретом быстрее, чем обращение без секрета, но прямое вычисление функции ещё сложнее. В терминах определения 3.1 мы представляем кандидата в функции с секретом с одинаковыми длинами начальных чисел генератора, публичной информации, секрета, входа и выхода:

$$c(n) = m(n) = pi(n) = ti(n) = n.$$

Генератор ключей производит пару из публичной информации и секрета (pi, ti), где ti — это само начальное число генератора, а $pi = A(ti)$ (таким образом, на генератор нужно $\frac{3n}{2} + O(1)$ гейтов). В этой конструкции вычисление функции производит код c для сообщения m следующим образом:

$$\mathrm{Eval}(pi, m) = A^{-1}(pi) \oplus A(m).$$

Вот матрица вычисления функции $\mathrm{Eval}(pi, m)$:

$$\begin{pmatrix} A^{-1} & A \end{pmatrix}.$$

Верхнюю оценку на её схемную сложность мы уже доказали в лемме 3.7; можно вычислить эту функцию схемой размера $\frac{7n}{2}$.

Обращение с секретом происходит следующим образом:

$$\mathrm{Inv}(\mathrm{ti}, c) = A^{-1}(A^{-1}(\mathrm{pi}) \oplus c) = A^{-1}(\mathrm{ti} \oplus c).$$

Благодаря линейности (отметим, что оценки из предыдущих параграфов здесь неприменимы, потому что в матрице обращения полно одинаковых столбцов) эту схему можно реализовать за $\frac{5n}{2} - 1$ гейтов: сначала за n гейтов можно вычислить $\mathrm{ti} \oplus c$, а затем применить A ещё за $\frac{3n}{2} - 1$ гейтов (см. лемму 3.7).

Наконец, противнику придётся обращать функцию более сложным способом, он ведь не знает ti:

$$m = A^{-1}(A^{-1}(\mathrm{pi}) \oplus c) = \mathcal{A}\begin{pmatrix}\mathrm{pi}\\ c\end{pmatrix}.$$

По лемме 3.6, схемная сложность этой функции не меньше $\frac{13n}{4} - 5$ гейтов, и каждый противник с менее чем $\frac{13n}{4} - 5$ гейтами в своём распоряжении не сможет её вычислить более чем на половине входов.

В этой конструкции вычисление функции оказывается сложнее, чем обращение без секрета. Чтобы это исправить, рассмотрим другую конструкцию, также семейство кандидатов в функции с секретом. Но теперь $c(n) = m(n) = n$ и $\mathrm{pi}(n) = \mathrm{ti}(n) = 0$, то есть ровным счётом никакой информации, ни секретной, ни публичной, тут нету. Эта конструкция — просто односторонняя функция Хильтгена. Формально функции определяются следующим образом:

$$\begin{aligned}\mathrm{Eval}(m) &= A(m),\\ \mathrm{Inv}(c) &= A^{-1}(c),\\ \mathrm{Adv}(c) &= A^{-1}(c).\end{aligned}$$

Конечно, этот кандидат в функции с секретом вовсе не надёжен даже в слабом смысле, ведь обращение реализуется вообще безо всякого секрета. Для сообщения m длины $|m| = n$ вычисляющая функцию

схема состоит из $n + 1$ гейтов, а обращение, по лемме 3.7, требует схем размером не меньше $\frac{3n}{2} - 1$ гейтов каждая. Поэтому во второй конструкции вычислять функцию легко, а обращать трудно, как для противника, так и для честного участника протокола.

3.6 Семейство функций с секретом, надёжных в слабом смысле

В предыдущем разделе мы построили два семейства кандидатов в функции с секретом. В одном из них вычислять функцию было легко, а обращать сложно, а в другом обращение с секретом было простым, а вычисление и обращение без секрета — сложными.

Теперь объединим эти две функции. В результате и обращение с секретом, и вычисление функции окажутся проще, чем обращение без секрета.

Разделим вход на две части: первую часть m_1 длины n подвергнем нашей первой (менее тривиальной) конструкции, а ко второй части m_2 длины αn применим вторую конструкцию. Мы выберем α позже, так, чтобы максимизировать относительную сложность для противника.

Теперь каждый участник описывается блочно-диагональной матрицей:

$$\begin{aligned} \mathrm{Eval}(pi, m) &= \begin{pmatrix} A^{-1} & A & 0 \\ 0 & 0 & A_* \end{pmatrix} \begin{pmatrix} pi \\ m_1 \\ m_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \\ \mathrm{Inv}(ti, c) &= \begin{pmatrix} A^{-1} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} ti \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}, \end{aligned}$$

$$\mathrm{Adv}(\mathrm{pi}, m) = \begin{pmatrix} A^{-2} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} \mathrm{pi} \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix},$$

где A_* обозначает матрицу с той же структурой, что и A, но для размерности αn вместо n. Тогда в терминах определения 3.1 мы получаем семейство кандидатов в функции с секретом, где входы и выходы функции длиннее начального числа генератора, а также публичной информации и секрета:

$$\begin{aligned} \mathrm{pi}(n) &= \mathrm{ti}(n) = n, \\ c(n) &= m(n) = (1+\alpha)n. \end{aligned}$$

Лемма 3.7 даёт верхние оценки сложности вычисления функции и обращения с секретом:

$$\begin{aligned} C(\mathrm{Eval}) &\leq \frac{7n}{2} + \alpha n + 1, \\ C(\mathrm{Inv}) &\leq \frac{5n}{2} + \frac{3\alpha n}{2} - 2. \end{aligned}$$

А лемма 3.6 даёт нижнюю оценку сложности обращения функции без секрета:

$$C(\mathrm{Adv}) \geq \frac{13n}{4} + \frac{3\alpha n}{2} - 7.$$

Таким образом, чтобы получить семейство надёжных в слабом смысле функций с секретом, нам достаточно выбрать α так, чтобы

$$\begin{aligned} \frac{13}{4} + \frac{3\alpha}{2} &> \frac{7}{2} + \alpha, \\ \frac{13}{4} + \frac{3\alpha}{2} &> \frac{5}{2} + \frac{3\alpha}{2}. \end{aligned}$$

Второе неравенство тривиально, а из первого получается, что $\alpha > \frac{1}{2}$.

Мы бы хотели максимизировать порядок надёжности в слабом смысле (см. определение 3.3); поскольку генерация секрета в этой конструкции всегда строго быстрее вычисления и обращения с секретом,

мы на самом деле максимизируем

$$\min\left\{\lim_{n\to\infty}\frac{C(\mathrm{Adv}_n)}{C(\mathrm{Inv}_n)}, \lim_{n\to\infty}\frac{C(\mathrm{Adv}_n)}{C(\mathrm{Eval}_n)}\right\} = \min\left\{\frac{\frac{13}{4}+\frac{3\alpha}{2}}{\frac{5}{2}+\frac{3\alpha}{2}}, \frac{\frac{13}{4}+\frac{3\alpha}{2}}{\frac{7}{2}+\alpha}\right\}.$$

Это выражение достигает максимума при $\alpha = 2$, и порядок надёжности при этом достигает $\frac{25}{22}$. В итоге мы доказали следующую теорему.

Теорема 3.1 Существует семейство функций с секретом, надёжных в слабом смысле, с длиной начального числа генератора $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, длинами входа и выхода функций $c(n) = m(n) = 3n$ и порядком надёжности в слабом смысле $\frac{25}{22}$.

Пример 3.2. Приведём минимальный пример, для которого эта конструкция даёт нетривиальную гарантию надёжности. Как мы уже упоминали, все конструкции здесь основаны на линейной функции $f : \mathbb{B}^n \to \mathbb{B}^n$, разработанной А. Хильтгеном [69]. Наименьшее n, для которого f действительно труднее обратить, чем вычислить, равно 7, но поскольку нам нужно, чтобы n делилось на 4, мы в этом примере будем рассматривать $n = 8$. В этом случае $C(f) = 9$, $C(f^{-1}) = 11$, $C(f^{-2}) = 11$. Лемма 3.7 позволяет оценить сложности порождения ключей, вычисления функции и её обращения с секретом:

$$\begin{aligned} C(\mathrm{Key}_8) &= 11, \\ C(\mathrm{Eval}_8) &= 29, \\ C(\mathrm{Inv}_8) &= 30. \end{aligned}$$

С другой стороны, лемма 3.6 говорит нам, что

$$C(\mathrm{Adv}_8) \geq 31.$$

Таким образом, мы получили функцию с секретом, для которой обращение без знания секретного ключа требует по меньшей мере 31 гейт, в то время как вычисление самой функции и её обращение с секретом можно реализовать за 29 и 30 гейтов соответственно.

3.7 Функция с секретом с экспоненциальной гарантией надёжности

Мы построили конструкцию семейства линейных надёжных в слабом смысле функций с секретом, которые гарантировали, что любая схема с числом гейтов менее требуемого не сможет обратить эти функции на более чем половине их входов.

Сейчас мы используем эту конструкцию, чтобы создать систему, у которой будут уже суперполиномиальные гарантии надёжности (а именно $2^{-c\sqrt{n}+o(\sqrt{n})}$). Обозначим через h функцию, которую противник должен был вычислять в предыдущем разделе, а её матрицу — через X.

Рассмотрим теперь линейную функцию H, определённую следующей блочно-диагональной матрицей:

$$H\left(\vec{x}^{(1)},\vec{x}^{(2)},\ldots,\vec{x}^{(m)}\right)=\begin{pmatrix} X & 0 & \ldots & 0\\ 0 & X & \ldots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \ldots & X\end{pmatrix}\begin{pmatrix}\vec{x}^{(1)}\\ \vec{x}^{(2)}\\ \vdots\\ \vec{x}^{(m)}\end{pmatrix}.$$

По лемме 3.5, схемная сложность функции H составляет по меньшей мере $mC(h)$. Размерности матрицы X составляют $(1+\alpha)n\times(2+\alpha)n$; введём для удобства обозначение $n'=(1+\alpha)n$.

Лемма 3.8 Зафиксируем многочлен p. Если схема вычисляет функцию H на более чем $\frac{1}{p(m)}$ доле её входов, и для каждого блока H:

— все столбцы X различны;

— каждая строка X содержит не менее u ненулевых элементов;

— после удаления из X любых t столбцов оставшаяся матрица всё ещё содержит по крайней мере одну строку с не менее чем двумя ненулевыми элементами;

то сложность этой схемы составляет не менее $(u+t)(m-\log p(m))$.

Доказательство. Прежде всего вспомним, что H состоит из m отдельных блоков с непересекающимися множествами переменных X_i; обозначим $h_i = H|_{X_i}$. Поскольку X_i не пересекаются, ошибки в вычислении функций h_i независимы: если схема C вычисляет функцию h_i на доле β_i её входов, а функцию h_j — на доле β_j её входов, она не может вычислить H более чем на доле $\beta_i\beta_j$ её входов. Следовательно, в матрице H есть не более $\log p(m)$ блоков, где схема C может себе позволить ошибиться на половине входов. На протяжении этого доказательства мы будем называть их «ужасными» блоками.

После этого замечания мы снова начинаем то же самое доказательство методом исключения гейтов, к которому мы уже несколько раз прибегали. Рассмотрим верхний гейт в некотором топологическом порядке и две переменных, которые в него входят. В доказательстве леммы 3.5 мы маркировали переменные как «хорошие» и «плохие» в зависимости от того, входят ли они в блок, в котором все «хорошие» переменные уже закончились. На этот раз мы делаем то же самое, но заранее маркируем все переменные в «ужасных» блоках как «плохие».

Как и в предыдущих доказательствах, если в верхний гейт входит хотя бы одна «плохая» переменная, мы присвоим значение именно ей и удалим на этом шаге из схемы всего один гейт. А когда в верхний гейт входят две «хороших» переменных, мы всегда имеем возможность удалить как минимум два гейта из схемы. Давайте немного подробнее рассмотрим эту ситуацию.

Перед нами могут возникнуть всё те же три основных случая, которые мы рассматривали в доказательстве леммы 3.4. Однако на этот раз мы не можем подставить в точности то значение, которое хотим подставить. На этот раз нам придётся всё время оставаться в той половине входов, на которой схема ошибается на менее чем половине оставшихся входов (поскольку исходная схема ошибается не более чем

на половине входов, такая подсхема должна существовать). Однако мы можем объединить третий и первый случаи без потери общности: формула может зависеть исключительно от $x \wedge y$ не в большей степени, чем от $x \oplus y$; оба случая одинаковыми рассуждениями приводят к вероятности ошибки не менее $\frac{1}{2}$.

Остаток доказательства полностью повторяет лемму 3.4. Мы ведём доказательство по индукции, исключая по два гейта, если в верхний гейт входят две «хороших» переменных, и по одному, если есть среди них «плохая». Таким образом, общая сложность не меньше, чем количество «хороших» переменных, умноженное на два, плюс количество оставшихся «плохих» переменных. «Ужасные» блоки приходится полностью выбрасывать: в конце концов, их сложность и на самом деле может быть равна нулю. Таким образом, мы получаем нижнюю оценку схемной сложности $(t + u)(m - \log p(m))$. □

Заметим также, что копирование исходных матриц в большую блочно-диагональную конструкцию не меняет параметры (в том числе порядок обратимости) семейства функций с секретом, надёжных в слабом смысле. Таким образом, мы получаем следующую теорему.

Теорема 3.2 Существует семейство надёжных в слабом смысле функций с секретом $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}_{n=1}^{\infty}$ с длиной начального числа генератора $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, длинами входа и выхода функций $c(n) = m(n) = 3n$, схемными сложностями

$$C(\mathrm{Inv}_n) \leq \frac{11n}{2} + O(1), \quad C(\mathrm{Eval}_n) \leq \frac{11n}{2} + O(1), \quad C(\mathrm{Key}_n) = n + 1$$

и порядком надёжности в слабом смысле $\frac{25}{22}$.

Более того, для любой константы $\delta > 0$ ни один противник, располагающий менее чем $\frac{25}{4}n - \frac{5}{2}\delta\sqrt{n}$ гейтами, не способен обратить эти функции с секретом на более чем $2^{-\delta\sqrt{n}+o(\sqrt{n})}$ доле их входов.

Глава 4

Новые алгебраические конструкции криптографических примитивов

4.1 Алгебраическая криптография

Криптография с открытым ключом, начиная с самого своего появления [39, 107], активно пользуется алгебраическими конструкциями. Например, как уже говорилось в Главе 1, система RSA основана на методах теории чисел; уже сама её конструкция содержит вычисление функции Эйлера $\varphi(n)$.

Однако обычно под *алгебраической криптографией* с открытым ключом принято понимать несколько более конкретное понятие. Хорошо изученные алгебраические структуры могут предоставить значительно бóльшие возможности для анализа. Поэтому алгебраическая криптография рассматривает конструкции, в которых отображения кодирования и декодирования являются гомоморфизмами групп. В [148] даётся следующее определение *гомоморфной криптосистемы.*

Определение 4.1 Пусть H — конечная группа, $H \neq \{e\}$, G — конечно порождённая группа, $f : G \to H$ — эпиформизм. Пусть даны множество R представителей классов смежности G по подгруппе $\mathrm{Ker}(f)$ и множество A слов в некотором алфавите вместе с таким отображением $P : A \to G$, что $\mathrm{Im}(P) = \mathrm{Ker}(f)$. Набор (R, A, P) называется *гомоморфной криптосистемой* над H относительно f, если:

— имеются вероятностные алгоритмы для случайного порождения элементов множеств A, G, H, для вычисления обратного элемента и произведения в группах G и H, вероятностная сложность которых полиномиальна от размера задания групп G, H и мно-

жества A;

— для всякого $g \in R$ его образ $f(g)$ можно вычислить за полиномиальное от размера задания участвующих групп время, также как и для всякого элемента $h \in H$ можно вычислить его единственный прообраз $g \in R$: $f(g) = h$.

— P является секретно–обратимым отображением (trapdoor).

Если перейти на более привычный криптографический язык, то в этом определении *открытым ключом* являются G, H, R, A, P и $f|_R$, а *секретным ключом* является алгоритм для вычисления P^{-1} (та секретная информация, которая требуется для обращения секретно–обратимого отображения P). При *кодировании* каждой букве исходного сообщения $h \in H$ сопоставляется (единственный) элемент $r \in R$, для которого $f(r) = h$, и случайный элемент $a \in A$; в качестве кода выдаётся $P(a)r \in G$. А при *декодировании* нужно для данного $g \in G$ найти тот (единственный) $r \in R$ и $a \in A$, для которых $rg^{-1} = P(a)$; тогда, чтобы вычислить исходное сообщение, достаточно будет применить f, ведь $f(g) = f(r)$.

Формально даже в этом, более узком, смысле алгебраическая криптография с открытым ключом всё равно появилась практически одновременно с криптографией с открытым ключом; криптосистема, основанная на квадратичных вычетах, была первой гомоморфной криптосистемой и одной из первых криптосистем вообще [53, 54].

Пример 4.1. Рассмотрим $n = pq$, где p и q — простые числа размера $O(\log n)$. Положим

$$G = \{g \in \mathbb{Z}_n^* \mid J_n(g) = 1\},$$

где J_n — символ Якоби по модулю n, и $H = \mathbb{Z}_2^+$. Тогда для данного $g_0 \in G$, не являющегося квадратом, тройка (R, A, P), где

$$R = \{1, g_0\}, \quad A = \mathbb{Z}_n^+, \quad P(g) : g \mapsto g^2,$$

представляет собой гомоморфную криптосистему над H относительно гомоморфизма $f : G \to H$ с ядром $\mathrm{Ker}(f) = \{g^2 \mid g \in \mathbb{Z}_n^*\}$.

Однако собственно алгебраическая криптография в её современном понимании ведёт начало от работ, связанных с криптографическими примитивами, построенными на эллиптических кривых [80, 91]. Основные конструкции этих примитивов на самом деле мало отличаются от конструкций, основанных на дискретном логарифме, таких, как протокол согласования ключа Диффи–Хеллмана. Основное отличие в том, что вычисления ведутся в абелевой группе точек эллиптической кривой вида $y^2 = x^3 + ax + b$ или $y^2 + xy = x^3 + ax^2 + b$ над конечным полем; это позволяет существенно уменьшить размеры ключей и сделать криптосистемы более эффективными [1, 22, 65, 136]. В последнее время в США был принят ряд криптографических стандартов, основанных на эллиптических кривых.

Отметим одну важную особенность. И классические конструкции, и криптография на эллиптических кривых имеют дело с *абелевыми* группами. Эта дополнительная структура позволяет сделать алгоритмы кодирования и декодирования ещё более эффективными, а также существенно упрощает анализ возникающих задач. Однако за последние десять лет, начиная с основополагающих работ Питера Шора [19, 116], абелевы конструкции оказались подвергнуты новому типу атак, со стороны *квантовых вычислений*. Задачи разложения чисел на множители и дискретного логарифма, на которых, с некоторыми вариациями, были основаны все наиболее популярные криптосистемы, включая криптосистемы, основанные на эллиптических кривых, получили простые и эффективные алгоритмы решения на квантовом компьютере (см. также [86, 94] и [12, Глава 20]). Это происходит вследствие того, что на квантовом компьютере при помощи квантовых преобразований Фурье можно эффективно решить задачу

вычисления порядка элемента в абелевой группе, частным случаем которой, собственно, и является дискретный логарифм. Решение этой задачи настолько помогает для взлома криптографических примитивов, основанных на абелевых группах конструкций, что в ситуации, когда квантовые компьютеры уже присутствуют на горизонте (хотя их практические перспективы пока не до конца ясны), ограничиваться коммутативными конструкциями уже недостаточно: хотелось бы получить и *неабелевы* конструкции, от которых можно ожидать большей устойчивости [10].

Именно поэтому основанные на эллиптических кривых конструкции и классические криптографические примитивы Диффи-Хеллмана и RSA были обобщены, и задача построения гомоморфных криптосистем была поставлена в [45, 139]. Первые шаги в использовании алгебраических конструкций, в частности теории групп, для построения криптографических примитивов были сделаны в работах [15, 93, 95, 102]. Важный шаг на пути разработки неабелевых криптографических примитивов составили работы И. Аншель, М. Аншеля и Д. Голдфельда, одну из конструкций которых мы будем подробно рассматривать в разделе 4.9 [7–9].

Дальнейшая разработка этих идей продолжалась в работах Д. Ю. Григорьева и И. Н. Пономаренко. Был разработан аппарат определений, таких, как Определение 4.1. Были построены конструкции неабелевых гомоморфных криптосистем с открытым ключом [148], конструкции гомоморфных криптосистем над кольцами [57], была разработана общая схема конструкции более сложных гомоморфных криптосистем из блоков [59] (эта схема будет играть важную роль в настоящей работе), исследована связь между гомоморфными криптосистемами и кодированием булевских схем [58].

В работе [146] Д. Ю. Григорьев предложил использовать для

криптографических целей теорию инвариантов групп. В разделе 4.5 мы обсудим эти конструкции более подробно, а сейчас перейдём к обсуждению взлома с доказательством — второй основополагающей идеи этой главы. Глава основана на результатах, опубликованных в [56, 147].

4.2 Ослабленные результаты современной криптографии

Как мы уже упоминали, современная криптография практически не позволяет предоставить строгие доказательства надёжности тех или иных примитивов. Дело здесь в том, что классическое понятие криптографической надёжности естественным образом связано со сложностью в среднем, а не классической сложностью в худшем случае.

Например, в классическом понятии *семантической надёжности* криптосистемы [53] потенциальный взломщик, зная распределение на множестве M исходных сообщений, получает на вход закодированное сообщение и публичный ключ, но не может его декодировать значительно чаще, чем алгоритм, который не знает ничего, кроме M (M нужно потому, что, например, если известно, что всё время кодируется одно и то же сообщение, противник действительно сможет его раскодировать, и ему для этого не обязательно будет смотреть на код и публичный ключ). Основная проблема здесь в том, что вероятности в этом определении берутся в том числе и по распределению на входах криптосистемы, на сообщениях; это, конечно, по существу, потому что на практике обычно достаточно взламывать криптосистему не на всех сообщениях, а на существенной их доле.

Недавние результаты позволили связать криптографическое понятие надёжности с предположениями о сложности определённых задач в худшем случае [4, 42, 104, 105]. Однако пока что эти предположения выглядят достаточно искусственными и, как и раньше, не транс-

лируются в предположения о (не)равенстве тех или иных сложностных классов. Вообще говоря, представляется, что современной криптографии ещё очень далеко до *доказуемо* надёжных конструкций. Есть даже результаты, говорящие о том, что это может оказаться невозможным или приведёт к маловероятным последствиям для сложностных классов [25].

Поэтому вполне естественно, что исследователи, столкнувшись со слишком трудной проблемой, начали работу над альтернативными определениями, критериями и контекстами, пытаясь доказать надёжность хоть чего-нибудь, хоть в каких-нибудь определениях.

Первый естественный подход — рассмотреть не криптосистемы с открытым ключом, а более слабые криптографические примитивы, такие, например, как *односторонние функции* (отметим, что односторонняя функция ещё не означает криптосистемы с открытым ключом, для неё нужна функция с секретом, trapdoor function). И действительно, результаты об односторонних функциях оказывается проще доказать. Например, полные односторонние функции (*полной* мы называем такую функцию f, что если односторонние функции вообще существуют, то f тоже односторонняя) известны уже достаточно давно [50, 85], а полные криптосистемы с открытым ключом появились только в последние годы в работах Д. Харника и Д. Ю. Григорьева с соавторами [55, 66]. Более того, недавно Л. А. Левин разработал более или менее естественные конструкции комбинаторных полных односторонних функций [152]. Нам удалось получить новые результаты в этом направлении [56, 147], и подробнее это обсуждается в главе 2.

Другой подход, тоже приводящий к более слабым результатам, заключается в том, чтобы изменить само понятие надёжности. Если рассмотреть более слабые определения надёжности, оказывается возможным построить даже доказуемо надёжные примитивы. Как при-

мер такого подхода можно привести теорию *слабо односторонних* (feebly one-way) функций, развитую А. Хилтгеном [69, 70]; этот сюжет и наши результаты в этом направлении были изложены в главе 3.

С другой стороны, были получены частичные результаты в направлении, где надёжность определена для *противника в худшем случае*. Этот противник должен взламывать криптосистему *всегда*, для всех сообщений. Поэтому неудивительно, что в таком контексте можно получить результаты, связывающие надёжность со сложностными предположениями [44, 83]. Желающим найти детальный обзор этого и других результатов мы рекомендуем [58, 59, 92].

В следующем разделе будет введён другой подход, ослабляющий понятие надёжности в другом смысле. Оказывается, что наше понятие *доказуемого взлома*, хотя и довольно искусственное, позволяет связать доказуемый взлом криптосистем, основанных на теории инвариантов, и протоколов согласования ключа Аншель-Аншеля-Голдфелда с предположениями о классах сложности в худшем случае.

4.3 Доказуемый взлом

Рассмотрим систему с тремя участниками: Алисой, Бобом и Чарли. Пусть, как это обычно предполагается в криптографии, Алиса (A) и Боб (B) пытаются общаться друг с другом по некоторому криптографическому протоколу (в протоколе согласования ключа Алиса и Боб — его равноправные участники, в криптосистеме Алиса формирует пару из публичного и секретного ключа и выдаёт публичный ключ, а Боб кодирует сообщение и пересылает его Алисе по открытому каналу), а Чарли (C) пытается расшифровать их беседу, перехватывая сообщения, которые Боб посылает Алисе. Однако на этот раз задача Чарли выглядит по-другому: ему нужно не просто расшифровать сообщение Боба, но и каким-нибудь образом доказать, что сообщение

у Боба было именно такое. Например, Чарли может сам не особенно доверять своей расшифровке, или у него есть начальник, которому нужен не просто ответ, а ответ с доказательством. Именно это мы понимаем под задачей *доказуемого взлома*.

В контексте доказуемого взлома Чарли должен получить не только сообщение m из кода c, но ещё и какое-нибудь доказательство того, что m можно закодировать в виде c. Что может служить таким доказательством? Мы примем за определение следующую естественную мысль: Чарли должен предъявить не только сообщение m, но и такую строку случайных битов r, для которой алгоритм E, получая на вход r и m, выдаёт тот же код c.

Замечание 4.5. В последующем тексте мы (эквивалентно) переопределим алгоритм кодирования E как детерминированный полиномиальный в худшем случае алгоритм с доступом к случайной строке r. Он получает в качестве входа публичный ключ e, исходное сообщение m и случайную строку r, а на выход подаёт закодированное сообщение $E(r, e, m) = c$.

Конечно, искомая строка случайных битов не обязана быть единственной: может быть несколько строк $\{r_1, \ldots, r_k\}$, из которых получается один и тот же шифр, т.е. для публичного ключа pk и сообщения m

$$E(m, pk, r_1) = \ldots = E(m, pk, r_k).$$

В этом случае, разумеется, противнику достаточно предъявить *какую-нибудь* строку случайных битов, приводящую к искомому шифру; при $k > 1$ у Чарли всё равно нет никаких шансов отличить одну от другой.

Первое неформальное обсуждение доказуемого взлома началось в связи с криптосистемой Рабина–Голдвассер–Микали, основанной на квадратичных вычетах [53]. Было показано, что доказуемый взлом этой криптосистемы означал бы, что задача разложения чисел на мно-

жители лежит в RP, т.е. в классе задач, которые решаются полиномиальным вероятностным алгоритмом с односторонней ошибкой (подробнее об этом и других сложностных классах можно прочесть в [12, 50] и многих других источниках). Однако мы не знаем ни одной работы, где это (или хотя бы определение доказуемого взлома) было бы формально описано и изучено.

Как мы уже упоминали в разделе 4.2, одним из главных нерешённых вопросов современной теоретической криптографии является возможность построения надёжных примитивов, основанных на каких-нибудь естественных сложностных предположениях, таких, как $P \neq NP$. В этой главе мы представляем два немного отличающихся определения доказуемого взлома и доказываем, что два различных криптографических протокола, а именно протокол согласования ключа Аншель-Аншеля-Голдфельда и криптосистемы, основанные на инвариантах групп, являются устойчивыми относительно доказуемого взлома в худшем случае, если $NP \not\subseteq RP$. Кроме того, мы разрабатываем способы сделать криптосистемы, основанные на инвариантах, устойчивыми и в обычном криптографическом смысле.

4.4 Определения

Сначала мы определим доказуемый взлом криптосистем с открытым ключом, а затем расширим данное определение на протоколы согласования ключа. Мы представим два различных определения, одно стандартное, в среднем, другое в худшем случае.

Определение 4.2 Будем говорить, что противник C осуществляет *доказуемый взлом* криптосистемы (G, E, D), если существует такой многочлен p, что для равномерного распределения по множеству сообщений m и множеству случайных битов всех участвующих в конструкции алгоритмов (публичный ключ pk берётся из пары (pk, sk),

сгенерированной алгоритмом порождения ключа $G(1^n)$)

$$\Pr\left[C(E(m, pk, r), pk) = (m, r')\right] \geq \frac{1}{p(n)},$$

где $E(m, pk, r') = E(m, pk, r)$, а n — параметр надёжности.

Параметр надёжности в определении 4.2 играет роль длины ключа; параметр надёжности нужно выбирать так, чтобы время работы алгоритмов взлома, подсчитанное с этим параметром, было достаточно большим, чтобы взлом за это время был лишён смысла.

Если E — детерминированный алгоритм, то доказуемый взлом эквивалентен обычному взлому (множество случайных битов пусто). Противник может подтвердить, что он правильно угадал сообщение, попросту закодировав его ещё раз. Это и есть в точности идея доказуемого взлома. Противник должен не просто угадать сообщение, но ещё и доказать, что это действительно корректный код этого сообщения. В некоторых из наших примеров первая задача будет тривиальной, что не помешает второй задаче иметь большую сложность.

Мы также введём ещё одно, совсем сильное определение — противника, который взламывает криптосистему в худшем случае. Разница с обычным определением заключается в том, что противник должен добиваться успеха на *всех* входах.

Определение 4.3 Противник C осуществляет *доказуемый взлом в худшем случае* криптосистемы (G, E, D), если существует такой многочлен p, что для всех сообщений m, всех пар ключей (pk, sk), сгенерированных алгоритмом порождения ключа $G(1^n)$, и всех случайных битов алгоритма кодирования E

$$\Pr\left[C(E(m, pk, r), pk) = (m, r')\right] \geq \frac{1}{p(n)},$$

где $E(m, pk, r') = E(m, pk, r)$, n — параметр сложности, а распределение берётся по множеству случайных битов взламывающего алгоритма C.

Будем говорить, что криптосистема (G, E, D) *надёжна против доказуемого взлома* (*в худшем случае*), если не существует полиномиальной вероятностной машины Тьюринга C, которая бы осуществляла доказуемый взлом (в худшем случае) криптосистемы (G, E, D).

Замечание 4.6. Легко придумать тривиальную криптосистему, которая надёжна против доказуемого взлома (надёжна в смысле определения 4.2, что автоматически делает её надёжной и против взлома по определению 4.3). Пусть Боб передаёт своё сообщение открыто (раскодирование тогда становится тривиальным), но при этом добавляет в конец сообщения значение некоторой односторонней функции от своей случайной строки. Алиса может не обращать внимания на эту функцию, но Чарли придётся её обратить, чтобы получить подходящий набор случайных битов Боба. Правда, такая конструкция требует по крайней мере существования односторонних функций, а оно пока не доказано. И тем не менее, цель нашей работы из задачи построения криптосистем, надёжных против доказуемого взлома, превращается в задачу построения таких криптосистем, которые к тому же ещё и могут быть сделаны надёжными в обычном криптографическом смысле. Конечно, мы не сможем *доказать* их «традиционную» надёжность, но мы представим конструкции, которые являются, на наш взгляд, достаточно надёжными криптосистемами.

4.5 Криптосистемы, основанные на инвариантах групп, и их доказуемый взлом

4.5.1 Криптосистемы, основанные на инвариантах групп

В работе [146] Д. Ю. Григорьев предложил новый класс криптосистем — криптосистемы, основанные на инвариантах групп. В такой криптосистеме Алиса выбирает группу $G \leq GL(n, F)$, действующую на линейном пространстве F^n для некоторого поля F. В качестве сек-

ретного ключа Алиса выбирает некоторое множество X и такой инвариант $f : F^n \to X$, что $\forall g \in G\ f(g\mathbf{x}) = f(\mathbf{x})$. Она также выбирает заданное базисом подпространство сообщений $M \subseteq F^n$; выбирает она его так, чтобы разные сообщения $m \in M$ представляли разные орбиты группы G на множестве M: $\forall\ m_1 \neq m_2 \in M\ f(m_1) \neq f(m_2)$. Таким образом, криптосистема, основанная на инвариантах, определяется набором (G, f, M). В качестве публичного ключа Алиса передаёт набор образующих группы G (его размер должен быть полиномиален от n) и базис пространства M.

В качестве M мы будем, как правило, рассматривать множество из двух элементов; в любом случае, дальнейшие конструкции подразумевают, что Алиса способна найти m по $f(m)$, то есть либо она умеет обращать f, либо M настолько мало, что можно перебрать все его элементы. Как бы то ни было, уметь кодировать один бит достаточно для того, чтобы закодировать сообщение любой длины.

Боб выбирает вектор $m \in M$ (где m — сообщение, которое нужно закодировать) и случайный элемент $g \in G$. Затем Боб передаёт Алисе gm. Алиса может расшифровать сообщение, вычислив известный ей инвариант $f(gm) = f(m)$. Поскольку Алиса выбирала пространство сообщений M трансверсально орбитам, значение инварианта $f(m)$ однозначно позволит ей найти исходное сообщение m. Будем называть набор (G, f, M) *допустимым*, если он корректно определяет криптосистему, основанную на инвариантах, т.е. для G, f и M выполняются приведённые в первом абзаце условия.

Становится очевидным, что главная сложность в конструировании криптосистем, основанных на инвариантах, заключается в том, чтобы найти хорошо скрытый инвариант. В дальнейшем мы покажем несколько способов выбрать такой инвариант. Эти способы аналогичны предложенным в [59] и могут быть в общем описаны следующей

конструкцией. Рассмотрим дерево, каждая вершина которого содержит набор (G, f, M), причём общий предок нескольких узлов получается из них при помощи той или иной теоретико-групповой конструкции (см. раздел 4.6). Алиса строит это дерево от листьев к корню, на каждом шаге запоминая G, f и M. Ниже мы опишем рекурсивные правила, по которым по потомкам узла в нём строятся G, f и M. После создания дерева Алиса берёт криптосистему (G, f, M), получившуюся в корне дерева, и использует её для кодирования вышеописанным способом.

Таким образом, противник сможет взломать криптосистему, если он будет знать структуру этого дерева (мы предполагаем, что криптосистемы в листьях дерева взломать нетрудно, иначе нет смысла его строить). С точки зрения надёжности эта структура эквивалентна полному описанию инварианта (по ней инвариант можно восстановить за полиномиальное время), так что её тоже можно рассматривать как секретный ключ Алисы. Надёжность полученной криптосистемы будет основываться на сложности задач сопряжения и принадлежности к подгруппе, как и надёжность конструкций в работе [59]. В разделе 4.6 мы подробно изложим эту конструкцию.

4.5.2 Криптосистема на инвариантах, доказуемый взлом которой влечёт NP⊆RP

Наша конструкция будет основана на модулярной группе. *Модулярная группа* — это группа $SL_2(\mathbb{Z})$ матриц 2×2 с определителем 1 (*унимодулярных матриц*). Групповые свойства этой конструкции подробно изложены, например, в [11].

В [23, Следствие 11.5] Бласс и Гуревич доказали, что для модулярной группы *ограниченная задача принадлежности* (bounded membership problem, BM) является NP-полной. Задача формулирует-

ся следующим образом.

Задача 4.1 Пусть X — унимодулярная матрица, S — конечное множество унимодулярных матриц, а N — положительное целое число. Можно ли представить X в виде $\prod_{i=1}^{m} Y_i$, где $m \leq N$, и для каждого i либо Y_i, либо Y_i^{-1} лежит в S?

Замечание 4.7. Не стоит путать эту задачу с другими задачами из работы [23], о которых там доказывается, что они DistNP-полны (полны относительно сложности в среднем). Основное отличие в данном случае заключается в том, что здесь речь идёт о задаче принадлежности *группе*, а DistNP-полные задачи получаются из проверки принадлежности различным *полугруппам*.

Рассмотрим в качестве G унимодулярную группу

$$G = \left\{ \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, x \in \mathbb{Z} \right\}.$$

В качестве инварианта f мы рассмотрим тривиальный инвариант

$$f \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_2,$$

а в качестве пространства сообщений M — множество векторов

$$M = \left\{ \begin{pmatrix} 1 \\ x \end{pmatrix}, x \in \mathbb{Z} \right\}.$$

Боб выбирает элемент $g \in G$, получая его случайным образом в результате произведения не более чем N образующих. Затем он применяет g к вектору–сообщению m и передаёт полученный вектор gm и N. Алиса вычисляет $f(gm)$ и решает, каким был вектор m.

Отметим, что взломать эту «криптосистему» можно совершенно тривиально: кодирование не меняет ту часть вектора, которая отвечает за само сообщение. Однако мы сейчас увидим, что её доказуемый взлом NP-труден.

Теорема 4.1 Если существует противник C, который осуществляет доказуемый взлом в худшем случае описанной выше криптосистемы (G, f, M) за время, полиномиальное от суммы длин сообщения и публичного ключа, то $NP \subseteq RP$.

Доказательство. Идея доказательства состоит в том, что доказуемый взлом NP-труден, потому что к решению задачи ограниченной принадлежности в подгруппе модулярной группы легко свести известную NP-трудную задачу, что было показано в [23].

Во-первых, отметим, что

$$\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \lambda + \mu \\ 0 & 1 \end{pmatrix}.$$

Таким образом, задача ограниченной принадлежности в подгруппе модулярной группы эквивалентна задаче, можно ли заданное число представить в виде ограниченной суммы других заданных чисел. Это задача Суммы Целых Чисел (Integer Sum Problem), NP-полнота которой доказана в [23].

Если полиномиальный алгоритм решает задачу поиска с вероятностью успеха $\frac{1}{n^{Const}}$, эту вероятность легко увеличить до $\frac{3}{4}$, повторяя алгоритм полиномиальное число раз и выдавая ответ простым большинством. Поэтому, если полиномиально ограниченный противник доказуемо взламывает в худшем случае представленную криптосистему, то $NP \subseteq RP$. □

В разделах 4.6 и 4.7 мы представим конструкции, которые позволяют сделать криптосистемы, основанные на инвариантах групп, более разумными с точки зрения обычной криптографической надёжности.

4.6 Дерево групп

4.6.1 Общие замечания

Основанный на инварианте модулярной группы протокол, который мы описали в предыдущем разделе, на практике взломать ничуть не сложнее, чем криптосистему, которую мы описали в замечании 4.2. Обе конструкции, к сожалению, взламываются тривиально. В этой части нашей работы мы представим конструкцию, которая позволит нам «спрятать» эти примитивы в большом дереве групп. Мы также можем использовать эту конструкцию для того, чтобы улучшить надёжность протокола согласования ключа Аншель-Аншеля-Голдфельда.

Мы следуем идеям работы [59], чтобы получить дерево троек «группа–инвариант–сообщения». Если знать структуру дерева, можно будет эффективно вычислить инвариант в его корне, а если не знать, то инвариант окажется «спрятан» за достаточно сложными задачами.

В оставшейся части этого раздела мы сконцентрируемся на криптосистемах, основанных на инвариантах групп (введённых в разделе 4.5), поскольку к протоколам согласования ключа идеи статьи [59] можно применить прямо и дословно, а вот для криптосистем, основанных на инвариантах, мы разработаем несколько новых конструкций. Далее мы рассмотрим те же операции, что и в [59], и проследим судьбу инвариантов, а также предложим несколько новых операций, относящихся сугубо к инвариантам групп. Но сначала нужно ввести базовые определения.

Каждой вершине v дерева мы сопоставляем набор (G_v, f_v, M_v). Эти наборы получаются рекурсивно по построению дерева: нужно применять к каждой вершине, начиная от листьев и заканчивая корнем, одну из описанных ниже операций. Для каждой вершины v группа G_v является матричной группой $G_v \leq GL(n, R)$ для некоторого n и

некоторого кольца R.

Всему дереву мы будем сопоставлять полученный в его корне набор (G, f, M), где $G \leq GL(n, R)$ — группа, f — инвариант, т.е. такая функция $f : R^n \to R$, что $\forall g \in G\ \forall x \in R^n\ f(gx) = f(x)$, и $M \subset F^n$ — канонический набор *сообщений*, взятых из прообразов разных значений инварианта f: $\forall m \neq m' \in M\ f(m) \neq f(m')$.

Публичный ключ состоит из заданного линейным базисом кольца R, n, группы G, заданной матричными образующими, и M. Цель построения дерева в том, чтобы спрятать секретный ключ — стоящий в корне инвариант.

Замечание 4.8. Заметим, что в ситуациях, когда мы меняем инвариант, мы можем либо изменить инвариант с f на $f \circ h$, либо сменить пространство сообщений с M на $h(M)$. Поскольку нам нужно спрятать именно инвариант, а пространство сообщений мы будем выдавать публично, мы всегда будем выбирать первый из этих способов.

Наша цель — совместить полученную таким способом обычную криптографическую надёжность с надёжностью против доказуемого взлома в худшем случае, который мы доказали в теореме 4.1. Чтобы это сделать, мы разместим доказуемо надёжную конструкцию, основанную на модулярной группе, в один из листьев дерева. Тогда Чарли, чтобы решить задачу принадлежности в корне дерева, придётся решить задачу принадлежности для каждого листа (наша конструкция будет обладать этим свойством).

4.6.2 База рекурсии

Введём несколько формальных обозначений. Рассмотрим класс групп $\mathcal{G}$, замкнутый относительно некоторого множества теоретико-групповых операций $\mathcal{O}$ (список операций будет приведён ниже), которые преобразуют допустимые наборы в допустимые наборы. Для множества

$\mathcal{G}_0 \subset \mathcal{G}$ (которое является базой конструкции) мы рекурсивно определим класс $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ четвёрок (G, f, M, T) следующим образом.

— *База рекурсии*: любая четвёрка (G, f, M, T), где $G \in \mathcal{G}_0$, (G, f, M) является допустимым набором, а T — дерево из одной вершины, маркированной набором (G, f, M).

— *Шаг рекурсии*: для всех наборов четвёрок $\{(G_i, f_i, M_i, T_i)\}_{i=1}^{s}$ и любой s–арной операции $o \in \mathcal{O}$ класс $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ содержит четвёрку (G, f, M, T), где $G = o(G_1, \ldots, G_s)$, $f = o(f_1, \ldots, f_s)$, $M = o(M_1, \ldots, M_s)$, а T — дерево, получающееся из деревьев $T_1, \ldots, T_s$ добавлением нового корня с меткой o, к которому корни поддеревьев $T_1, \ldots, T_s$ добавляются как потомки.

4.6.3 Шаг рекурсии

В этом подразделе мы рассматриваем конкретные операции, которые можно применять к допустимым наборам. Первый их набор был рассмотрен в [58]; нам только нужно проверить, что происходит с инвариантами в каждой из этих конструкций.

1. *Изменение базового кольца* $\phi : R \to R'$. Если кольцо уменьшается (R' вкладывается в R с некоторым вложением $\varphi : R' \to R$, и $\phi\varphi = \mathrm{id}$), инвариант f преобразуется в инвариант $\phi(f)$, который определяется формулой $\phi(f)(x') = f(\varphi(x'))$. Если $\forall x \in R^n, g \in G$ $f(x) = f(gx)$, то

$$\forall x' \in R'^n, g \in G \quad \phi(f)(gx') = f(g\varphi(x')) = f(\varphi(x')) = \phi(f)f(x').$$

   А вот если кольцо увеличивается, с инвариантами могут происходить менее приятные вещи (поскольку в кольце появляются новые элементы, старые равенства могут уже не выполняться). Поэтому в качестве операции по расширению дерева групп мы

допускаем только уменьшение базового кольца. В дереве это соответствует вершине с одним потомком.

Кроме прочего, это свойство позволяет нам заключить, что каждый инвариант, известный из теории инвариантов над полями, останется инвариантом и над кольцами, являющимися подкольцами этих полей; например, всякий инвариант над $\mathbb{C}$ останется инвариантом и над $\mathbb{Z}$.

Однако изменение базового кольца требует особой осторожности насчёт пространства сообщений. Если в пространстве сообщений были такие разные представители $m \neq m' \in M$, что $\phi(m) = \phi(m')$, то соответствующие сообщения в новом пространстве сообщений $\phi(M)$ станут неразличимы. Поэтому есть смысл применять эту операцию только тогда, когда $\phi(M)$ остаётся нетривиальным.

2. *Сопряжение* $g \mapsto h^{-1}gh$. Инвариант $f(x)$ под действием сопряжения превращается в инвариант $f'(x) = f(hx)$. Если $\forall g \in G \forall x \in R^n$ $f(gx) = f(x)$, то $\forall g \in G$ $\forall x \in R^n$

$$f'(h^{-1}ghx) = f(hh^{-1}ghx) = f(g(hx)) = f(hx) = f'(x).$$

Пространство сообщений M остаётся неизменным.

3. *Прямое произведение* $G_1, G_2 \mapsto G_1 \times G_2$. Будем рассматривать естественное представление прямого произведения: если $G_1 \leq GL(n_1, F)$ и $G_2 \leq GL(n_2, F)$, то $G_1 \times G_2 \leq GL(n_1 + n_2, F)$, и эта группа действует покомпонентно (а составляющие её матрицы блочно–диагональны). В этой ситуации, если $f_1(x)$ и $f_2(x)$ были инвариантами G_1 и G_2 соответственно, то любой элемент $f \in \langle f_1(x), f_2(y)\rangle \leq R[x, y]$ будет инвариантом $G_1 \times G_2$. Можно выбрать случайный элемент этого множества, а пространство сообщений в любом случае превратится в $M_1 \times M_2$ (если столько различ-

ных сообщений не требуются, всегда можно выбрать несколько случайных из них, а остальные отбросить).

4. *Сплетение* $G \wr H$, где $G \leq GL(n, R)$, $H \leq S_m$. В этом случае мы рассматриваем естественное представление $G \wr H$ на R^{mn}; $G \wr H$ действует в нём как

$$(g_1, \ldots, g_m, \pi) \begin{pmatrix} x_1 \\ \ldots \\ x_m \end{pmatrix} = \begin{pmatrix} g_1 x_{\pi(1)} \\ \ldots \\ g_m x_{\pi(m)} \end{pmatrix}.$$

Тогда для любого инварианта f, если $\forall g \in G, x \in R^n\ f(gx) = f(x)$, то то же самое будет верно и для $G \wr H$, если рассмотреть f^m как действующий покомпонентно инвариант. Перестановка ничего не меняет в инвариантном равенстве. Пространство сообщений тогда вырастет до M^m (и вновь, если так много сообщений нам не нужно, мы можем либо выбрать несколько случайных, либо, например, оставить в качестве пространства сообщений только диагональ $\Delta = \{(x, \ldots, x) \mid x \in M\}$).

Кроме этих способов, соответствующих групповым операциям, теория инвариантов может предложить несколько новых способов. В качестве некоторых элементов наших преобразований троек $o \in \mathcal{O}$ мы возьмём преобразования, которые оставляют группу прежней ($o(G) = G$), а изменения касаются только инварианта f и пространства сообщений M. Следующие конструкции сработают, только если f — полиномиальный инвариант.

1. *Хессиан* $H(f)$. Если f — полиномиальный инвариант G, и $\forall g \in G \leq GL(n, F)\ \det g = \pm 1$ (отметим, что в этой конструкции F должно быть полем), то

$$H(f) = \det \left(\frac{\partial^2 f}{\partial z_i \partial z_j} \right)$$

также будет инвариантом G. Группа G и пространство сообщений M остаются без изменений.

2. *Якобиан* J. Если $f_1, \ldots, f_n$ — полиномиальные инварианты $G \leq SL(n, F)$ (отметим, что в этой конструкции F должно быть полем), то

$$J(f_1, \ldots, f_n) = \det\left(\frac{\partial f_i}{\partial z_j}\right)$$

также будет инвариантом G. Таким способом мы можем объединить n одинаковых групп с разными инвариантами в один набор; скорее всего, это преобразование будет наиболее полезно на первом уровне дерева, где мы вольны выбрать сколько угодно одинаковых групп.

4.7 Листья дерева

В предыдущем разделе мы показали, как построить новый инвариант из существующих (проделать рекурсивный шаг конструкции). Однако остаётся вопрос: что должно быть базой этой рекурсии? Что мы можем поместить в листьях дерева?

4.7.1 Общие замечания

Первое замечание состоит в том, что в информатике мы не можем на самом деле работать над $\mathbb{C}$ или $\mathbb{R}$. Всё, что мы делаем при помощи компьютера, на самом деле происходит над $\mathbb{Q}$. Теория инвариантов над $\mathbb{Q}$ существенно отличается от классической теории инвариантов над $\mathbb{C}$. К счастью, отказываться от неё тоже не понадобится: если функция f была инвариантом группы $G \leq GL(n, \mathbb{C})$, представленной в виде матриц с рациональными коэффициентами, она всё равно останется инвариантом группы $G \leq GL(n, \mathbb{Q})$, поскольку у элементов G рациональные коэффициенты. Поэтому в дальнейшем мы будем иногда

рассматривать инварианты над $\mathbb{C}$, но только в тех случаях, когда они без изменений переносятся на случай $\mathbb{Q}$.

Мы также могли бы рассматривать инварианты над конечными полями, так называемые *модулярные инварианты*; при этом теория заметно усложняется технически (ср. пример 4 в разделе 4.7.3).

4.7.2 Классы Черна

В качестве примера классической конструкции из теории инвариантов мы напоминаем конструкцию *классов Черна* (см., например, [120]). Классы Черна — один из самых больших известных источников инвариантов конечных групп. Идея проста: рассмотрим орбиту a^G некоторого элемента $a \in F^n$ (мы будем предполагать, что G действует над полем) и заметим, что произведение

$$\prod_{b \in a^G} (x + b),$$

где x — формальная переменная, инвариантно под действием G (элементы G просто переставляют сомножители в этом произведении). Коэффициенты этого многочлена и называются *классами Черна*. Например, $\sum_{b \in a^G} b$ — первый класс Черна — это инвариант группы G.

Все классы Черна являются просто симметрическими функциями элементов орбиты действия G. Если a рассмотреть как неизвестное (переменную), мы получим искомые инварианты. Аналогичные конструкции верны и для компактных групп.

4.7.3 Примеры инвариантов конечных групп

В этом подразделе мы приведём несколько примеров инвариантов различных конечных групп. Разумеется, это лишь несколько характерных примеров, их легко можно дополнить другими.

Пример 4.2. У симметрической группы S_n есть мономиальное представление $F^n\ S_n \to GL(n, F)$; в этом представлении элементы S_n переставляют переменные. Кольцо инвариантов в этом представлении порождается всеми симметрическими многочленами, от $x_1 + \ldots + x_n$ до $x_1 \ldots x_n$. Это простейший пример классов Черна.

Пример 4.3. Образующую циклической группу $\mathbb{Z}_n$ можно представить как матрицу $g \in GL(m, F)$, для которой $g^n = e$ (унипотентную матрицу подходящего порядка). Чтобы функция f была инвариантом представления циклической группы, достаточно проверить, что она не меняется под действием единственной образующей: $f(x) = f(gx)$.

Например, циклическая группа $\mathbb{Z}_n$ имеет естественное представление в виде группы, порождённой $\xi_n e$, где ξ_n — примитивный корень из единицы степени n, а e — единичная матрица. Очевидно, инвариантом этой группы будет любой однородный многочлен. Можно сделать следующий шаг и рассмотреть представление циклической группы $\mathbb{Z}_n$, порождённое матрицей

$$\begin{pmatrix} \xi_1 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & \xi_m \end{pmatrix},$$

где ξ_i — корни из единицы, $\xi_i^n = 1$. Тогда кольцом инвариантов этой группы будет кольцо многочленов $\mathbb{C}[x_1^n, \ldots, x_m^n]$.

Заметим, что инварианты зависят не только от самих групп, но и от их представлений; одна и та же группа в разных представлениях будет иметь разные инварианты.

Пример 4.4. Диэдральная группа D_{2k} имеет представление как группа симметрий правильного многоугольника $D_{2k} \to GL(2, \mathbb{R})$. В этом представлении D_{2k} порождается двумя матрицами:

$$D_{2k} = \left\langle \begin{pmatrix} \cos\frac{2\pi}{k} & -\sin\frac{2\pi}{k} \\ \sin\frac{2\pi}{k} & \cos\frac{2\pi}{k} \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \right\rangle.$$

Тогда кольцо инвариантов диэдральной группы в этом представлении порождается многочленами

$$q = x^2 + y^2, \qquad h = \prod_{i=0}^{k-1} \left(\left(\cos \frac{2\pi i}{k} \right) x + \left(\sin \frac{2\pi i}{k} \right) y \right).$$

Поскольку мы хотим вычислять над $\mathbb{Q}$, нам требуется каким-либо образом перевести непрерывные тригонометрические функции на язык рациональных чисел. Это можно сделать сугубо формальным алгебраическим путём, введя дополнительные функциональные символы sin и cos с известными соотношениями на них $\sin^2 x + \cos^2 x = 1$, формулы двойного угла и т.д.).

Пример 4.5. Для нечётного простого числа p диэдральная группа D_{2p} имеет представление $D_{2p} \to GL(2, \mathbb{F}_p)$ над конечным полем $\mathbb{F}_p$. Это представление порождается матрицами

$$D_{2k} = \left\langle \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \right\rangle.$$

В этом случае кольцо инвариантов изоморфно $\mathbb{F}_p[y, (xy^{p-1} - x^p)^2]$. Однако если перейти к дуальному представлению (просто транспонировав матрицы), кольцо инвариантов существенно изменится: оно теперь будет изоморфно $\mathbb{F}_p[x^2, y(y^{p-1} - x^{p-1})]$. В этом примере было важно, что группа была представлена над конечным полем степени, не взаимно простой со степенью группы.

Эти два примера показывают, как сильно инварианты зависят от конкретного представления. Несколько других примеров инвариантов в контексте криптосистем, основанных на инвариантах, можно найти в [146].

4.7.4 Инварианты классических групп

В этом подразделе мы представим несколько примеров хорошо известных инвариантов классических групп. Они также могут лежать в

листьях дерева.

Пример 4.6. Рассмотрим ортогональную группу чётной размерности $SO(2l, F)$. У неё есть хорошо известный *инвариант Диксона*: если $\mathrm{char} F \neq 2$, а мы предположим, что так и есть, он имеет вид $(-1)^{\det g}$ для $g \in SO(2l, F)$. Отметим, что у этого инварианта всего два значения, поэтому он хорош для кодирования только одного бита.

Пример 4.7. Симплектическая группа $Sp(2n, F)$ по определению сохраняет невырожденную кососимметрическую билинейную форму. Значение этой формы и будет инвариантом (и, в отличие от предыдущего примера, полиномиальным инвариантом).

4.8 Атаки на криптосистемы, основанные на инвариантах

Когда наша цель — представить новую криптосистему или семейство криптосистем, представляется естественным проанализировать возможные атаки на эти криптосистемы. В этом разделе мы рассмотрим несколько возможных атак на криптосистемы, основанные на инвариантах, а также дадим практические рекомендации о том, как от них защититься.

4.8.1 Атаки линейной алгеброй

Обычно самые тяжёлые атаки на алгебраические криптосистемы проходят посредством линейной алгебры: противник составляет систему линейных уравнений и находит секретный ключ (самый известный пример такого подхода взламывает схему Polly Cracker [46]; лишь в последнее время удалось улучшить эту схему так, чтобы атаки линейной алгеброй стали менее эффективными [88]).

Предположим, что инвариант f — это многочлен степени d. Тогда противник может рассмотреть его как многочлен с $\binom{n+d+1}{d}$ неопределёнными коэффициентами. Чтобы вычислить коэффициенты, про-

тивник должен решить уравнения $f(g_i m_j) = f(m_j)$ для всех элементов пространства сообщений $m_j \in M$ и всех образующих $g_i \in G$. Пространство решений этой системы даст противнику инвариант, разделяющий орбиты элементов M (вместе, конечно, с тривиальными инвариантами, такими, например, как $f = \mathrm{const}$). Если d константно, то эта атака действительно будет успешной, поэтому Алиса должна выбирать инварианты степени $d = d(n)$, растущей так, чтобы асимптотически $\binom{n+d+1}{d}$ было суперполиномиальной функцией от n.

Пример 4.8. Предположим, что мы пытаемся построить криптосистему, основанную на инвариантах, на базе мономиального представления симметрической группы S_n, порождённой транспозициями τ_{ij} и её инвариант первой степени

$$f(x_1, \ldots, x_n) = x_1 + \ldots + x_n.$$

В качестве пространства сообщений мы должны выбрать несколько векторов так, чтобы у них различались суммы коэффициентов; мы обозначим их через $m_i = (m_{i1}, \ldots, m_{in})$. Противник, применяющий атаку линейной алгеброй, просто рассмотрит многочлен

$$h = \lambda_1 x_1 + \ldots + \lambda_n x_n$$

и решит систему уравнений, удостоверяющих, что транспозиции не меняют h. Уравнение, соответствующее τ_{ij}, имеет вид $h(\tau_{ij} x) = h(x)$, что эквивалентно просто $\lambda_i = \lambda_j$. Таким образом, у противника получится вычислить правильный инвариант (возможно, умноженный на константу, что в данном случае не важно) после выполнения полиномиального алгоритма. Заметим, что для того чтобы эта атака не смогла преуспеть, можно было бы выбрать такие сообщения, чтобы в пространстве сообщений оказались сообщения с одинаковыми суммами элементов. Противнику не обязательно искать *тот самый* инвариант, ему достаточно найти *какой-нибудь* инвариант, разделяющий

векторы из M.

4.8.2 Атаки методом Монте-Карло и размеры орбит

Другая проблема происходит из размеров орбит элементов M. Действительно, предположим, что элемент $m \in M$ имеет орбиту m^G полиномиального размера. Тогда у противника есть полиномиальный шанс декодировать $E(m)$, просто пытаясь случайно выбрать элемент $g \in G$ и сравнивая $E(m)$ и gm. Таким образом, элементы пространства сообщений нужно выбирать так, чтобы у них были орбиты большого размера.

Пример 4.9. В качестве тривиального, но характерного примера рассмотрим пространство сообщений, состоящее из нулевого вектора и какого-нибудь другого вектора (при этом не важно, какая группа — всё будет выполняться для любой подгруппы $GL(n, F)$ и любого инварианта). Размер орбиты нулевого вектора равен 1, и противнику вообще ничего не надо делать: если он видит нулевой вектор, значит, и сообщения было нулевым; а если он видит ненулевой вектор, значит, мы «закодировали» что-то другое.

4.8.3 Атака, реконструирующая дерево групп

Наконец, противник может попытаться восстановить дерево, при помощи которого Алиса построила свой инвариант. На этом пути он может столкнуться с задачей, к примеру, поиска такой матрицы a, что $a^{-1}Ga = H$ для данных групп G и H. Это хорошо известная сложная задача; например, в [87] показано, что задача изоморфизма графов сводится к задаче сопряжения групп. Атаки такого рода были подробно рассмотрены в работе [58]; те же соображения можно применить и здесь, потому что задача восстановления дерева проще не стала. На

самом деле она стала сложнее, потому что теперь узлы дерева дополнены инвариантами, которые тоже могут нетривиально меняться от узла к узлу. Следовательно, для восстановления дерева придётся восстановить не только группы, его составляющие, но и инварианты.

4.9 Схема согласования ключа Аншель-Аншеля-Голдфельда и её устойчивость против взлома с доказательством

Прежде всего вспомним определение алгебраической схемы согласования ключа Аншель-Аншеля-Голдфельда [8]. Рассмотрим группу G, и пусть два игрока A и B выбирают некоторые подгруппы G, заданные своими образующими:

$$G_A = \langle a_1, \ldots, a_m \rangle, \qquad G_B = \langle b_1, \ldots, b_n \rangle.$$

Замечание 4.9. Отметим, что все рассуждения в этом разделе проходят без изменений, если вместо G_A и G_B рассматривать $\tilde{G}_A$ и $\tilde{G}_B$, которые порождаются теми же образующими

$$\langle a_1, \ldots, a_m \rangle \text{ и } \langle b_1, \ldots, b_n \rangle$$

соответственно, но не как группы, а как полугруппы. Все коммутаторы берутся в объемлющей группе G.

Группа G и элементы a_i, $1 \leq i \leq m$, и b_j, $1 \leq j \leq n$, выдаются публично. Оба игрока A и B случайно выбирают секретные элементы $a \in G_A$ и $b \in G_B$ как произведения не более чем N образующих, а затем передают друг другу следующую последовательность:

$$X_A = \{a^{-1} b_j a\}_{j=1}^{n}, \qquad X_B = \{b^{-1} a_i b\}_{i=1}^{m}.$$

У игрока A (соотв. B) изначально имеется представление элемента a (соотв. b) в подгруппе G_A (соотв. G_B), а после этой передачи появляется новый набор образующих $b^{-1} a_i b$ (соотв. $a^{-1} b_i a$), соответствующих исходным a_i. Значит, он может вычислить представление элемента

$b^{-1}ab$ (соотв. $a^{-1}ba$), используя элементы последовательности X_A (соотв. X_B). Таким образом, у обоих участников образуется общий ключ, а именно коммутатор

$$a^{-1}(b^{-1}ab) = [a, b] = (a^{-1}ba)^{-1}b.$$

Очевидное необходимое условие надёжности этого протокола состоит в том, что множество всех коммутаторов элементов $a \in G_A$ и $b \in G_B$ должно состоять как минимум из двух элементов.

Чтобы доказуемо взломать этот протокол согласования ключа, нужно найти представления определённых элементов a' в G_A и b' в G_B, где

$$X_A = \{a'^{-1}b_j a'\}_{j=1}^{n}, \qquad X_B = \{b'^{-1}a_i b'\}_{i=1}^{m}.$$

Теорема 4.2 Рассмотрим модулярную группу $G = SL_2(\mathbb{Z})$. Существуют такие подгруппы $G_A \leq G$ и $G_B \leq G$, что если протокол согласования ключа Аншель-Аншеля-Голдфельда для группы G и её подгрупп G_A и G_B подвержен доказуемому взлому в худшем случае, то $NP \subseteq RP$. То же самое верно, если вместо G_A и G_B рассмотреть $\tilde{G}_A$ и $\tilde{G}_B$, порождённые теми же образующими $\langle a_1, \ldots, a_m\rangle$ и $\langle b_1, \ldots, b_n\rangle$ соответственно, но не как группы, а как полугруппы.

Доказательство. Предположим, что есть такая вероятностная машина Тьюринга M, работающая полиномиально долго, что для бесконечной последовательности параметров надёжности N и входа

$$I = \{a_1, \ldots, a_m, b_1, \ldots, b_n, a^{-1}b_1a, \ldots, a^{-1}b_ma, b^{-1}a_1b, \ldots, b^{-1}a_nb\}$$

выполняется следующее условие:

$$\Pr[M(I) = a'_1, s_1, \ldots, a'_f, s_f, b'_1, t_1, \ldots, b'_g, t_g] \geq 1/p(N),$$

где $G_A = \langle a_1, \ldots, a_m\rangle$ и $G_B = \langle b_1, \ldots, b_n\rangle$ — подгруппы модулярной группы, p — некоторый полином, $a \in G_A$, $b \in G_B$, $a' = \prod_{i=1}^{f} a_i'^{s_i}$,

$b' = \prod_{j=1}^{g} b_j^{'t_j}$, $a_i' \in \{a_i\}_{i=1}^{m}$, $b_j' \in \{b_j\}_{j=1}^{n}$, $a'^{-1} b_j a' = a^{-1} b_j a$ для всех $1 \le j \le n$, $b'^{-1} a_i b' = b^{-1} a_i b$ для всех s_i, а t_j лежат в $\{-1, 1\}$ для всех $1 \le i \le f$ и $1 \le j \le g$, $f, g \le N$. Отметим, что мы можем проверить корректность ответа M, поэтому мы предполагаем, что M никогда не ошибается (но может иногда выдавать ответ «не знаю»).

Используя M, можно построить вероятностную полиномиальную машину Тьюринга M', которая содержит $p(N)/2$ копий M и на входе $(X, \{Y_i\}_i, N)$ для некоторого $X \in G$ выполняет следующий алгоритм.

1. Если $X = \prod_{i=1}^{m} Y_i^{'s_i}$, где $Y' \in \{Y_i\}_i$, $m \le N$, $s_i \in \{-1, 1\}$ (если рассматривать G_A и G_B как полугруппы, здесь нужно брать только положительные степени), то $\Pr[M' \text{ принимает}] \ge 1/2$.
2. В противном случае, $\Pr[M' \text{ принимает}] = 0$.

На вход всем копиям M мы подадим $a = b = X, a_i = b_i = Y_i$, а затем вычислим все $a^{-1} b_1 a, \ldots, a^{-1} b_m a$, $b^{-1} a_1 b, \ldots, b^{-1} a_n b$ за полиномиальное время. По [23, Следствие 11.5], задача BM NP-полна, следовательно, $NP \subseteq RP$.

□

Замечание 4.10. Если $\tilde{G}_A$ и $\tilde{G}_B$ — полугруппы, то задача BM сложна, более того, в среднем [128].

Заметим, что описанный протокол согласования ключа может быть неустойчив против атак линейной алгеброй (см. подраздел 4.8.1): он даёт противнику решение задачи сопряжения, которое может оказаться единственным, при условии, что порождённое G_A (или G_B) кольцо совпадает с полным матричным кольцом. Чтобы этот протокол стал более устойчив против атак линейной алгеброй, можно заменить G древовидной конструкцией из групп или полугрупп, как это делалось в разделе 4.6.

Формально говоря, мы строим некую рекурсивную конструкцию для класса групп $\mathcal{G}$, замкнутого относительно некоторого набора тео-

ретико–групповых операций $\mathcal{O}$; на этот раз нам не нужно беспокоиться о допустимых наборах, и все операции определены просто на группах из класса $\mathcal{G}$. Для множества $\mathcal{G}_0 \subset \mathcal{G}$ (базы нашей конструкции) мы рекурсивно определяем класс $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ пар (G, T).

Рекурсивное определение делается точно так же, как в 4.6.2, опуская конструкции инвариантов и пространств сообщений. В нашем случае, множество $\mathcal{O}$ допустимых операций состоит из изменения базового кольца, прямых произведений, сплетений и сопряжений (всё так же, как для криптосистем, основанных на инвариантах, но без специфических операций над инвариантами).

Для матричных групп надёжность протокола согласования ключа Аншель-Аншеля-Голдфельда основана на следующей задаче.

Задача 4.2 *Задача линейной транспортировки* (Linear Transporter Problem, LTP). Рассмотрим коммутативное кольцо R и R–модуль V. Пусть $G \leq GL(V, R)$. По данным $u \in V$ и $v \in u^G = \{u^g : g \in G\}$ найти такой элемент $g \in G$, что $v = u^g$.

Если противник может эффективно решить LTP, он может взломать протокол согласования ключа Аншель-Аншеля-Голдфельда. В [59] было доказано следующее утверждение (Лемма 3.4).

Утверждение 4.1 Пусть $G \in \mathcal{G}$. Тогда по данному дереву G, LTP для G можно решить за время, полиномиальное от размера дерева и времени решения задачи LTP для листьев дерева.

Литература

[1] Advances in Elliptic Curves in Cryptography / Ed. by I. Blake, G. Seroussi, N. Smart. Cambridge University Press, 2005. Vol. 317 of *London Mathematical Society*.

[2] *Agrawal M., Kayal N., Saxena N.* PRIMES is in P // Annals of Mathematics. 2004. Vol. 160, N. 2. P. 781–793.

[3] *Ajtai M.* Σ^1_1-formulae on finite structures // Annals of Pure and Applied Logic. 1983. Vol. 24. P. 1–48.

[4] *Ajtai M., Dwork C.* A public-key cryptosystem with worst-case/average-case equivalence // Proceedings of the 29^{th} Annual ACM Symposium on Theory of Computing. 1997. P. 284–293.

[5] *Al-Kadi I. A.* The origins of cryptology: The Arab contributions // Cryptologia. April 1992. Vol. 16, N. 2. P. 97–126.

[6] *Allender E.* Circuit Complexity before the Dawn of the New Millennium // Proceedings of the 16^{th} Conference on Foundations of Software Technology and Theoretical Computer Science. 1996. P. 1–18.

[7] *Anshel I., Anshel M., Fisher B., Goldfeld D.* New Key Agreement Protocols in Braid Group Cryptography // Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA, Lecture Notes in Computer Science. Vol. 2020. 2001. P. 13–27.

[8] *Anshel I., Anshel M., Goldfeld D.* An algebraic method for public-key cryptography // Mathematical Research Letters. 1999. Vol. 6. P. 287–291.

[9] *Anshel I., Anshel M., Goldfeld D.* Non-abelian key agreement protocols // Discrete Applied Mathematics. 2003. Vol. 130, N. 1. P. 3–12.

[10] *Anshel M.* Braid group cryptography and quantum cryptoanaly-

sis // Proceedings of the 8^{th} International Wigner Symposium. 2003. P. 13–27.

[11] *Apostol T. M.* Modular Functions and Dirichlet Series in Number Theory. Springer, New York, 1990.

[12] *Arora S., Barak B.* Complexity Theory: A Modern Approach. Cambridge University Press. В печати. Черновик свободно доступен по адресу `http://www.cs.princeton.edu/theory/complexity/`, 2009.

[13] *Bach E.* Explicit bounds for primality testing and related problems // Mathematics of Computation. 1990. Vol. 55, N. 191. P. 355–380.

[14] *Bauer F. L.* Decrypted Secrets: Methods and Maxims of Cryptology. Springer, 1997.

[15] *Benaloh J.* Dense Probabilistic Encryption // Proceedings of the 1st Annual Workshop on Selected Areas in Cryptology. 1994. P. 120–128.

[16] *Bennett C. H., Bernstein E., Brassard G., Vazirani U.* Strengths and weaknesses of quantum computing // SIAM Journal of Computing. 1997. Vol. 26, N. 5. P. 1510–1523.

[17] *Bennett C. H., Bessette F., Brassard G., Salvail L., Smolin J.* Experimental Quantum Cryptography // Journal of Cryptology. 1992. Vol. 5, N. 1. P. 3–28.

[18] *Bennett C. H., Brassard G.* Quantum Cryptography: Public key distribution and coin tossing // Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing. 1984. P. 175.

[19] *Bennett C. H., Shor P. W.* Quantum Information Theory // IEEE Transactions on Information Theory. 1998. Vol. 44, N. 6. P. 2724–2742.

[20] *Berger R.* The undecidability of the domino problem. 1966. Vol. 66 of *Memoirs of the American Mathematical Society.*

[21] *Biham E., Shamir A.* Differential Cryptanalysis of the Data Encryption Standard. Springer Verlag, 1993.

[22] *Blake I., Seroussi G., Smart N.* Elliptic Curves in Cryptography. Cambridge University Press, 1999. Vol. 265 of *London Mathematical Society.*

[23] *Blass A., Gurevich Y.* Matrix Transformation is Complete for the Average Case // SIAM Journal of Computing. 1995. Vol. 24, N. 1. P. 3–29.

[24] *Blum N.* A Boolean Function Requiring 3n Network Size // Theoretical Computer Science. 1984. Vol. 28. P. 337–345.

[25] *Bogdanov A., Trevisan L.* On Worst-Case to Average-Case Reductions for NP Problems // Proceedings of the 44^{th} Annual IEEE Symposium on the Foundations of Computer Science. 2003. P. 308–317.

[26] *Bogdanov A., Trevisan L.* Average-Case Complexity // Foundations and Trends in Theoretical Computer Science / Ed. by M. Sudan. 2006. Vol. 2.

[27] *Boneh D., Venkatesan R.* Breaking RSA may not be Equivalent to Factoring // Proceedings of the 17^{th} Annual Eurocrypt Conference, Lecture Notes in Computer Science. Vol. 1233. 1998. P. 59–71.

[28] *Book R. V., Otto F.* String Rewriting Systems. Springer-Verlag, 1993.

[29] *Brown D. R. L.* Breaking RSA May Be as Difficult as Factoring: Tech. Rep. 2005/380: Cryptology ePrint Archive, 2005.

[30] *Cai J.* With probability 1, a random oracle separates PSPACE from the polynomial-time hierarchy // Journal of Computer and System Sciences. 1989. Vol. 38. P. 68–85.

[31] *Church A.* An Unsolvable Problem of Elementary Number Theory // American Journal of Mathematics. 1936. Vol. 58. P. 345–363.

[32] *Cook S. A.* The complexity of theorem-proving procedures // Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. 1971. P. 151–158.

[33] *Copeland B. J.* The Church-Turing Thesis // The Stanford Encyclopedia of Philosophy / Ed. by E. N. Zalta. Fall 2002. http://plato.stanford.edu/archives/fall2002/entries/church-turing/.

[34] *Coppersmith D.* The data encryption standard (DES) and its strength against attacks // IBM Journal of Research and Development. 1994. Vol. 38, N. 3. P. 243–250.

[35] *Crandall R., Pomerance C.* Prime Numbers: A Computational Perspective. Springer, 2005.

[36] *Culik K.* An aperiodic set of 13 Wang tiles // Discrete Mathematics. 1996. Vol. 160. P. 245–251.

[37] *Culik K., Kari J.* An aperiodic set of Wang cubes // Journal of Universal Computer Science. 1995. Vol. 1. P. 675–686.

[38] *Damgard I., Landrock P., Pomerance C.* Average case error estimates for the strong probable prime test // Mathematics of Computation. 1993. Vol. 61, N. 203. P. 177–194.

[39] *Diffie W., Hellman M.* New directions in cryptography // IEEE Transactions on Information Theory. 1976. Vol. IT-22. P. 644–654.

[40] *Diffie W., Hellman M.* Exhaustive Cryptanalysis of the NBS Data Encryption Standard // IEEE Computer. 1977. Vol. 10, N. 6. P. 76–84.

[41] *DiVincenzo D. P.* Quantum Computation // Science. 1995. Vol. 270, N. 5234. P. 255–261.

[42] *Dwork C.* Positive Applications of Lattices to Cryptography // Proceedings of the 22nd International Symposium on Mathemati-

cal Foundations of Computer Science, Lecture Notes in Computer Science. Vol. 1295. 1997. P. 44–51.

[43] *Ehrsam, W. F. et al.* Product Block Cipher System for Data Security. United States Patent $3,962,539$. February 1975.

[44] *Evan S., Yacobi Y.* Cryptography and NP-completeness // 7th International Colloquium on Automata, Languages and Programming (ICALP'80). 1980. P. 195–207.

[45] *Feigenbaum J., Merritt M.* Open questions, talk abstracts, and summary of discussions // DIMACS series in discrete mathematics and theoretical computer science. AMS, 1991. Vol. 2. P. 1–45.

[46] *Fellows M., Koblitz N.* Combinatorial cryptosystems galore! // Contemporary Mathematics. 1994. Vol. 168. P. 51–61.

[47] *Feynman R.* Simulating physics with computers // International Journal of Theoretical Physics. 1982. Vol. 21. P. 467.

[48] *Furst M., Saxe J., Sipser M.* Parity, circuits, and the polynomial-time hierarchy // Mathematical Systems Theory. 1984. Vol. 17. P. 13–27.

[49] *Garey M. R., Johnson D. S.* Computers and Intractability, A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco, CA, 1979.

[50] *Goldreich O.* Introduction to Complexity Theory. Lecture Notes. Weizmann Institute of Science, 1998–99.

[51] *Goldreich O.* Foundations of Cryptography. Basic Tools. Cambridge University Press, 2001.

[52] *Goldreich O.* Foundations of Cryptography II. Basic Applications. Cambridge University Press, 2004.

[53] *Goldwasser S., Bellare M.* Lecture notes on cryptography. Summer course on cryptography at MIT, 2001.

[54] *Goldwasser S., Micali S.* Probabilistic Encryption // Journal of

Computer and System Sciences. 1984. Vol. 28. P. 270–299.

[55] *Grigoriev D., Hirsch E. A., Pervyshev K.* A Complete Public-Key Cryptosystem: Tech. Rep. 006-046: Electronic Colloquium on Computational Complexity. To appear in *Groups, Complexity, and Cryptology*, 2006.

[56] *Grigoriev D., Kojevnikov A. A., Nikolenko S. I.* Invariant-Based Cryptosystems and Their Security Against Provable Break: Tech. Rep. 158: Max-Planck-Institüt preprints, 2007.

[57] *Grigoriev D., Ponomarenko I. N.* Homomorphic Public-Key Cryptosystems over Groups and Rings // Quaderni di Mathematica. 2004. Vol. 13. P. 305–325.

[58] *Grigoriev D., Ponomarenko I. N.* Homomorphic Public-Key Cryptosystems and Encrypting Boolean Circuits // Applicable Algebra in Engineering, Communication, and Computing. 2006. Vol. 17. P. 239–255.

[59] *Grigoriev D., Ponomarenko I. N.* Constructions in public-key cryptography over matrix groups // Contemporary Mathematics: Algebraic Methods in Cryptography / Ed. by L. Gerritzen, D. Goldfeld, M. Kreuzer, R. Gerhard, V. Shpilrain. Providence, RI: AMS, 2007. Vol. 418. P. 103–120.

[60] *Gu J., Purdom P. W., Franco J., Wah B. W.* Algorithms for the Satisfiability Problem. Cambridge University Press, 2000.

[61] *Gurari E.* An Introduction to the Theory of Computation. Computer Science Press, 1989.

[62] *Gurevich Y.* Complete and incomplete randomized NP problems // Proceedings of the 28^{th} Annual IEEE Symposium on the Foundations of Computer Science. 1987. P. 111–117.

[63] *Gurevich Y.* Average case completeness // Journal of Computer and System Sciences. 1991. Vol. 42, N. 3. P. 346–398.

[64] Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity / Ed. by J. van Leeuwen. The MIT Press/Elsevier, 1990.

[65] *Hankerson D., Menezes A., Vanstone S.* Guide to Elliptic Curve Cryptography. Springer-Verlag, 2004.

[66] *Harnik D., Kilian J., Naor M., Reingold O., Rosen A.* On robust combiners for oblivious transfers and other primitives // Proceedings of EuroCrypt '05, Lecture Notes in Computer Science. Vol. 3494. 2005. P. 96–113.

[67] *Hastad J.* Computational Limitations for Small Depth Circuits. MIT Press, Cambridge, MA, 1987.

[68] *Hellman M., Diffie W., Merkle R. C.* Cryptographic apparatus and method. United States Patent 4,200,770. April 1980.

[69] *Hiltgen A. P.* Constructions of Freebly-One-Way Families of Permutations // Proc. of AsiaCrypt '92. 1992. P. 422–434.

[70] *Hiltgen A. P.* Towards a Better Understanding of One-Wayness: Facing Linear Permutations // Proceedings of EuroCrypt '98, Lecture Notes in Computer Science. Vol. 1233. 1998. P. 319–333.

[71] *Hopcroft J. E., Ullman J. D.* Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading Mass, 1979.

[72] *Immerman M.* Languages which capture complexity classes // SIAM Journal of Computing. 1987. Vol. 4. P. 760–778.

[73] *Jaeger G.* Quantum Information: An Overview. Berlin: Springer, 2006.

[74] *Kahn D.* The Codebreakers. New York, Macmillan, 1967.

[75] *Karl J.* A small aperiodic set of Wang tiles // Discrete Mathematics. 1996. Vol. 160. P. 259–264.

[76] *Karp R. M.* Reducibility Among Combinatorial Problems // Complexity of Computer Computations. New York: Plenum, 1972.

[77] *Katz J., Lindell Y.* Introduction to Modern Cryptography. CRC Press, 2007.

[78] *Kelly T.* The myth of the Skytale // Cryptologia. July 1998. P. 244–260.

[79] *Kitaev A.* Quantum computations: algorithms and error correction // Russian Mathematical Surveys. 1997. Vol. 52, N. 6. P. 53–112.

[80] *Koblitz N.* Elliptic Curve Cryptosystems // Mathematics of Computation. 1987. Vol. 48. P. 203–209.

[81] *Kojevnikov A. A., Nikolenko S. I.* New Combinatorial Complete One-Way Functions // Proceedings of the 25^{th} Symposium on Theoretical Aspects of Computer Science. Bordeaux, France, 2008. P. 457–466.

[82] *Lamagna E. A., Savage J. E.* On the logical complexity of symmetric switching functions in monotone and complete bases: Tech. rep. Rhode Island: Brown University, July 1973.

[83] *Lempel A.* Cryptography in transition // Computing Surveys. 1979. Vol. 11, N. 4. P. 215–220.

[84] *Levin L. A.* Average Case Complete Problems // SIAM Journal of Computing. 1986. Vol. 15, N. 1. P. 285–286.

[85] *Levin L. A.* One-Way Functions and Pseudorandom Generators // Combinatorica. 1987. Vol. 7, N. 4. P. 357–363.

[86] *Lo H.-K., Spiller T., Popescu S.* Introduction to Quantum Computation and Information. World Scientific Publishing Company, 1998.

[87] *Luks E. M.* Permutation Groups and Polynomial-Time Computation // DIMACS Series in Discrete Mathematics and Theoretical Computer Science / (DIMACS, 1991). Vol. 11. AMS, 1993. P. 139–175.

[88] *Ly L. V.* Polly Two: A New Algebraic Polynomial-based Public-Key

Scheme // Applicable Algebra in Engineering, Communication, and Computing. 2006. Vol. 17. P. 267–283.

[89] *Menezes A., van Oorschot P., Vanstone S.* Handbook of Applied Cryptography. Boca Raton, Florida: CRC Press, 1997.

[90] *Miller G. L.* Riemann's Hypothesis and Tests for Primality // Journal of Computer and System Sciences. 1976. Vol. 13, N. 3. P. 300–317.

[91] *Miller V. S.* Use of Elliptic Curves in Cryptography // Proceedings of Crypto '85. Vol. 218 of *Lecture Notes in Computer Science*. 1985. P. 417–426.

[92] *Myasnikov A. G., Shpilrain V., Ushakov A.* Group–based cryptography. Birkhäuser, 2008.

[93] *Naccache D., Stern J.* A new public-key cryptosystem based on higher residues // Proceedings of the 5th ACM Conference on Computer and Communication Security. 1998. P. 59–66.

[94] *Nielsen M. A., Chuang I. L.* Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[95] *Okamoto T., Uchiyama S.* A New Public-Key Cryptosystem as Secure as Factoring // Proceedings of EuroCrypt'98, Lecture Notes in Computer Science. Vol. 1403. 1998. P. 308–317.

[96] *Papadimitriou C. H.* Complexity Theory. Addison Wesley, 1994.

[97] *Paul W. J.* A $2,5n$ lower bound on the combinational complexity of Boolean functions // SIAM Journal of Computing. 1977. Vol. 6. P. 427–443.

[98] *Pomerance C.* A Tale of Two Sieves // Notices of the American Mathematical Society. 1996. Vol. 43, N. 12. P. 1473–1485.

[99] *Post E.* Finite Combinatory Processes — Formulation 1 // Journal of Symbolic Logic. 1936. Vol. 1. P. 103–105.

[100] *Post E.* A variant of a recursively unsolvable problem // Bulletin of

the American Mathematical Society. 1946. Vol. 52. P. 264–268.

[101] *Post E.* Recursive Unsolvability of a Problem of Thue // Journal of Symbolic Logic. 1947. Vol. 12. P. 1–11.

[102] *Rappe D. K.* Algebraisch homomorphe Kryptosysteme. Diplomarbeit, Dem Fachbereich Mathematik der Universitat Dortmund. 2000.

[103] *Razborov A. A.* Bounded arithmetic and lower bounds // Feasible Mathematics II / Ed. by P. Clote, J. Remmel. Birkhäuser, 1995. Vol. 13 of *Progress in Computer Science and Applied Logic*. P. 344–386.

[104] *Regev O.* On lattices, learning with errors, random linear codes, and cryptography // Proceedings of the 37^{th} Annual ACM Symposium on Theory of Computing. 2005. P. 84–93.

[105] *Regev O.* Lattice-Based Cryptography // Proceedings of the 26^{th} Annual International Cryptology Conference (CRYPTO'06), Lecture Notes in Computer Science. Vol. 4117. 2006. P. 131–141.

[106] *Rivest R. L., Kaliski B.* RSA Problem // Encyclopedia of Cryptography and Security. Kluwer Publishing House, 2005.

[107] *Rivest R. L., Shamir A., Adleman L.* A Method for Obtaining Digital Signatures and Public-Key Cryptosystems // Communications of the ACM. 1978. Vol. 21, N. 2. P. 120–126.

[108] *Ruohonen K.* On some variants of Post's correspondence problem // Acta Informatica. 1983. Vol. 19, N. 4. P. 357–367.

[109] *Savage J. E.* The Complexity of Computing. New York: Wiley, 1976.

[110] *Shamir A.* Factoring Numbers in $O(\log n)$ Arithmetic Steps // Information Processing Letters. 1979. Vol. 8, N. 1. P. 28–31.

[111] *Shannon C. E.* A Symbolic Analysis of Relay and Switching Circuits. M. Sc. thesis / Massachusetts Institute of Technology, Dept. of Electrical Engineering. 1940.

[112] *Shannon C. E.* A Mathematical Theory of Communication // Bell System Technical Journal. 1948. Vol. 27, N. 4. P. 379–423, 623–656.

[113] *Shannon C. E.* Communication Theory of Secrecy Systems // Bell System Technical Journal. 1949. Vol. 28, N. 4. P. 656–717.

[114] *Shannon C. E.* The Synthesis of Two-Terminal Switching Circuits // Bell System Technical Journal. 1949. Vol. 28. P. 59–98.

[115] *Shannon C. E., Riordan J.* The Number of Two-Terminal Series-Parallel Networks // Journal of Mathematics and Physics. 1942. Vol. 31. P. 83–93.

[116] *Shor P. W.* Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer // SIAM Journal of Computing. 1997. Vol. 26, N. 5. P. 1484–1509.

[117] *Singh S.* The Code Book: the evolution of secrecy from Mary Queen of Scots to quantum cryptography. New York: Doubleday, 1997.

[118] *Sipser M.* On Relativization and the Existence of Complete Sets // Proceedings of ICALP'82, Lecture Notes in Computer Science. Vol. 140. 1982. P. 523–531.

[119] *Sipser M.* Introduction to the Theory of Computation. Course Technology, 2005.

[120] *Smith L.* Polynomial Invariants of Finite Groups. A. K. Peters, Wellesley, Massachusets, 1996. Vol. 6 of *Research Notes in Mathematics*.

[121] *Smolensky R.* Algebraic methods in the theory of lower bounds for Boolean circuit complexity // Proceedings of the 19^{th} Annual ACM Symposium on Theory of Computing. 1987. P. 77–82.

[122] *Solovay R. M., Strassen V.* A fast Monte-Carlo test for primality // SIAM Journal of Computing. 1977. Vol. 6, N. 1. P. 84–85.

[123] *Stockmeyer L.* The complexity of decision problems in automata theory and logic: Ph.D. thesis / Massachusetts Institute of Technol-

ogy. 1974.

[124] *Stockmeyer L.* On the combinational complexity of certain symmetric Boolean functions // Mathematical Systems Theory. 1977. Vol. 10. P. 323–326.

[125] *Stockmeyer L.* Classifying the computational complexity of problems // Journal of Symbolic Logic. 1987. Vol. 52. P. 1–43.

[126] The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions / Ed. by M. Davis. Raven Press, New York, 1965.

[127] *Turing A.* On Computable Numbers, with an Application to the Entscheidungsproblem // Proceedings of the London Mathematical Society. Series 2. 1936. Vol. 42. P. 230–265.

[128] *Venkatesan R., Rajagopalan S.* Average Case Intractability of Matrix and Diophantine Problems // Proceedings of the 24^{th} Annual ACM Symposium on Theory of Computing. 1992. P. 632–642.

[129] *Vernam G. S.* Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications // Journal of the IEEE. 1926. Vol. 55. P. 109–115.

[130] *Vollmer H.* Introduction to Circuit Complexity: a Uniform Approach. Springer Verlag, 1999.

[131] *Wang H.* Proving theorems by pattern recognition—II // Bell System Technical Journal. 1961. Vol. 40, N. 1. P. 1–41.

[132] *Wang J.* Random instances of bounded string rewriting are hard // Journal of Computing and Information. 1995. Vol. 1, N. 1. P. 11–23.

[133] *Wang J.* Average-case intractible NP problems // Advances in Languages, Algorithms, and Complexity. 1997. P. 313–378.

[134] *Wang J.* Distributional Word Problem for Groups // SIAM Journal of Computing. 1999. Vol. 28, N. 4. P. 1264–1283.

[135] *Wang J., Belanger J.* On the NP-Isomorphism Problem with Re-

spect to Random Instances // Journal of Computer and System Sciences. 1995. Vol. 50, N. 1. P. 151–164.

[136] *Washington L.* Elliptic Curves: Number Theory and Cryptography. Chapman & Hall / CRC, 2003.

[137] *Wegener I.* The Complexity of Boolean Functions. B. G. Teubner, and John Wiley & Sons, 1987.

[138] *Yao A. C.-C.* Separating the polynomial-time hierarchy by oracles // Proceedings of the 26^{th} Annual IEEE Symposium on the Foundations of Computer Science. 1985. P. 1–10.

[139] *Yao A. C.-C.* How to Generate and Exchange Secrets // Proceedings of the 27^{th} Annual IEEE Symposium on the Foundations of Computer Science. 1986. P. 162–187.

[140] *Yao A. C.-C.* On ACC and threshold circuits // Proceedings of the 31^{st} Annual IEEE Symposium on the Foundations of Computer Science. 1990. P. 619–627.

[141] *Валиев К. А.* Квантовая информатика: компьютеры, связь и криптография // Вестник Российской Академии Наук. 2000. Т. 70, № 8. С. 688–695.

[142] *Верещагин Н. К., Шень А.* Логические формулы и схемы // Математическое просвещение. Третья серия. 2000. Т. 4. С. 53–80.

[143] *Всемирнов М. А., Гирш Э. А., Данцин Е. Я., Иванов С. В.* Алгоритмы для пропозициональной выполнимости и верхние оценки их сложности // Записки научных семинаров ПОМИ. 2001. Т. 277. С. 14–46.

[144] *Гирш Э. А., Николенко С. И.* Надежная в слабом смысле функция с секретом // Препринты ПОМИ. 2008. № 16.

[145] *Горбатов В. А.* Фундаментальные основы дискретной математики. Информационная математика. М.: Наука. Физматлит, 2000.

[146] *Григорьев Д. Ю.* Криптография с открытым ключом и теория инвариантов // Записки научных семинаров ПОМИ. 2002. Т. 293. С. 26–38.

[147] *Григорьев Д. Ю., Кожевников А. А., Николенко С. И.* Алгебраическая криптография: новые конструкции и их надёжность относительно доказуемого взлома // Алгебра и Анализ. 2008. Т. 20, № 6. С. 119–147.

[148] *Григорьев Д. Ю., Пономаренко И. Н.* О неабелевых гомоморфных криптосистемах с открытым ключом // Записки научных семинаров ПОМИ. 2002. Т. 293. С. 39–58.

[149] *Китаев А., Шень А., Вялый М.* Классические и квантовые вычисления. М., МЦНМО, 1999.

[150] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М., МЦНМО, 2004.

[151] *Левин Л. А.* Универсальные переборные задачи // Проблемы передачи информации. 1973. Т. 9, № 3. С. 265–266.

[152] *Левин Л. А.* Односторонние функции // Проблемы передачи информации. 2003. Т. 39, № 1. С. 92–103.

[153] *Лупанов О. Б.* Об одном подходе к синтезу управляющих систем - принципе локального кодирования // Проблемы кибернетики. 1965. Т. 14. С. 31–110.

[154] *Марков А. А.* О минимальных контактно-вентильных двухполюсниках для монотонных симметрических функций // Проблемы кибернетики. 1962. Т. 8. С. 117–121.

[155] *Нечипорук Э. И.* Об одной булевской функции // Доклады Академии Наук СССР. 1966. Т. 169, № 4. С. 765–766.

[156] *Разборов А. А.* Нижние оценки монотонной сложности логического перманента // Математические заметки. 1985. Т. 37, № 6. С. 887–900.

[157] *Разборов А. А.* Нижние оценки размера схем ограниченной глубины в полном базисе, содержащем функцию логического сложения // Математические заметки. 1987. Т. 41, № 4. С. 598–608.

[158] *Разборов А. А.* Нижние оценки сложности реализации симметрических булевых функций контактно-вентильными схемами // Математические заметки. 1990. Т. 48, № 6. С. 79–90.

[159] *Разборов А. А.* О сложности вычислений // Математическое просвещение. Третья серия. 1999. Т. 3. С. 127–141.

[160] *Субботовская Б. А.* О реализации линейных функций формулами в базисе $\vee$, &, $\neg$ // Доклады Академии Наук СССР. 1961. Т. 136, № 3. С. 553–555.

[161] *Субботовская Б. А.* О сравнении базисов при реализации функций алгебры логики формулами // Доклады Академии Наук СССР. 1963. Т. 149, № 4. С. 784–787.

[162] *Хопкрофт Д., Мотвани Р., Ульман Д.* Введение в теорию автоматов, языков и вычислений. М., «Вильямс», 2002.

[163] *Храпченко В. М.* О сложности реализации линейной функции в классе π-схем // Математические заметки. 1971. Т. 9, № 1. С. 36–40.

[164] *Шоломов Л. А.* О реализации недоопределённых булевых функций схемами из функциональных элементов // Проблемы кибернетики. 1969. Т. 21. С. 215–226.

[165] *Яблонский С. В.* О классах функций алгебры логики, допускающих простую схемную реализацию // Успехи математических наук. 1957. Т. 12, № 6. С. 189–196.