Криптография и теория сложности

Н. П. Варновский

В небольшой по объему журнальной статье невозможно дать систематическое изложение основ какой-либо математической теории. Поэтому основное внимание мы уделяем разъяснению важнейших идей, связанных с применением теоретико-сложностного подхода в криптографии. Изложение по необходимости недостаточно формальное — для математической криптографии типичны многостраничные определения. Предполагается знакомство читателя с основами теории сложности вычислений: понятиями машины Тьюринга, классов P и NP (см. [2]), а также со статьей В. В. Ященко в настоящем номере журнала, стр. 53–70.

1. Введение

В теоретической криптографии существуют два основных подхода к определению стойкости криптосистем и криптографических протоколов (в дальнейшем мы будем также использовать общий термин — криптографические схемы): теоретико-информационный и теоретико-сложностной. Теоретико-информационный подход предполагает, что противник, атакующий криптографическую схему, не имеет даже теоретической возможности получить информацию, достаточную для осуществления своих целей. Классическим примером здесь может служить шифр Вернама с одноразовыми ключами, абсолютно стойкий против пассивного противника.

Подавляющее большинство используемых на практике криптографических схем не обладает столь высокой стойкостью. Более того, обычно бывает несложно указать алгоритм, который выполняет стоящую перед противником задачу, но не практически, а в принципе. Рассмотрим следующий пример.

Пример 1 (Криптосистема с открытым ключом). *Криптосистема с открытым ключом* полностью определяется тремя алгоритмами: генерации ключей, шифрования и дешифрования. Алгоритм генерации ключей G общедоступен; всякий желающий может подать ему на вход случайную строку r надлежащей длины и получить пару ключей (K_1, K_2).

Открытый ключ K_1 публикуется, а секретный ключ K_2 и случайная строка r хранятся в секрете. Алгоритмы шифрования E_{K_1} и дешифрования D_{K_2} таковы, что если (K_1, K_2) — пара ключей, сгенерированная алгоритмом G, то $D_{K_2}(E_{K_1}(m)) = m$ для любого открытого текста m. Для простоты изложения предполагаем, что открытый текст и криптограмма имеют одинаковую длину n. Кроме того, считаем, что открытый текст, криптограмма и оба ключа являются строками в двоичном алфавите.

Предположим теперь, что противник атакует эту криптосистему. Ему известен открытый ключ K_1, но неизвестен соответствующий секретный ключ K_2. Противник перехватил криптограмму d и пытается найти сообщение m, где $d = E_{K_1}(m)$. Поскольку алгоритм шифрования общеизвестен, противник может просто последовательно перебрать все возможные сообщения длины n, вычислить для каждого такого сообщения m_i криптограмму $d_i = E_{K_1}(m_i)$ и сравнить d_i с d. То сообщение, для которого $d_i = d$, и будет искомым открытым текстом. Если повезет, то открытый текст будет найден достаточно быстро. В худшем же случае перебор будет выполнен за время порядка $2^n T(n)$, где $T(n)$ — время, требуемое для вычисления функции E_{K_1} от сообщений длины n. Если сообщения имеют длину порядка 1000 битов, то такой перебор неосуществим на практике ни на каких самых мощных компьютерах.

Мы рассмотрели лишь один из возможных способов атаки на криптосистему и простейший алгоритм поиска открытого текста, называемый обычно алгоритмом полного перебора. Используется также и другое название: «метод грубой силы». Другой простейший алгоритм поиска открытого текста — угадывание. Этот очевидный алгоритм требует небольших вычислений, но срабатывает с пренебрежимо малой вероятностью (при больших длинах текстов). На самом деле противник может пытаться атаковать криптосистему различными способами и использовать различные, более изощрённые алгоритмы поиска открытого текста. Естественно считать криптосистему стойкой, если любой такой алгоритм требует практически неосуществимого объема вычислений или срабатывает с пренебрежимо малой вероятностью. (При этом противник может использовать не только детерминированные, но и вероятностные алгоритмы.) Это и есть теоретико-сложностной подход к определению стойкости. Для его реализации в отношении того или иного типа криптографических схем необходимо выполнить следующее:

1. Дать формальное определение схемы данного типа.
2. Дать формальное определение стойкости схемы.
3. Доказать стойкость конкретной конструкции схемы данного типа.

Здесь сразу же возникает ряд проблем.

Во-первых, в криптографических схемах, разумеется, всегда используются фиксированные значения параметров. Например, криптосистемы разрабатываются для ключей длины, скажем, в 256 или 512 байтов. Для применения же теоретико-сложностного подхода необходимо, чтобы задача, вычислительную сложность которой предполагается использовать, была массовой. Поэтому в теоретической криптографии рассматриваются математические модели криптографических схем. Эти модели зависят от некоторого параметра, называемого параметром безопасности, который может принимать сколь угодно большие значения (обычно для простоты предполагается, что параметр безопасности может пробегать весь натуральный ряд).

Во-вторых, определение стойкости криптографической схемы зависит от той задачи, которая стоит перед противником, и от того, какая информация о схеме ему доступна. Поэтому стойкость схем приходится определять и исследовать отдельно для каждого предположения о противнике.

В-третьих, необходимо уточнить, какой объем вычислений можно считать «практически неосуществимым». Из сказанного выше следует, что эта величина не может быть просто константой, она должна быть представлена функцией от растущего параметра безопасности. В соответствии с тезисом Эдмондса алгоритм считается эффективным, если время его выполнения ограничено некоторым полиномом от длины входного слова (в нашем случае — от параметра безопасности). В противном случае говорят, что вычисления по данному алгоритму практически неосуществимы. Заметим также, что сами криптографические схемы должны быть эффективными, т. е. все вычисления, предписанные той или иной схемой, должны выполняться за полиномиальное время.

В-четвёртых, необходимо определить, какую вероятность можно считать пренебрежимо малой. В криптографии принято считать таковой любую вероятность, которая для любого полинома p и для всех достаточно больших n не превосходит $1/p(n)$, где n — параметр безопасности.

Итак, при наличии всех указанных выше определений, проблема обоснования стойкости криптографической схемы свелась к доказательству отсутствия полиномиального алгоритма, который решает задачу, стоящую перед противником. Но здесь возникает ещё одно и весьма серьёзное препятствие: современное состояние теории сложности вычислений не позволяет доказывать сверхполиномиальные нижние оценки сложности для конкретных задач рассматриваемого класса. Из этого следует, что на данный момент стойкость криптографических схем может быть установлена лишь с привлечением каких-либо недоказанных предположений. Поэтому основное направление исследований состоит в поиске

наиболее слабых достаточных условий (в идеале — необходимых и достаточных) для существования стойких схем каждого из типов. В основном рассматриваются предположения двух типов — общие (или теоретико-сложностные) и теоретико-числовые, т. е. предположения о сложности конкретных теоретико-числовых задач. Все эти предположения в литературе обычно называются криптографическими.

Ниже мы кратко рассмотрим несколько интересных математических объектов, возникших на стыке теории сложности и криптографии. Более подробный обзор по этим вопросам можно найти в книге [1].

2. Криптография и гипотеза P≠NP

Как правило, знакомство математиков-неспециалистов с теорией сложности вычислений ограничивается классами P и NP и знаменитой гипотезой P≠NP.

Напомним вкратце необходимые сведения из теории сложности вычислений. Пусть Σ — множество всех конечных строк в двоичном алфавите. Подмножества $L \subseteq \Sigma$ в теории сложности принято называть языками. Говорят, что машина Тьюринга M работает за полиномиальное время (или просто, что она полиномиальна), если существует полином p такой, что на любом входном слове длины n машина M останавливается после выполнения не более, чем $p(n)$ операций. Машина Тьюринга M распознает (другой термин — принимает) язык L, если на всяком входном слове $x \in L$ машина M останавливается в принимающем состоянии, а на всяком слове $x \notin L$ — в отвергающем. Класс P — это класс всех языков, распознаваемых машинами Тьюринга, работающими за полиномиальное время. Функция $f : \Sigma \to \Sigma$ вычислима за полиномиальное время, если существует полиномиальная машина Тьюринга такая, что если на вход ей подано слово $x \in \Sigma$, то в момент останова на ленте будет записано значение $f(x)$. Язык L принадлежит классу NP, если существуют предикат $P(x, y) : \Sigma \times \Sigma \to \{0, 1\}$, вычислимый за полиномиальное время, и полином p такие, что $L = \{x | \exists y P(x, y) \& |y| \leqslant p(|x|)\}$. Таким образом, язык L принадлежит NP, если для всякого слова из L длины n можно угадать некоторую строку полиномиальной от n длины и затем с помощью предиката P убедиться в правильности догадки. Ясно, что P⊆NP. Является ли это включение строгим — одна из самых известных нерешённых задач математики. Большинство специалистов считают, что оно строгое (так называемая гипотеза P≠NP). В классе NP выделен подкласс максимально сложных языков, называемых NP-полными: любой NP-полный язык распознаваем за полиномиальное время тогда и только тогда, когда P=NP.

Для дальнейшего нам потребуется ещё понятие вероятностной машины Тьюринга. В обычных машинах Тьюринга (их называют детерминированными, чтобы отличить от вероятностных) новое состояние, в которое машина переходит на очередном шаге, полностью определяется текущим состоянием и тем символом, который обозревает головка на ленте. В вероятностных машинах новое состояние может зависеть ещё и от случайной величины, которая принимает

значения 0 и 1 с вероятностью 1/2 каждое. Альтернативно, можно считать, что вероятностная машина Тьюринга имеет дополнительную случайную ленту, на которой записана бесконечная двоичная случайная строка. Случайная лента может читаться только в одном направлении и переход в новое состояние может зависеть от символа, обозреваемого на этой ленте.

Рассмотрим теперь следующий естественный вопрос: не является ли гипотеза P≠NP необходимым и достаточным условием для существования стойких криптографических схем?

Необходимость, и в самом деле, во многих случаях почти очевидна. Вернемся к примеру 1. Определим следующий язык

$L = \{(K_1, d, i) |$ существует сообщение m такое, что $E_{K_1}(m) = d$ и его i-ый бит равен $1\}$.

Ясно, что $L \in$ NP: вместо описанного во введении полного перебора можно просто угадать открытый текст m и проверить за полиномиальное время, что $E_{K_1}(m) = d$ и i-ый бит m равен 1. Если да, то входное слово (K_1, d, i) принимается, в противном случае — отвергается.

В предположении P=NP существует детерминированный полиномиальный алгоритм, распознающий язык L. Зная K_1 и d, с помощью этого алгоритма можно последовательно, по биту, вычислить открытый текст m. Тем самым криптосистема нестойкая.

Тот же подход: угадать секретный ключ и проверить (за полиномиальное время) правильность догадки, применим в принципе и к другим криптографическим схемам. Однако, в некоторых случаях возникают технические трудности, связанные с тем, что по информации, которая имеется у противника, искомая величина (открытый текст, секретный ключ и т. п.) восстанавливается неоднозначно.

Что же касается вопроса о достаточности предположения P≠NP, то здесь напрашивается следующий подход: выбрать какую-либо NP-полную задачу и построить на её основе криптографическую схему, задача взлома которой (т. е. задача, стоящая перед противником) была бы NP-полной. Такие попытки предпринимались в начале 80-х годов, в особенности в отношении криптосистем с открытым ключом, но к успеху не привели. Результатом всех этих попыток стало осознание следующего факта: даже если P≠NP, то любая NP-полная задача может оказаться трудной лишь на некоторой бесконечной последовательности входных слов. Иными словами, в определение класса NP заложена мера сложности «в худшем случае». Для стойкости же криптографической схемы необходимо, чтобы задача противника была сложной «почти всюду». Таким образом, стало ясно, что для криптографической стойкости необходимо существенно более сильное предположение, чем P≠NP. А именно, предположение о существовании односторонних функций.

3. Односторонние функции

Говоря неформально, односторонняя функция — это эффективно вычислимая функция, для задачи инвертирования которой не существует эффективных алгоритмов. Под инвертированием понимается массовая задача нахождения по заданному значению функции одного (любого) значения из прообраза (заметим, что обратная функция, вообще говоря, может не существовать).

Поскольку понятие односторонней функции — центральное в математической криптографии, ниже мы даем его формальное определение.

Пусть $\Sigma^n = \{0,1\}^n$ — множество всех двоичных строк длины n. Под функцией f мы понимаем семейство $\{f_n\}$, где $f_n : \Sigma^n \to \Sigma^m$, $m = m(n)$. Для простоты изложения мы предполагаем, что n пробегает весь натуральный ряд и что каждая из функций f_n всюду определена.

Функция f называется *честной*, если существует полином q такой, что $n \leqslant q(m(n))$ для всех n.

Определение 1. Честная функция f называется *односторонней*, если

1. Существует полиномиальный алгоритм, который для всякого x вычисляет $f(x)$.

2. Для любой полиномиальной вероятностной машины Тьюринга A выполнено следующее. Пусть строка x выбрана наудачу из множества Σ^n (обозначается $x \in_R \Sigma^n$). Тогда для любого полинома p и всех достаточно больших n

$$Pr\{f(A(f(x))) = f(x)\} < 1/p(n).$$

Вероятность здесь определяется случайным выбором строки x и случайными величинами, которые A использует в своей работе.

Условие 2 качественно означает следующее. Любая полиномиальная вероятностная машина Тьюринга A может по данному y найти x из уравнения $f(x) = y$ лишь с пренебрежимо малой вероятностью.

Заметим, что требование честности нельзя опустить. Поскольку длина входного слова $f(x)$ машины A равна m, ей может не хватить полиномиального (от m) времени просто на выписывание строки x, если f слишком сильно «сжимает» входные значения.

Ясно, что из предположения о существовании односторонних функций следует, что P≠NP. Однако, не исключена следующая ситуация: P≠NP, но односторонних функций нет.

Существование односторонних функций является необходимым условием для стойкости многих типов криптографических схем. В некоторых случаях этот факт устанавливается достаточно просто. Обратимся опять

к примеру 1. Рассмотрим функцию f такую, что $f(r) = K_1$. Она вычислима с помощью алгоритма G за полиномиальное время. Покажем, что если f — не односторонняя функция, то криптосистема нестойкая. Предположим, что существует полиномиальный вероятностный алгоритм A, который инвертирует f с вероятностью по крайней мере $1/p(n)$ для некоторого полинома p. Здесь n — длина ключа K_1. Противник может подать на вход алгоритму A ключ K_1 и получить с указанной вероятностью некоторое значение r' из прообраза. Далее противник подаёт r' на вход алгоритма G и получает пару ключей (K_1, K_2'). Хотя K_2' не обязательно совпадает с K_2, тем не менее по определению криптосистемы $D_{K_2'}(E_{K_1}(m)) = m$ для любого открытого текста m. Поскольку K_2' найден с вероятностью по крайней мере $1/p(n)$, которая в криптографии не считается пренебрежимо малой, криптосистема нестойкая.

Для других криптографических схем подобный результат доказывается не столь просто. В работе Импальяццо и Луби [7] доказана необходимость односторонних функций для существования целого ряда стойких криптографических схем.

Из всего сказанного следует, что предположение о существовании односторонних функций является самым слабым криптографическим предположением, которое может оказаться достаточным для доказательства существования стойких криптографических схем различных типов. На выяснение того, является ли это условие и в самом деле достаточным, направлены значительные усилия специалистов. Трудность задачи построения криптографических схем из односторонних функций можно пояснить на следующем примере. Пусть f — односторонняя функция и нам требуется построить *криптосистему с секретным ключом*. В такой криптосистеме имеется только один ключ — секретный, который известен и отправителю, и получателю шифрованного сообщения. Алгоритмы шифрования E_K и дешифрования D_K оба зависят от этого секретного ключа K и таковы, что $D_K(E_K(m)) = m$ для любого открытого текста m. Ясно, что если криптограмму d сообщения m вычислять как $d = f(m)$, то противник, перехвативший d, может вычислить m лишь с пренебрежимо малой вероятностью. Но во-первых, непонятно, каким образом сможет восстановить сообщение m из криптограммы законный получатель? Во-вторых, из того, что функция f односторонняя следует лишь, что противник не может вычислить всё сообщение целиком. А это — весьма низкий уровень стойкости. Желательно, чтобы противник, знающий криптограмму d, не мог вычислить ни одного бита открытого текста.

На настоящий момент доказано, что существование односторонних функций является необходимым и достаточным условием для существования стойких криптосистем с секретным ключом, а также стойких

криптографических протоколов нескольких типов, включая протоколы электронной подписи. С другой стороны, имеется результат Импальяццо и Рудиха [9], который является достаточно сильным аргументом в пользу того, что для некоторых типов криптографических схем (включая протоколы распределения ключей типа Диффи-Хеллмана) требуются более сильные предположения, чем предположение о существовании односторонних функций. К сожалению, этот результат слишком сложный, чтобы его можно было разъяснить в настоящей статье.

4. Псевдослучайные генераторы

Существенный недостаток шифра Вернама состоит в том, что ключи одноразовые. Можно ли избавиться от этого недостатка за счёт некоторого снижения стойкости? Один из способов решения этой проблемы состоит в следующем. Отправитель и получатель имеют общий секретный ключ K длины n и с помощью некоторого достаточно эффективного алгоритма g генерируют из него последовательность $r = g(K)$ длины $q(n)$, где q — некоторый полином. Такая криптосистема (обозначим её Cr) позволяет шифровать сообщение m (или совокупность сообщений) длиной до $q(n)$ битов по формуле $d = r \oplus m$, где $\oplus$ — поразрядное сложение битовых строк по модулю 2. Дешифрование выполняется по формуле $m = d \oplus r$. Из результатов Шеннона вытекает, что такая криптосистема не является абсолютно стойкой, т. е. стойкой против любого противника (в чем, впрочем, нетрудно убедиться и непосредственно). Но что будет, если требуется защищаться только от полиномиально ограниченного противника, который может атаковать криптосистему лишь с помощью полиномиальных вероятностных алгоритмов? Каким условиям должны удовлетворять последовательность r и алгоритм g, чтобы криптосистема Cr была стойкой? Поиски ответов на эти вопросы привели к появлению понятия псевдослучайного генератора, которое было введено Блюмом и Микали [3].

Пусть $g : \{0,1\}^n \to \{0,1\}^{q(n)}$ — функция, вычислимая за полиномиальное (от n) время. Такая функция называется генератором. Интуитивно, генератор g является псевдослучайным, если порождаемые им последовательности неотличимы никаким полиномиальным вероятностным алгоритмом от случайных последовательностей той же длины $q(n)$. Формально этот объект определяется следующим образом.

Пусть A — полиномиальная вероятностная машина Тьюринга, которая получает на входе двоичные строки длины $q(n)$ и выдаёт в результате своей работы один бит. Пусть

$$P_1(A,n) = Pr\{A(r) = 1 | r \in_R \{0,1\}^{q(n)}\}.$$

Вероятность здесь определяется случайным выбором строки r и случайными величинами, которые A использует в своей работе. Пусть

$$P_2(A,n) = Pr\{A(g(s)) = 1 | s \in_R \{0,1\}^n\}.$$

Эта вероятность определяется случайным выбором строки s и случайными величинами, которые A использует в своей работе. Подчеркнём, что функция g вычисляется детерминированным алгоритмом.

Определение 2. Генератор g называется *криптографически стойким псевдослучайным генератором*, если для любой полиномиальной вероятностной машины Тьюринга A, для любого полинома p и всех достаточно больших n

$$|P_1(A,n) - P_2(A,n)| < 1/p(n).$$

Всюду ниже мы для краткости будем называть криптографически стойкие псевдослучайные генераторы просто псевдослучайными генераторами. Такое сокращение является общепринятым в криптографической литературе.

Нетрудно убедиться, что для существования псевдослучайных генераторов необходимо существование односторонних функций. В самом деле, сама функция g должна быть односторонней. Доказательство этого простого факта мы оставляем читателю в качестве упражнения. Вопрос о том, является ли существование односторонних функций одновременно и достаточным условием, долгое время оставался открытым. В 1982 г. Яо [10] построил псевдослучайный генератор, исходя из предположения о существовании *односторонних перестановок*, т. е. сохраняющих длину взаимнооднозначных односторонних функций. За этим последовала серия работ, в которых достаточное условие всё более и более ослаблялось, пока наконец в 1989–1990 гг. Импальяццо, Левин и Луби [8] и Хостад [6] не получили следующий окончательный результат.

Теорема 1. *Псевдослучайные генераторы существуют тогда и только тогда, когда существуют односторонние функции.*

Псевдослучайные генераторы находят применение не только в криптографии, но и в теории сложности, и в других областях дискретной математики. Обсуждение этих приложений выходит за рамки настоящей статьи. Здесь же в качестве иллюстрации мы рассмотрим описанную в начале данного раздела криптосистему Cr, использующую псевдослучайный генератор в качестве алгоритма g. Прежде всего, нам необходимо дать определение стойкости криптосистемы с секретным ключом.

Пусть E_K — алгоритм шифрования криптосистемы с секретным ключом. Обозначим результат его работы $d = E_K(m)$, здесь K — секретный

ключ длиной n битов, а m — открытый текст длиной $q(n)$ битов. Через m_i обозначается i-ый бит открытого текста. Пусть A — полиномиальная вероятностная машина Тьюринга, которая получает на вход криптограмму d и выдаёт пару (i, σ), где $i \in \{1, \ldots, q(n)\}$, $\sigma \in \{0, 1\}$. Интуитивно, криптосистема является стойкой, если никакая машина Тьюринга A не может вычислить ни один бит открытого текста с вероятностью успеха, существенно большей, чем при простом угадывании.

Определение 3. Криптосистема называется стойкой, если для любой полиномиальной вероятностной машины Тьюринга A, для любого полинома p и всех достаточно больших n

$$Pr\{A(c) = (i, \sigma) \ \& \ \sigma = m_i \mid K \in_R \{0,1\}^n, \ m \in_R \{0,1\}^{q(n)}\} < 1/2 + 1/p(n).$$

Эта вероятность (всюду ниже для краткости мы её обозначаем просто Pr) определяется случайным выбором секретного ключа K, случайным выбором открытого текста m из множества всех двоичных строк длины $q(n)$ и случайными величинами, которые A использует в своей работе.

Покажем, что криптосистема Cr с псевдослучайным генератором в качестве g является стойкой в смысле данного определения. Предположим противное, т. е. что существуют полиномиальный вероятностный алгоритм A и полином p такие, что $Pr \geqslant 1/2 + 1/p(n)$ для бесконечно многих n. Рассмотрим алгоритм B, который получает на входе двоичную строку r длины $q(n)$, выбирает $m \in_R \{0,1\}^{q(n)}$, вычисляет $d = m \oplus r$ и вызывает A как подпрограмму, подавая ей на вход строку d. Получив от A пару (i, σ), B проверяет, действительно ли $m_i = \sigma$ и если да, то выдаёт 1, в противном случае — 0, и останавливается. Легко видеть, что B работает за полиномиальное (от n) время. Убедимся, что алгоритм B отличает псевдослучайные строки, порождённые генератором g, от случайных строк длины $q(n)$. В самом деле, если строки r, поступающие на вход B, являются случайными, то d — криптограмма шифра Вернама и, согласно теореме Шеннона, $Pr = 1/2$. Если строки r порождены генератором g, то криптограммы d имеют такое же распределение вероятностей, как в криптосистеме Cr, и, согласно предположению, $Pr \geqslant 1/2 + 1/p(n)$ для бесконечно многих n. Полученное противоречие с определением псевдослучайного генератора доказывает утверждение о стойкости криптосистемы Cr.

Разумеется, стойкость криптосистемы с секретным ключом можно определять различным образом. Например, можно рассматривать стойкость против атаки с выбором открытого текста: противник может предварительно выбрать полиномиальное количество открытых текстов и получить их криптограммы, после чего он получает ту криптограмму, по

которой ему требуется вычислить хотя бы один бит соответствующего открытого текста. Нетрудно убедиться, что криптосистема Cr с псевдослучайным генератором в качестве g является стойкой и против атаки с выбором открытого текста.

Таким образом, мы убедились, что с помощью псевдослучайных генераторов можно строить стойкие криптосистемы. Основное направление исследований в данной области — поиск методов построения эффективных псевдослучайных генераторов на основе различных криптографических предположений. Показателем эффективности здесь служит количество операций, затрачиваемых на вычисление каждого очередного бита псевдослучайной последовательности.

5. Доказательства с нулевым разглашением

Предположим, что Алиса знает доказательство некоторой теоремы и желает убедить Боба в том, что теорема верна. Конечно, Алиса может просто передать доказательство Бобу на проверку. Но тогда впоследствии Боб сможет сам, без помощи Алисы, доказывать третьим лицам эту теорему. А может ли Алиса убедить Боба так, чтобы он не получил при этом никакой информации, которая помогла бы ему восстановить доказательство теоремы? Этим двум, казалось бы взаимно исключающим требованиям, удовлетворяют протоколы доказательства с нулевым разглашением. Последнее понятие было введено Гольдвассер, Микали и Ракоффом в 1985 г. [4].

Рассматривается следующая модель протокола. В распоряжении Алисы и Боба имеются вероятностные машины Тьюринга $\mathbf{P}$ и $\mathbf{V}$ соответственно. Вычислительные ресурсы, которые может использовать Алиса, неограничены, в то время как машина $\mathbf{V}$ работает за полиномиальное время. Машины $\mathbf{P}$ и $\mathbf{V}$ имеют общую коммуникационную ленту для обмена сообщениями. После записи сообщения на коммуникационную ленту машина переходит в состояние ожидания и выходит из него, как только на ленту будет записано ответное сообщение. Машины $\mathbf{P}$ и $\mathbf{V}$ имеют также общую входную ленту, на которую записано входное слово x. Утверждение, которое доказывает Алиса, суть «$x \in L$», где L — некоторый фиксированный (известный и Алисе, и Бобу) язык. Чтобы избежать тривиальности, язык L должен быть трудным (например, NP-полным), иначе Боб сможет самостоятельно проверить, что $x \in L$. По существу, протокол доказательства состоит в том, что Боб, используя случайность, выбирает некоторые вопросы, задаёт их Алисе и проверяет правильность ответов. Выполнение протокола завершается, когда машина $\mathbf{V}$ останавливается,

при этом она выдаёт 1, если доказательство принято, и 0 — в противном случае.

Пусть A и B — две интерактивные, т. е. взаимодействующие через общую коммуникационную ленту, вероятностные машины Тьюринга. Через $[B(x), A(x)]$ обозначается случайная величина — выходное слово машины A, когда A и B работают на входном слове x. Через $|x|$ обозначается длина слова x.

Определение 4. *Интерактивным доказательством для языка* L называется пара интерактивных машин Тьюринга $(\mathbf{P}, \mathbf{V})$ такая, что выполняются следующие два условия.

1. (Полнота). Для всех $x \in L$

$$Pr\{[\mathbf{P}(x), \mathbf{V}(x)] = 1\} = 1.$$

2. (Корректность). Для любой машины Тьюринга $\mathbf{P}^*$, для любого полинома p и для всех $x \notin L$ достаточно большой длины

$$Pr\{[\mathbf{P}^*(x), \mathbf{V}(x)] = 1\} < 1/p(|x|).$$

Полнота означает, что если входное слово принадлежит языку L и оба участника, и Алиса, и Боб, следуют протоколу, то доказательство будет всегда принято. Требование корректности защищает Боба от нечестной Алисы, которая пытается обмануть его, «доказывая» ложное утверждение. При этом Алиса может каким угодно образом отклоняться от действий, предписанных протоколом, т. е. вместо машины Тьюринга $\mathbf{P}$ использовать любую другую машину $\mathbf{P}^*$. Требуется, чтобы вероятность обмана была в любом случае пренебрежимо малой.

Определение 5. Интерактивный протокол доказательства для языка L называется *доказательством с абсолютно нулевым разглашением*, если, кроме условий 1 и 2, выполнено ещё и следующее условие.

3. (Свойство нулевого разглашения). Для любой полиномиальной вероятностной машины Тьюринга $\mathbf{V}^*$ существует вероятностная машина Тьюринга $\mathbf{M}_{\mathbf{V}^*}$, работающая за полиномиальное в среднем время, и такая, что для всех $x \in L$

$$\mathbf{M}_{\mathbf{V}^*}(x) = [\mathbf{P}(x), \mathbf{V}^*(x)].$$

Машина $\mathbf{M}_{\mathbf{V}^*}$ называется моделирующей машиной для $\mathbf{V}^*$. Предполагается, что математическое ожидание времени её работы ограничено полиномом от длины x. Это означает, что в принципе $\mathbf{M}_{\mathbf{V}^*}$ может, в

зависимости от того, какие значения примут используемые в её работе случайные переменные, работать достаточно долго. Но вероятность того, что время её работы превысит некоторую полиномиальную границу, мала. Для каждой машины $\mathbf{V}^*$ строится своя моделирующая машина; последняя может использовать $\mathbf{V}^*$ как подпрограмму. Через $\mathbf{M}_{\mathbf{V}^*}(x)$ обозначается случайная величина — выходное слово машины $\mathbf{M}_{\mathbf{V}^*}$, когда на входе она получает слово x.

Свойство нулевого разглашения защищает Алису от нечестного Боба, который, произвольно отклоняясь от действий, предписанных протоколом (используя $\mathbf{V}^*$ вместо $\mathbf{V}$), пытается извлечь из его выполнения дополнительную информацию. Условие 3 означает, что Боб может при этом получить только такую информацию, которую он смог бы вычислить и самостоятельно (без выполнения протокола) за полиномиальное время.

Приведем в качестве примера протокол доказательства с абсолютно нулевым разглашением для языка ИЗОМОРФИЗМ ГРАФОВ из работы Гольдрайха, Микали и Вигдерсона [5]. Входным словом является пара графов $G_1 = (U, E_1)$ и $G_0 = (U, E_0)$. Здесь U — множество вершин, которое можно отождествить с множеством натуральных чисел $\{1, \ldots, n\}$, E_1 и E_0 — множества рёбер такие, что $|E_1| = |E_0| = m$. Графы G_1 и G_0 называются изоморфными, если существует перестановка φ на множестве U такая, что $(u, v) \in E_0$ тогда и только тогда, когда $(\varphi(u), \varphi(v)) \in E_1$ (обозначается $G_1 = \varphi G_0$). Задача распознавания изоморфизма графов — хорошо известная математическая задача, для которой на данный момент не известно полиномиальных алгоритмов. С другой стороны, неизвестно, является ли эта задача NP-полной, хотя есть веские основания предполагать, что не является.

Протокол IG

Пусть φ — изоморфизм между G_1 и G_0. Следующие четыре шага выполняются в цикле m раз, каждый раз с независимыми случайными величинами.

1. $\mathbf{P}$ *выбирает случайную перестановку π на множестве U, вычисляет $H = \pi G_1$ и посылает этот граф* $\mathbf{V}$.

2. $\mathbf{V}$ *выбирает случайный бит α и посылает его* $\mathbf{P}$.

3. Если $\alpha = 1$, то $\mathbf{P}$ *посылает* $\mathbf{V}$ *перестановку π, в противном случае — перестановку $\pi \circ \varphi$.*

4. Если перестановка, полученная $\mathbf{V}$*, не является изоморфизмом между G_α и H, то* $\mathbf{V}$ *останавливается и отвергает доказательство. В противном случае выполнение протокола продолжается.*

Если проверки п.4 дали положительный результат во всех m циклах, то $\mathbf{V}$ *принимает доказательство.*

Заметим, что если в протоколе IG машина $\mathbf{P}$ получает изоморфизм φ в качестве дополнительного входного слова, то ей для выполнения протокола не требуются неограниченные вычислительные ресурсы. Более того, в этом случае $\mathbf{P}$ может быть полиномиальной вероятностной машиной Тьюринга.

Теорема 2 ([5]). *Протокол IG является доказательством с абсолютно нулевым разглашением для языка ИЗОМОРФИЗМ ГРАФОВ.*

Полнота протокола IG очевидна.

Для доказательства корректности достаточно заметить, что бит α, который $\mathbf{V}$ выбирает на шаге 2, указывает $\mathbf{P}$, для какого из графов — G_0 или G_1 — требуется продемонстрировать изоморфизм с графом H. Если G_0 и G_1 не изоморфны, то H может быть изоморфен, в лучшем случае, одному из них. Поэтому проверка п. 4 даст положительный результат с вероятностью $\leqslant 1/2$ в одном цикле и с вероятностью $\leqslant 1/2^m$ во всех m циклах.

Доказательство свойства нулевого разглашения значительно сложнее. Поэтому мы воспроизводим только основную идею. Прежде всего, заметим, что основная задача машины $\mathbf{V}^*$ — получить максимально возможную информацию об изоморфизме между G_0 и G_1. Естественно предположить, что она, в отличие от $\mathbf{V}$, будет выдавать в качестве выходного слова не один бит, а всю полученную в результате выполнения протокола информацию, включая графы H и перестановки, полученные соответственно на шагах 1 и 3 протокола IG. Моделирующая машина $\mathbf{M}_{\mathbf{V}^*}$ должна уметь строить такие же графы и перестановки, не зная при этом изоморфизм φ! Поэтому $\mathbf{M}_{\mathbf{V}^*}$ пытается угадать тот бит α, который будет запросом машины $\mathbf{V}^*$ на шаге 2. Для этого $\mathbf{M}_{\mathbf{V}^*}$ выбирает случайный бит β, случайную перестановку ψ и вычисляет $H = \psi G_\beta$. Далее $\mathbf{M}_{\mathbf{V}^*}$ запоминает состояние машины $\mathbf{V}^*$ и вызывает её как подпрограмму, подавая ей на вход граф H. Ответом машины $\mathbf{V}^*$ будет некоторый бит α. Если $\alpha = \beta$, то моделирование в данном цикле завершено успешно, поскольку $\mathbf{M}_{\mathbf{V}^*}$ может продемонстрировать требуемый изоморфизм. Если же $\alpha \neq \beta$, то $\mathbf{M}_{\mathbf{V}^*}$ восстанавливает ранее сохранённое состояние машины $\mathbf{V}^*$ и повторяет попытку.

Если в определении свойства нулевого разглашения заменить равенство случайных величин $\mathbf{M}_{\mathbf{V}^*}(x)$ и $[\mathbf{P}(x), \mathbf{V}^*(x)]$ требованием, чтобы их распределения вероятностей «почти не отличались», то получится дру-

гая разновидность доказательств — *доказательства со статистически нулевым разглашением.*

Еще один тип — *доказательства с вычислительно нулевым разглашением.* В этом случае требуется, чтобы моделирующая машина создавала распределение вероятностей, которое неотличимо от $[\mathbf{P}(x), \mathbf{V}^*(x)]$ никаким полиномиальным вероятностным алгоритмом (неотличимость здесь определяется аналогично тому, как это делалось в определении псевдослучайного генератора).

Подчеркнём особо, что во всех трёх определениях нулевого разглашения условия накладываются на действия моделирующей машины только на тех словах, которые принадлежат языку.

Помимо интереса к доказательствам с нулевым разглашением как к нетривиальному математическому объекту, они исследуются также и в связи с практическими приложениями. Наиболее естественный и важный тип таких приложений — протоколы аутентификации. С помощью такого протокола Алиса может доказать Бобу свою аутентичность. Предположим, например, что Алиса — это интеллектуальная банковская карточка, в которой реализован алгоритм $\mathbf{P}$, а Боб — это компьютер банка, выполняющий программу $\mathbf{V}$. Прежде чем начать выполнение каких-либо банковских операций, банк должен убедиться в подлинности карточки и идентифицировать её владельца, или, говоря на языке криптографии, карточка должна пройти аутентификацию. В принципе для этой цели можно использовать приведенный выше протокол IG. В этом случае в памяти банковского компьютера хранится пара графов (G_0, G_1), сопоставленная Алисе, а на интеллектуальной карточке — та же пара графов и изоморфизм φ. Предполагается, что, кроме Алисы, этот изоморфизм никто не знает (кроме, быть может, Боба) и поэтому с помощью протокола IG карточка доказывает свою аутентичность. При этом свойство полноты означает, что карточка наверняка докажет свою аутентичность. Свойство корректности защищает интересы банка от злоумышленника, который, не являясь клиентом банка, пытается пройти аутентификацию, используя фальшивую карточку. Свойство нулевого разглашения защищает клиента от злоумышленника, который, подслушав одно или более выполнений протокола аутентификации данной карточки, пытается пройти аутентификацию под именем Алисы. Для практического применения очень важным свойством протокола IG является то, что алгоритм $\mathbf{P}$, получивший в качестве дополнительного входа изоморфизм φ, работает за полиномиальное время. Вместо протокола IG можно использовать, вообще говоря, любое другое доказательство с нулевым разглашением, в котором алгоритм $\mathbf{P}$ обладает этим свойством. Но для реальных приложений протокол IG, как и большинство подобных протоколов, не

эффективен: большое количество циклов, слишком длинные сообщения и т. д. Поиск более эффективных доказуемо стойких протоколов — одно из основных направлений исследований в данной области.

Список литературы

[1] *Анохин М. И., Варновский Н. П., Сидельников В. М., Ященко В. В.* Криптография в банковском деле. М.: МИФИ, 1997.

[2] *Гэри М., Джонсон Д.* Вычислительные машины и трудно решаемые задачи. М.: Мир, 1982.

[3] *Blum M., Micali S.* How to generate crytographically strong sequences of pseudo-random bits // SIAM J. Comput. **V. 13**, No 4, 1984. P. 850–864.

[4] *Goldwasser S., Micali S., Rackoff C.* The knowledge complexity of interactive proof systems // SIAM J. Comput. **V. 18**, No 1, 1989. P. 186–208.

[5] *Goldreich O., Micali S., Wigderson A.* Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems // J. ACM. **V. 38**, No 3, 1991. P. 691–729.

[6] *Håstad J.* Pseudo-random generators under uniform assumptions // Proc. 22nd Annu. ACM Symp. on Theory of Computing. 1990. P. 395–404.

[7] *Impagliazzo R., Luby M.* One-way functions are essential for complexity based cryptography // Proc. 30th Annu. Symp. on Found. of Comput. Sci. 1989. P. 230–235.

[8] *Impagliazzo R. , Levin L., Luby M.* Pseudo-random generation from one-way functions // Proc. 21st Annu. ACM Symp. on Theory of Computing. 1989. P. 12–24.

[9] *Impagliazzo R., Rudich S.* Limits on the provable consequences of one-way permutations // Proc. 21st Annu. ACM Symp. on Theory of Computing. 1989. P. 44–61.

[10] *Yao A.C.* Theory and applications of trapdoor functions // Proc. 23rd Annu. Symp. on Found. of Comput. Sci. 1982. P. 80–91.