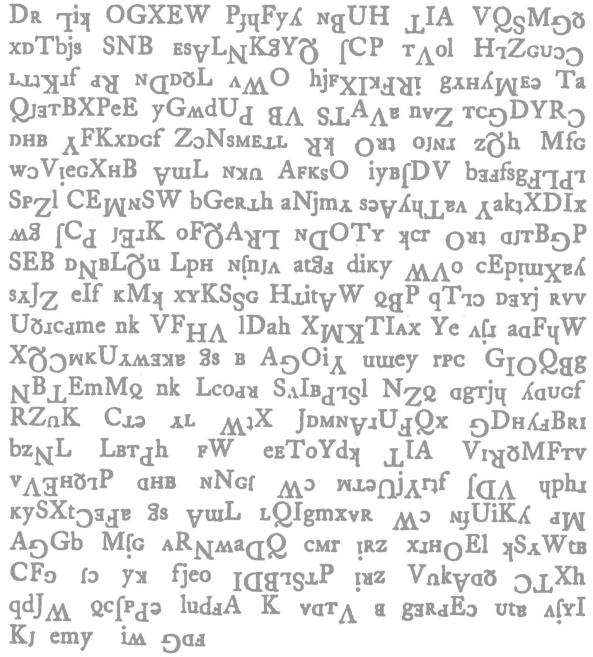
С. В. Агиевич

Криптографические методы



Версия: 2014.07.11

ОГЛАВЛЕНИЕ

Глава 1
ВВЕДЕНИЕ

Проблема защиты информации имеет давнюю историю. Везде с появлением письменности появлялась и «тайнопись» — способ преобразования информации в секретную форму, понимаемую только доверенными лицами. От греческого «κρυπτοσ λογοσ» (тайное слово) и произошло название современной науки о защите информации — криптологии.

Криптология условно делится на две дисциплины: криптография — защита, и криптоанализ — атака, нападение. Криптографы разрабатывают надежные (на их взгляд) системы зашиты информации, криптоаналитики пытаются найти в этих системах уязвимости. Криптографы могут гордиться тем, что атаки на действующие криптографические алгоритмы и протоколы становятся все более редкими. Криптоаналитики знают, что атаки не могут стать хуже и со временем только разовьются.

Криптология опирается на математику. Известно, что известный английский математик Харди, работавший в начале XX века, гордился тем, что его наука — теория чисел — никогда не будет использоваться в сколь либо практических целях. Харди с удивлением узнал бы, что в современных криптографических системах используется теорема Эйлера, простые Ферма, закон квадратичной взаимности Гаусса, спаривания Вейля и другие объекты и свойства, казалось бы, чистой математики. Современная криптографическая система — это своего рода изобретение, которое решает задачи взаимодействия между людьми в «цифровом мире». Обычные изобретения строятся на законах физики, но в цифровом мире не остается ничего другого, как использовать законы математики.

Настоящий документ представляет собой выборочные главы курса лекций «Криптографические методы», которые автор читает на Факультете прикладной математики и информатики Белорусского государственного университета. Главы вошли в недавно изданный учебник «Криптология». [1]

Планируется, что документ будет корректироваться и расширяться. Поэтому титульная страница снабжена номером версии (в формате YYYY.MM.DD). Следует отдавать предпочтение документу самой последней доступной версии.

[1] Харин Ю. С., Агиевич С. В., Васильев Д. В., Матвеев Г. В. Криптология. — Минск: БГУ, 2013.

Глава 2
ЭЛЕМЕНТЫ ТЕОРИИ СЛОЖНОСТИ ВЫЧИСЛЕНИЙ

2.1. Вычислительные задачи

Пусть, как обычно, $\{0,1\}^*$ — множество всех слов конечной длины в двоичном алфавите. Для слова $x \in \{0,1\}^*$ через $|x|$ обозначаем его длину, $\{0,1\}^l$ — множество слов длины l, $\perp$ — пустое слово (длины 0), α^l — слово, составленное из l экземпляров символа $\alpha \in \{0,1\}$.

Под *вычислительной задачей* понимают задачу вычисления значений функции $f\colon \{0,1\}^* \to \{0,1\}^*$. Аргумент x называют *входными данными* или просто *входом* задачи, значение $f(x)$ — *выходными данными*, *выходом* или *ответом*.

Будем считать, что входы и выходы отличаются от пустого слова $\perp$. Это слово зарезервируем для описания различных ошибочных ситуаций при вычислениях.

Входы и выходы задачи могут представлять различные объекты: числа, многочлены, векторы, графы, функции и др. Входными данными интересующих нас задач, как правило, являются наборы натуральных чисел. Эти числа будем представлять их двоичной записью: слово $a_1 a_2 \dots a_l$ кодирует число $a = \sum_{i=1}^{l} a_i 2^{l-i}$. Длину l кодового представления обычно будем выбирать минимально возможной: $l = \lfloor \log_2 a \rfloor + 1$.

Если x представляет только a, то длина $l = |x|$ известна и a легко восстанавливается по x. Если же a является только частью x, то для восстановления a можно включить в x кодовое представление l. Например, можно закодировать число l серией из $l-1$ нулей, и тогда получится гамма-код Элиаса. Это двоичное слово из не более чем $2\lfloor \log_2 a \rfloor + 1$ символов, которое однозначно представляет a. Существуют и другие способы кодирования натуральных чисел. Будем использовать те, которые дают кодовые слова длины $O(\log a)$.

Код целого числа b — это код его знака, дополненный в случае $b \neq 0$ кодом натурального $|b|$. Код рационального числа a/b — это объединение кодов a и b. Понятно, что можно составить разумные правила кодирования произвольных структурированных наборов данных. Интересно, что одни из таких правил — так называемая абстрактно-синтаксическая нотация версии 1 (ACH.1) — часто используются в криптографии для описания ключей, долговременных параметров, сообщений протоколов и пр.

Область определения функции f может включать не все двоичные слова. Другими словами, f может быть *частичной* (а не полной) функцией на $\{0,1\}^*$. Исключение определенных входов связано со спецификой задачи и особенностями кодирования входных данных. Будем по возможности расширять область определения f до $\{0,1\}^*$, устанавливая, например, что все недопустимые входы представляют один и тот же фиксированный допустимый вход. Тем не менее, в необходимых случаях будем подчеркивать, что f является частичной функцией.

2.2. Задачи распознавания и поиска

В криптографии работают с вычислительными задачами двух типов: *распознавания* и *поиска*.

В задаче распознавания функция f может принимать только два значения: 1 (да) или 0 (нет). Такую функцию называют *предикатом*. С предикатом удобно отождествить

язык (множество слов)

$$L = f^{-1}(1) = \{x \in \{0,1\}^* \colon f(x) = 1\}$$

и вести речь о проверке принадлежности входа x языку L, т. е. о распознавании L.

Известной задачей распознавания является задача проверки простоты числа. Язык этой задачи:

$$PRIMES = \{p \in \mathbb{N} \colon p \text{ — простое}\}.$$

Здесь и далее мы предполагаем, что натуральные числа и другие объекты неявно кодируются двоичными словами.

В задаче поиска описывается множество $R \subseteq \{0,1\}^* \times \{0,1\}^*$. Входу x соответствует не один, а целое множество выходов — это элементы множества $R(x) = \{y \in \{0,1\}^* \colon (x,y) \in R\}$. Функция f задается неявно. Говорят, что f *решает* R, если $f(x) \in R(x)$. Множество $R(x)$ может оказаться пустым, и тогда f не определена в точке x, т. е. f, вообще говоря, является частичной функцией.

Следующие задачи поиска часто используются в криптографии.

Задача факторизации (`Factor`). Входом `Factor` является составное натуральное n, выходом — нетривиальный (т. е. отличный от 1 и n) делитель d числа n. Другими словами, $\texttt{Factor}(n) = \{d \in \mathbb{N} \colon 1 < d < n,\ d \mid n\}$.

Задача дискретного логарифмирования (`DL`). Пусть имеется семейство конечных циклических групп и пусть G — представитель этого семейства. Пусть G описывается словом $\mathrm{descr}(G)$, которое определяет порядок $q = |G|$, строение элементов G и групповую операцию над ними. В криптографии используются два основных способа выбора (или, как еще говорят, *инстанциирования*) группы G. Во-первых, в качестве G может быть выбрана подгруппа мультипликативной группы простого поля $\mathbb{F}_p$: $G \subseteq \mathbb{F}_p^*$, $q \mid p-1$. В этом случае для полного описания G достаточно использовать пару (p,q). Во-вторых, G может быть группой точек эллиптической кривой над конечным полем.

Входными данными `DL` является тройка $(\mathrm{descr}(G), g, g^a)$, где g — образующий G, $a \in \{0,1,\ldots,q-1\}$. Фигурирующее здесь число a называется *дискретным логарифмом* g^a по основанию g и является выходом `DL`.

Задача Диффи — Хеллмана (`CDH`). В задаче `CDH` также используется циклическая группа G порядка q, ее образующий g и произвольный элемент g^a. Используется дополнительный элемент g^b, $b \in \{0,1,\ldots,q-1\}$. Требуется по $(\mathrm{descr}(G), g, g^a, g^b)$ найти g^{ab}.

На самом деле существует целое семейство задач типа Диффи — Хеллмана. Мы определили только одну из них — так называемую вычислительную (computational). Отсюда буква `C` в названии задачи.

На связанных между собой задачах `DL` и `CDH` базируются два больших направления в криптографии: FFC (Finite Field Cryptography, случай $G \subset \mathbb{F}_p^*$) и ECC (Elliptic Curve Cryptography, выбор в качестве G группы точек эллиптической кривой). На задаче `Factor` базируется направление IFC (Integer Factorization Cryptography).

2.3. Машина Тьюринга

Для решения задач служат алгоритмы. Неформально говоря, алгоритм — это однозначно определенная последовательность инструкций по преобразованию входных данных в выходные.

Приведенное определение не является математически строгим (что такое инструкция? что значит однозначно определенная?). Для полной формализации используются модели вычислительных устройств, реализующих алгоритмы. Наиболее известная модель — *машина Тьюринга* (МТ), предложенная английским математиком А. Тьюрингом

в 1936 г. Согласно общепризнанному тезису Чёрча — Тьюринга любой алгоритм в интуитивном смысле этого слова может быть реализован некоторой машиной Тьюринга. Другими словами, принято отождествлять алгоритмы и МТ.

Машина Тьюринга M характеризуется следующими элементами:

1. *Лента.* Лента представляет собой набор ячеек, пронумерованных $1, 2, \ldots$. Число ячеек не ограничено. В каждой ячейке хранится символ алфавита $\Sigma = \{0, 1, \text{␣}\}$. Символы ячеек ленты образуют бесконечное вправо слово $a = a_1 a_2 \ldots$.
2. *Управляющее устройство.* Управляющее устройство выполняет манипуляции над символами ленты в зависимости от своего состояния $q \in Q$, где Q — конечное множество. Состояния меняются по правилам, заданным *функцией переходов* $\varphi\colon Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$. Определено начальное состояние q_0.
3. *Головка управляющего устройства.* Головка передвигается по ленте и в конкретный момент времени находится над ячейкой с номером $d \in \mathbb{N}$. Управляющее устройство может читать символ ячейки под головкой и менять только его. Все организовано так, что вычисления меняют физическую среду (ленту) локально. Это основной тезис интуитивных представлений о вычислениях.

Состояние всей машины задается тройкой (a, q, d). Управляющее устройство меняет состояния, выполняя следующие такты вычислений:

1) прочитывается символ a_d, находящийся под головкой;
2) вычисляется тройка $(q', a'_d, \Delta d) \leftarrow \varphi(q, a_d)$;
3) символ a'_d записывается в ячейку с номером d (вместо a_d);
4) состояние управляющего устройства меняется на q';
5) головка сдвигается на Δd позиций;
6) если $d + \Delta d = 0$, то машина останавливается.

В начале работы M в первые ячейки ленты записываются входные данные $x \in \{0, 1\}^*$, а остальные ячейки заполняются символом ␣, т. е. первоначально $a = x\text{␣␣}\ldots$. Управляющее устройство начинает работу в состоянии q_0, а головка устанавливается над первой ячейкой. После остановки машины слово, записанное на ленте, имеет вид $y\text{␣}\ldots$, где $y \in \{0, 1\}^*$. Слово-префикс y объявляется выходными данными M на входе x: $y = M(x)$.

Для работы машины M может потребоваться бесконечно много ячеек на ленте. С другой стороны, для исчерпывающего описания M достаточно указать конечные объекты: множество Q, функцию переходов φ и начальное состояние s_0. Таким образом, описание M можно закодировать двоичным словом $[M]$ конечной длины.

К сожалению, МТ является примитивным устройством и код его описания весьма неудобен. Далее мы будем использовать высокоуровневый способ кодирования, включающий стандартные алгоритмические конструкции типа `if` – `then` – `else`, `while`, `for`. При некоторых ограничениях программы на таких языках могут быть преобразованы к описанию $[M]$ с помощью специальной МТ (компилятора).

Будем задавать алгоритмические конструкции на русском языке, не придерживаясь жестких синтаксических ограничений. В наших описаниях алгоритмов обязательно будут рубрики «Вход», «Выход», «Шаги». Например, алгоритм Евклида определяется следующим образом.

Алгоритм Евклида

Вход: натуральные a и b, $a \geqslant b$.

Выход: $\gcd(a, b)$.

Шаги:

1. Пока $b \neq 0$ выполнить:
   (1) $a \leftarrow a \mod b$;

(2) $a \leftrightarrow b$.
2. Возвратить a.

2.4. Разрешимые и неразрешимые задачи

Пусть M останавливается на входе x с результатом y. Результат определен однозначно и, таким образом, M определяет функцию $f_M\colon x \mapsto y$. Функция f_M является частичной, ее значения не определены для тех входов x, на которых M не останавливается.

Определение 2.1. *Частичная функция $f\colon \{0,1\}^* \to \{0,1\}^*$ вычислима (задача f разрешима), если найдется машина Тьюринга M такая, что $f = f_M$ (совпадают и области определения, и значения функций). Говорят, что M вычисляет функцию f (решает задачу f).*

Для задач поиска последняя часть определения уточняется: машина M решает задачу поиска R, если f_M решает R.

Теорема 2.1. *Существуют неразрешимые задачи (невычислимые функции).*

Доказательство. Функций $f\colon \{0,1\}^* \to \{0,1\}^*$ больше, чем описаний машин Тьюринга $[M] \in \{0,1\}^*$. □

Возможно самый известный пример неразрешимой задачи — это *задача об остановке*, предложенная самим Тьюрингом. Входными данными задачи является описание $[M]$ машины M и слово $x \in \{0,1\}^*$. Выход: 1, если M остановится на входе x, и 0 в противном случае.

Теорема 2.2 (Тьюринг). *Задача об остановке неразрешима.*

Доказательство. Пусть h — предикат задачи об остановке: $h([M], x) = 1$ тогда и только тогда, когда M остановится на входе x. Предположим, от противного, что предикат h вычислим на некоторой машине A.

Построим машину B, которая обрабатывает вход $[M]$ следующим образом: если $A([M],[M]) = 1$, то B переходит в бесконечный цикл; в противном случае B возвращает 0. Описание B незначительно расширяет описание A: добавляется проверка выхода и, при необходимости, выполняется переход в бесконечный цикл.

Подадим на вход B ее описание $[B]$. Если B остановится, то $A([B],[B]) = 0$ и остановки быть не должно по определению h. Если B не остановится, то $A([B],[B]) = 1$ и, наоборот, остановка должна быть. Найденные противоречия доказывают требуемый результат. □

Еще одна знаменитая неразрешимая задача — это десятая проблема Гильберта. Входными данными здесь является многочлен от нескольких переменных с целыми коэффициентами (например, $x^n + y^n - z^n$, $n \in \mathbb{N}$). Требуется определить, имеет многочлен целочисленные корни или нет. Неразрешимость десятой проблемы Гильберта доказал в 1971 г. советский математик Ю. Матиясевич.

2.5. Ресурсы

Для решения задач требуются ресурсы. Будем рассматривать два типа ресурсов: *время* и *память*. Нас будет интересовать их пиковое потребление. Пусть x — допустимый вход машины M в том смысле, что M останавливается на этом входе. Через $t_M(x)$ обозначим число тактов, которое выполнит M при вычислениях на входе x, а через $s_M(x)$ — крайнюю правую позицию головки при вычислениях.

Определение 2.2. *Машина Тьюринга M работает за время $T_M(l)$, $l \in \mathbb{N}$, если $T_M(l) = \max_x t_M(x)$, где максимум берется по допустимым $x \in \{0,1\}^l$. Аналогично, M работает на памяти $S_M(l)$, если $S_M(l) = \max_x s_M(x)$.*

Ясно, что $S_M(l) \leqslant T_M(l)$ для любой M. Поэтому время работы является более универсальной характеристикой сложности, чем память.

Будем рассматривать характеристики сложности в асимптотике $l \to \infty$. Напомним обозначения, которые касаются сравнения скорости роста функций:
- $u(l) = o(v(l))$, если $u(l)/f(l) \to 0$;
- $u(l) = O(v(l))$, если $|u(l)| \leqslant Cv(l)$ для некоторой константы C и всех достаточно больших l;
- $u(l) = \Omega(v(l))$, если $v(l) = O(u(l))$;
- $u(l) = \Theta(v(l))$, если $u(l) = O(v(l))$ и $v(l) = O(u(l))$.

Говорят, что M работает за *полиномиальное* время или является *полиномиальной* машиной Тьюринга (ПМТ), если найдется $c \in \mathbb{N}$ такое, что $T_M(l) \leqslant l^c$ для всех достаточно больших l. Последнее условие можно записать по-другому: $T_M(l) = l^{O(1)}$. Среди ПМТ выделяют машины, которые работают за *линейное* время: $T_M(l) = \Theta(l)$, за *квазилинейное*: $T_M(l) = l(\log l)^{O(1)}$, за *квадратичное*: $T_M(l) = \Theta(l^2)$, за *кубическое*: $T_M(l) = \Theta(l^3)$ и т. д. Согласно распространенному тезису Кобхэма, полиномиальные алгоритмы и только они являются *эффективными*, т. е. имеющими практическое значение.

Если время работы M нельзя ограничить многочленом, т. е. $T_M(l) = \Omega(l^c)$ для всех $c \in \mathbb{N}$, то говорят, что M работает за *суперполиномиальное* время. Среди суперполиномиальных машин выделяют *квазиполиномиальные*: $T_M(l) = 2^{(\log l)^{O(1)}}$, *субэкспоненциальные*: $T_M(l) = 2^{o(l)}$ и *экспоненциальные*: $T_M(l) = 2^{l^{O(1)}}$. Есть и *суперэкспоненциальные* машины, но мы их рассматривать не будем.

Таблица 2.1. Базовые арифметические задачи

Задача ($a \geqslant b$)	Время
Сложение $(a, b) \mapsto a + b$	$O(\log a + \log b)$
Вычитание $(a, b) \mapsto a - b$	$O(\log a + \log b)$
Умножение $(a, b) \mapsto ab$	$O(\log a \log b)$
Деление $(a, b) \mapsto (q, r)$: $a = qb + r$, $0 \leqslant r < b$	$O(\log q \log b)$
Наибольший общий делитель $(a, b) \mapsto \gcd(a, b)$	$O(\log^2 a)$
Обращение по модулю $(a, b) \mapsto b^{-1} \bmod a$ (a и b взаимно просты)	$O(\log^2 a)$
Возведение в степень по модулю $(a, b, e) \mapsto b^e \bmod a$	$O(\log e \log^2 a)$
Китайская система сравнений $(a_1, \ldots, a_k, b_1, \ldots, b_k) \mapsto b$: $b \equiv b_i \pmod{a_i}$ (a_i попарно взаимно просты)	$O\left(\log^2 a\right)$ $(a = a_1 \ldots a_k)$
Совершенная степень $a \mapsto (b, e)$: $a = b^e$, где e максимально	$O\left((\log a)^{1+o(1)}\right)$

В таблице 2.1 приводятся оценки времени работы алгоритмов, решающих базовые арифметические задачи с натуральными числами. Оценки могут уточняться. Например, умножение может быть выполнено за время $O((\log a)^{\log_2 3})$ с помощью алгоритма Карацубы — Офмана или даже за время $O(\log a \cdot \log\log a \cdot \log\log\log a)$ с помощью алгоритма

Шёнхаге — Штрассена. Тем не менее, даже приведенные оценки означают эффективность базовых арифметических операций.

2.6. Вероятностные машины

Пусть p — нечетное простое, $Q_p = \{b^2 : b \in \mathbb{F}_p^*\}$ — множество квадратичных вычетов по модулю p, $\bar{Q}_p = \mathbb{F}_p^* \setminus Q_p$ — множество квадратичных невычетов.

Рассмотрим задачу поиска квадратичного невычета по заданному модулю p. Для решения этой задачи можно использовать следующий алгоритм.

Алгоритм Поиск невычета (детерминированный)

Вход: p — нечетное простое.
Выход: $a \in \bar{Q}_p$.
Шаги:
1. Для $a = 1, 2, \ldots, p-1$:
   (1) если $a^{(p-1)/2} \equiv -1 \pmod p$, то возвратить a.

Алгоритм обязательно остановится с верным результатом в силу следующих фактов:
1) $a \in \bar{Q}_p$ тогда и только тогда, когда $a^{(p-1)/2} \equiv -1 \pmod p$ (критерий Эйлера);
2) $|Q_p| = |\bar{Q}_p| = (p-1)/2$.

При попытке обосновать эффективность алгоритма возникают трудности. С одной стороны, проверка на шаге 1.1 выполняется эффективно (за кубическое от $\log p$ время). С другой стороны, нельзя гарантировать, что потребуется полиномиальное число таких проверок. Пусть $n_2(p)$ — минимальный квадратичный невычет по модулю p. Доказано (теорема Бургесса), что $n_2(p) \leqslant p^{1/(4\sqrt{e})+\epsilon}$, $\epsilon > 0$. С такой оценкой время работы алгоритма будет экспоненциальным. Более сильная оценка $n_2(p) \leqslant 2\ln^2 p$, справедливая при выполнении расширенной гипотезы Римана, дает полиномиальное время. К сожалению, гипотеза на сегодня не доказана. Можно организовать поиск a не в начале натурального ряда, можно учитывать при поиске вид p (например, $p-1$ будет невычетом, если $p \equiv 3 \pmod 4$), но в целом трудности обоснования эффективности алгоритма сохранятся.

Выходом является переход к вероятностному поиску.

Алгоритм Поиск невычета (вероятностный)

Вход: p — нечетное простое.
Выход: $a \in \bar{Q}_p$.
Шаги:
1. Для $i = 1, 2, \ldots$:
   (1) $a \xleftarrow{R} \{1, 2, \ldots, p-1\}$;
   (2) если $a^{(p-1)/2} \equiv -1 \pmod p$, то возвратить a.

В новом алгоритме запись $u \xleftarrow{R} U$ означает случайный равновероятный выбор u из множества U. Выбранное на шаге 1.1 число a окажется невычетом с вероятностью $1/2$ и в среднем потребуется всего 2 попытки генерации.

Алгоритмы, в которых используется случайный выбор, называются *вероятностными*. Моделью таких алгоритмов является *вероятностная машина Тьюринга* (ВМТ). ВМТ снабжается дополнительной лентой, в ячейках которой записаны символы слова $\omega = \omega_1\omega_2\ldots$. Эти символы являются реализациями независимых в совокупности случайных величин с равномерным распределением на $\{0, 1\}$. Ячейки случайной ленты

читаются слева направо, без повторений. В функцию переходов φ, которая определяет функционирование управляющего устройства машины, добавляется еще один аргумент — очередной символ ω.

Число тактов и сам результат работы M на фиксированном входе x являются случайными величинами. В частности, машина может допускать ошибки в определении правильного ответа, т. е. решать задачу не наверняка, а только с какой-то вероятностью. Мы будем рассматривать три типа ошибок.

1. *Нульсторонние ошибки.* Машина M возвращает либо правильный ответ, либо пустое слово $\perp$, которое означает «ответ не найден».
2. *Односторонние ошибки.* Машина M решает задачу распознавания языка L, не ошибается на входах $x \in L$ и может ошибаться на входах $x \notin L$. Возможно альтернативное определение: M не ошибается на входах $x \notin L$ и может ошибаться на входах $x \in L$.
3. *Двусторонние ошибки.* Машина M решает задачу распознавания языка L и может ошибаться как на входах $x \in L$, так и на входах $x \notin L$.

Введем характеристики трудоемкости работы вероятностных машин.

Определение 2.3. *ВМТ M работает за среднее время $ET_M(l)$, $l \in \mathbb{N}$, если $ET_M(l) = \max_x \mathsf{E}\, t_M(x)$. Здесь максимум берется по допустимым $x \in \{0,1\}^l$, а $\mathsf{E}\, t_M(x)$ — среднее число тактов, которое выполнит M при вычислениях на входе x. Усреднение выполняется по всевозможным заполнениям случайной ленты.*

Определение 2.4. *ВМТ M решает задачу f с вероятностью $P_M(l)$, $l \in \mathbb{N}$, если $P_M(l) = \min_x \mathsf{P}\{M(x) = f(x)\}$, где минимум берется по допустимым $x \in \{0,1\}^l$.*

Если для $T_M(l)$, $ET_M(l)$ важна полиномиальная ограниченность, то для $P_M(l)$ важно быть достаточно большой. Будем говорить, что M работает с *преобладающей* вероятностью успеха, если $P_M(l) \geqslant 1 - \nu(l)$, где ν — пренебрежимо малая функция.

Определение 2.5. *Функция $\nu\colon \mathbb{N} \to [0,1]$ пренебрежимо мала, если для любого $c \in \mathbb{N}$ неравенство $\nu(l) < l^{-c}$ выполняется при всех достаточно больших $l \in \mathbb{N}$.*

Будем также говорить, что M работает с *обратно полиномиальной* вероятностью успеха, если найдется $c \in \mathbb{N}$ такое, что $1/P_M(l) \leqslant l^c$ при всех достаточно больших l.

2.7. Алгоритмы Лас-Вегас и Монте-Карло

Вероятностный алгоритм, который мы рассмотрели в предыдущем параграфе, всегда возвращает правильный ответ и работает за полиномиальное в среднем время. Такие алгоритмы принято называть *алгоритмами Лас-Вегас*.

Даже если среднее время работы вероятностной машины M полиномиально, вычисления на определенных входах при определенных ω могут выполняться суперполиномиально долго. В некоторых случаях время работы необходимо ограничивать. *Полиномиальная вероятностная машина Тьюринга* (ПВМТ) — это ВМТ, которая всегда работает за полиномиальное время, независимо от заполнения случайной ленты. Платой за ограничение по времени является возможность появления ошибок в ответах.

Полиномиальным ВМТ соответствуют *алгоритмы Монте-Карло*. Вот пример такого алгоритма.

Алгоритм Поиск невычета (полиномиальный вероятностный)

Вход: p — нечетное простое.

Выход: $a \in \bar{Q}_p$ или $\perp$.

Шаги:

1. Для $i = 1, \ldots, k$:
   (1) $a \xleftarrow{R} \{1, 2, \ldots, p-1\}$;
   (2) если $a^{(p-1)/2} \equiv -1 \pmod p$, то возвратить a.

2. Возвратить $\perp$.

Здесь $k = (\log p)^{O(1)}$ — параметр, который определяет время работы ($O(k \log^3 p)$) и вероятность получения правильного ответа ($1 - 2^{-k}$). Ошибки алгоритма могут быть только нульсторонними. Рассмотрим такие алгоритмы подробнее.

Пусть A — ВМТ, которая допускает только нульсторонние ошибки. Тогда A может быть преобразована в машину B, которая последовательно обращается к A до тех пор, пока не будет получен ответ, отличный от $\perp$. Машина B не ошибается. Если $\beta > 0$ — вероятность успеха A на некотором входе, то B на том же входе потребуется выполнить $1/\beta$ обращений к A в среднем. Действительно, среднее число обращений

$$\sum_{t=1}^{\infty} t\mathsf{P}\left\{\text{потребуется } t \text{ обращений}\right\} = \sum_{t=0}^{\infty} \mathsf{P}\left\{\text{потребуется} > t \text{ обращений}\right\} = \\ = \sum_{t=0}^{\infty} (1-\beta)^t = \frac{1}{\beta}.$$

Отсюда получаем оценку для среднего времени работы B:

$$ET_B(l) \leqslant \frac{T_A(l)}{P_A(l)}.$$

Если A работает за полиномиальное время с обратно полиномиальной вероятностью успеха, то B работает за полиномиальное в среднем время. Таким образом, переход от A к B есть преобразование алгоритма Монте-Карло в алгоритм Лас-Вегас. Возможно обратное преобразование, которое состоит в принудительной остановке B после определенного числа тактов работы с возвратом $\perp$ после остановки. Фактически этот прием применен в нашем последнем алгоритме поиска невычета.

Машина B может запустить A только k раз, снова ожидая ответ, отличный от $\perp$. Искомый ответ будет получен с вероятностью $\beta_B = 1 - (1 - \beta)^k$ за β_B/β обращений к A в среднем. Машина B, как и A, допускает нульсторонние ошибки, но их вероятность быстро уменьшается с ростом k.

Предположим теперь, что A решает задачу распознавания языка L и допускает односторонние ошибки:

$$\mathsf{P}\left\{A(x) = 1\right\} = \begin{cases} 1, & x \in L, \\ \epsilon, & x \notin L. \end{cases}$$

Здесь $0 < \epsilon < 1$. Построим машину B, который получает на вход x, запускает A на этом входе k раз и фиксирует ответы $y_1, \ldots, y_k$. Если все $y_i = 1$, то B возвращает 1. Если хотя бы один ответ $y_i = 0$, то B возвращает 0. Машина B также распознает язык L, также допускает односторонние ошибки, но уже с меньшими вероятностями:

$$\mathsf{P}\left\{B(x) = 1\right\} = \begin{cases} 1, & x \in L, \\ \epsilon^k, & x \notin L. \end{cases}$$

С ростом k время работы B увеличивается линейно, а вероятность ошибки уменьшается экспоненциально. Изменяя k, можно контролировать качество распознавания, сохраняя приемлемым время работы. Такой подход применяется в вероятностных алгоритмах проверки простоты.

Снизить можно и вероятности двусторонних ошибок. Пусть A допускает двусторонние ошибки и вероятность ошибки на входе x не превосходит $\epsilon < 1/2$. Перестроим алгоритм B так, чтобы он определял свой ответ по $y_1, \ldots, y_k$, руководствуясь *правилом*

большинства: если среди y_i единиц больше, чем нулей, то B возвращает 1, в противном случае B возвращает 0. Тогда вероятность ошибки B на входе x не превышает

$$\sum_{i=\lceil k/2\rceil}^{k} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i} \leqslant \exp\left(-k\frac{(1/2-\epsilon)^2}{(1/2+\epsilon)}\right).$$

Последняя оценка следует из неравенства Чернова (см. задание 2.5).

2.8. Сведéние

Пусть g — некоторая задача (распознавания или поиска). *Оракульная* машина Тьюринга M^g — это машина Тьюринга, снабженная дополнительной лентой, на которую можно записать допустимый вход задачи g и за один такт получить на его месте ответ. Символы оракульной ленты являются дополнительными входами и выходами функции переходов M^g.

Оракульная машина описывает работу обычной машины с гипотетическим внешним вычислительным устройством — *оракулом*. Считается, что ресурсы оракула неограничены и он вычисляет ответ задачи g в течение одного такта работы M^g. Время работы M^g учитывает число запросов к оракулу, но игнорирует сложность их обработки.

Определение 2.6. *Говорят, что имеется полиномиальная сводимость задачи f к задаче g и пишут $f \leqslant_P g$, если найдется оракульная ПМТ M^g, которая решает f. Если $f \leqslant_P g$ и $g \leqslant_P f$, то f и g полиномиально эквивалентны: $f \sim_P g$.*

Определение 2.7. *Говорят, что имеется вероятностная полиномиальная сводимость задачи f к задаче g и пишут $f \leqslant_R g$, если найдется оракульная ПВМТ M^g, которая решает f с преобладающей вероятностью успеха. Если $f \leqslant_R g$ и $g \leqslant_R f$, то f и g вероятностно полиномиально эквивалентны: $f \sim_R g$.*

Сведéние устанавливает частичный предпорядок на множестве задач. Доказывая факт сведéния, мы устанавливаем тем самым, что одна задача в некотором смысле не сложнее другой.

Проиллюстрируем сведéние на примере соотношения между задачами `Factor` и `Sqrt`. `Sqrt` — это задача поиска квадратных корней по модулю. Входными данными `Sqrt` является пара взаимно простых натуральных чисел (n, a), в которой a — квадратичный вычет по модулю n. Выходными данными является число $b \in \mathbb{Z}_n^*$ такое, что $b^2 \equiv a \pmod{n}$. Выход b называют *квадратным корнем* из a по модулю n. Корней может быть несколько. Запись $b \in \sqrt{a} \bmod n$ означает, что b — один из этих корней.

Теорема 2.3. $\texttt{Sqrt} \leqslant_R \texttt{Factor}$.

Доказательство. Для решения $\texttt{Sqrt}(n, a)$ представим n в виде $\prod_{i=1}^{k} p_i^{e_i}$, где p_i — различные простые; e_i — натуральные числа. Для этого выполним $O(\log n)$ обращений к оракулу `Factor`, последовательно разлагая n на множители до тех пор, пока все они не окажутся простыми. Простоту множителя можно проверить за полиномиальное время (см. п. 2.9).

Если мы найдем $b_i \in \sqrt{a} \mod p_i^{e_i}$, то сможем определить b как решение китайской системы сравнений

$$b \equiv \pm b_i \pmod{p_i^{e_i}}, \quad i = 1, 2, \ldots, k.$$

Решение можно найти за время $O(\log^2 n)$ (см. таблицу 2.1). Знаки в системе можно расставлять произвольно. Отметим (нам это потребуется немного позже), что если все p_i нечетны, то имеется 2^k различных корней b, определяемых различными вариантами расстановки знаков.

Корень $r \in \sqrt{a} \mod p^e$, $p \in \{p_1, \ldots, p_k\}$, можно найти следующим образом.

1. Для $p \neq 2$ определим невычет по модулю p. Для этого используем алгоритм из § 2.7 с числом итераций k. Алгоритм работает за время $O(k \log^3 p)$ с вероятностью успеха $1 - 2^{-k}$.
2. Для $p \neq 2$ определим корень $r_0 \in \sqrt{a} \bmod p$. Для этого используем алгоритм Тонелли — Шэнкса. На вход алгоритма передаются p, a и невычет, найденный на предыдущем шаге. Алгоритм работает за время $O(\log^4 p)$. Если $p = 2$, то $a \equiv 1 \pmod p$ и $r_0 = 1$.
3. Последовательно определим корни $r_i \in \sqrt{a} \mod p^{2^i}$, $i = 1, 2, \ldots, \lceil \log_2 e \rceil$, а затем найдем $r = r_{\lceil \log_2 e \rceil} \bmod p^e$. Корень r_i будем искать в виде $r_i = r_{i-1} + p^{2^{i-1}} s$. Должно выполняться сравнение

$$\left(r_{i-1} + p^{2^{i-1}} s\right)^2 = r_{i-1}^2 + 2p^{2^{i-1}} r_{i-1} s + p^{2^i} s^2 \equiv a \pmod{p^{2^i}},$$

откуда

$$s = \left(\frac{\left(a - r_{i-1}^2\right)/p^{2^{i-1}} \bmod p^{2^i}}{2r_{i-1}} \right) \bmod p^{2^{i-1}}$$

(все деления нацело). Время расчетов — $O(e^2 \log^2 p \log e)$.

В целом все корни b_i будут найдены за время

$$\sum_{i=1}^{k} O(k \log^3 p_i + \log^4 p_i + e_i^2 \log^2 p_i \log e_i) = O(\log^5 n)$$

с вероятностью успеха $(1 - 2^{-k})^k \geqslant 9/16$. Алгоритм нахождения корней допускает только нульсторонние ошибки. Повторяя алгоритм полиномиальное число раз, можно сделать вероятность успеха преобладающей. □

Теорема 2.4. `Factor` $\leqslant_R$ `Sqrt`.

Доказательство. Пусть требуется найти нетривиальный делитель d составного n. Если n — четное, то найти решение легко: $d = 2$. Поэтому будем считать, что n — нечетное.

Если n имеет вид p^e, где p — простое, то p можно найти за время $O((\log p)^{1+o(1)})$ (см. таблицу 2.1). Таким образом, мы можем ограничиться случаем, когда n имеет не менее двух различных нечетных простых делителей. В этом случае, как было отмечено в предыдущем доказательстве, из любого квадратичного вычета по модулю n извлекается не менее четырех различных квадратных корней.

Пусть A — алгоритм, который решает `Sqrt`. Следующий алгоритм решает `Factor` с помощью A.

1. Сгенерировать $c \xleftarrow{R} \{2, 3, \ldots, n-1\}$.
2. Если $\gcd(n, c) \neq 1$, то возвратить $\gcd(n, c)$.
3. Установить $a \leftarrow c^2 \bmod n$.
4. Установить $b \leftarrow A(n, a)$.
5. Если $b \equiv \pm c \pmod n$, то возвратить $\perp$.
6. Возвратить $\gcd(b + c, n)$.

Построенный алгоритм является вероятностным полиномиальным. Проанализируем результаты его работы. Ясно, что на шаге 2 возвращается нетривиальный делитель n. Число b, полученное на шаге 4, удовлетворяет сравнениям

$$0 \equiv b^2 - a \equiv b^2 - c^2 \equiv (b - c)(b + c) \pmod n.$$

Поэтому n делит произведение $(b - c)(b + c)$. Если условие на шаге 5 не выполняется, то n не делит ни один из множителей произведения и $\gcd(b + c, n)$ — нетривиальный делитель n.

Всего имеется не менее 4 корней из a по модулю n. Поэтому условие на шаге 5 будет нарушаться не менее чем в половине случаев, и искомый делитель n будет найден с вероятностью

$$\beta = \mathsf{P}\{\gcd(n,c) \neq 1\} + \mathsf{P}\{\gcd(n,c) = 1, b \not\equiv \pm c \pmod n\} \geqslant$$
$$\geqslant \mathsf{P}\{\gcd(n,c) \neq 1\} + \frac{1}{2}\mathsf{P}\{\gcd(n,c) = 1\}) > \frac{1}{2}.$$

Повторяя алгоритм полиномиальное число раз, можно сделать вероятность успеха преобладающей. □

Следствие 2.1. $\texttt{Sqrt} \sim_R \texttt{Factor}$.

2.9. Классы сложности

В этом параграфе мы обсудим ряд важных аспектов сложности вычислений. Эти аспекты связаны только с задачами распознавания. Нас не должно это смущать, поскольку задачи поиска, которые встречаются в криптографии, как правило, легко сводятся к задачам распознавания. Например, `Factor` сводится к распознаванию языка

$$L_{\texttt{Factor}} = \{(n,m)\colon n,m \in \mathbb{N}, \text{существует } d \in \{2,3,\ldots,m\}, \text{которое делит } n\}.$$

Действительно, нетривиальный делитель d числа n можно найти, обращаясь $O(\log n)$ раз к машине, которая распознает $L_{\texttt{Factor}}$, т. е. проверяет, что $2 \leqslant d \leqslant m$. Граница m задается при этих обращениях так, чтобы реализовать дихотомию множества $\{2,3,\ldots,n-1\}$.

Будем рассматривать далее задачу распознавания с предикатом f и языком L.

Определение 2.8. *Язык L (предикат f) принадлежит классу* $\mathbf{P}$*, если существует ПМТ M такая, что:*
1) *если $x \in L$, то $M(x) = 1$;*
2) *если $x \notin L$, то $M(x) = 0$.*

Определение 2.9. *Язык L (предикат f) принадлежит классу* $\mathbf{NP}$*, если существует ПМТ M и многочлен p такие, что:*
1) *если $x \in L$, то $M(x,y) = 1$ для некоторого слова y длины $|y| \leqslant p(|x|)$;*
2) *если $x \notin L$, то $M(x,y) = 0$ для любого y длины $|y| \leqslant p(|x|)$.*

Класс $\mathbf{P}$ — это класс языков, распознаваемых за полиномиальное время. В $\mathbf{NP}$ входят языки, которые также распознаются за полиномиальное время, но с использованием дополнительных данных — слов y. Слово y, которое часто называется *сертификатом*, подтвердает принадлежность $x \in L$, причем подтверждает доказательно, в том смысле, что для $x \notin L$ ни один из сертификатов не будет принят машиной M.

Знаменитая гипотеза $\mathbf{P} \neq \mathbf{NP}$ означает существование предиката f, для которого ответ $f(x)$ не может быть *вычислен* за полиномиальное время, но может быть *проверен* за полиномиальное время с помощью сертификата. Большинство специалистов считает, что гипотеза $\mathbf{P} \neq \mathbf{NP}$ справедлива.

В классе $\mathbf{NP}$ выделяют подкласс $\mathbf{NPC}$ так называемых $\mathbf{NP}$-*полных* языков. Языки из $\mathbf{NPC}$ имеют максимальную сложность: $L \in \mathbf{NPC}$, если для любого $L' \in \mathbf{NP}$ найдется вычислимая на ПМТ функция $g\colon \{0,1\}^* \to \{0,1\}^*$ такая, что $x \in L'$ тогда и только тогда, когда $g(x) \in L$. Если $\mathbf{NPC} \cap \mathbf{P} \neq \varnothing$, то $\mathbf{P} = \mathbf{NP}$ и наоборот.

Рассмотрим несколько примеров.

Пример 2.1. Задача `SAT` (от англ. satisfiability) формулируется следующим образом. Задана булева формула, составленная из переменных, скобок, символов $\neg$ (отрицание), $\vee$ (и), $\wedge$ (или). Требуется ответить, можно ли назначить переменным значения 1 (истина) и 0 (ложь) так, чтобы формула стала истинной. В начале 1970-х гг. С. Кук и Л. Левин получили результаты, которые означают, что `SAT` лежит в классе $\mathbf{NPC}$ (фактически Кук и Левин ввели этот класс). □

Пример 2.2. Задача `SubsetSum` определяется языком

$$\left\{(x_1, x_2, \ldots, x_n, y)\colon x_i, y \in \mathbb{Z},\ \text{существует } S \subseteq \{1, 2, \ldots, n\} \text{ такое, что } \sum_{i \in S} x_i = y\right\}.$$

Доказано, что этот язык является **NP**-полным. □

Пример 2.3. Пусть G — игра, в которой участвуют Алиса и Боб. Конфигурация (состояние) игры кодируется словом $x \in \{0,1\}^*$. Введем язык

$$L = \{x \in \{0,1\}^*\colon \text{Алиса имеет выигрышную стратегию в конфигурации } x\}.$$

Найдены примеры игр G, для которых $L \notin \mathbf{NP}$. Для этих игр нельзя найти сертификат, который позволял бы проверить наличие выигрышной стратегии за полиномиальное время. □

Еще один класс сложности связан с вероятностными алгоритмами.

Определение 2.10. *Язык L принадлежит классу* **BPP**, *если существует ПВМТ M такая, что*

1) *если $x \in L$, то* $\mathsf{P}\{M(x) = 1\} \geqslant 2/3$;
2) *если $x \notin L$, то* $\mathsf{P}\{M(x) = 0\} \geqslant 2/3$.

Другими словами, языки из класса **BPP** распознаются на машинах Монте-Карло с двусторонними ошибками, вероятность которых $\leqslant 1/3$. Выбор порога $1/3$ условен. Важно только, чтобы вероятность ошибки не превосходила $1/2 - \delta$ для некоторого фиксированного $\delta > 0$. Применяя M для распознавания x несколько раз и вынося решение по правилу большинства (см. § 2.7), можно достаточно быстро приблизить вероятность успешного распознавания к 1.

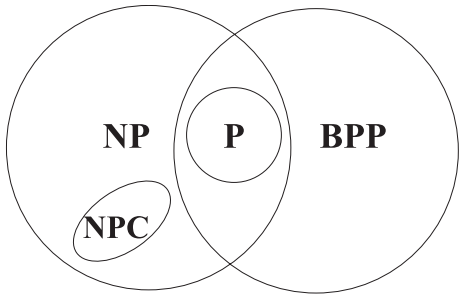


Рис. 2.1. Диаграмма классов сложности (гипотетическая)

На рисунке 2.1 представлено гипотетическое (признаваемое большинством специалистов) соотношение между описанными классами сложности.

2.10. Язык *PRIMES*

В этом параграфе мы рассмотрим сложность распознавания языка *PRIMES*.

Теорема 2.5 (Пратт). $PRIMES \in \mathbf{NP}$.

Доказательство. Пусть для $a \in \mathbb{Z}_n^*$ выполняется: $a^{n-1} \equiv 1 \pmod{n}$ и

$$a^{(n-1)/q_i} \not\equiv 1 \pmod{n}, \quad i = 1, 2, \ldots, k,$$

где $q_1, q_2, \ldots, q_k$ — все простые делители $n-1$. Тогда порядок a в группе $\mathbb{Z}_n^*$ равняется $n-1$. По теореме Лагранжа порядок элемента группы делит порядок группы. Следовательно, $n-1$ делит $|\mathbb{Z}_n^*| = \varphi(n)$, что справедливо только тогда, когда n — простое. Таким образом, выполнение указанных условий доказывает простоту числа n.

Рассмотрим машину M, которая берет на вход натуральное n и сертификат

$$\operatorname{cert}(n) = (a, (q_1, \operatorname{cert}(q_1)), \ldots, (q_k, \operatorname{cert}(q_k)))\,.$$

Простоту чисел q_i демонстрируют сертификаты $\operatorname{cert}(q_i)$, которые имеют такую же структуру, как и $\operatorname{cert}(n)$, т. е. включают основание из $\mathbb{Z}_{q_i}^*$, простые делители числа $q_i - 1$ и сертификаты этих делителей. В сертификаты делителей могут быть вложены новые сертификаты и так далее. В целом получается целое дерево сертификатов. Сертификаты $\operatorname{cert}(1)$, $\operatorname{cert}(2)$ отдельно определяются как пустые слова.

Общее число вершин в дереве сертификатов для $n = 2$ и для нечетного $n \geqslant 3$ не превосходит $2\log_2 n - 1$. Действительно, это верно для $n = 2$ и $n = 3$. Если $n > 3$, то $n - 1 = q_1 q_2 \ldots q_k$ — составное ($k \geqslant 2$) и число сертификатов по индукции не больше

$$1 + \sum_{i=1}^{k} (2\log_2 q_k - 1) = 1 + 2\log_2(q_1 q_2 \ldots q_k) - k < 2\log_2 n - 1.$$

Длина каждого сертификата за вычетом длины вложенных сертификатов есть $O(\log n)$. Поэтому $|\operatorname{cert}(n)| = O(\log^2 n)$.

Машина M обрабатывает $(n, \operatorname{cert}(n))$ следующим образом.

1. Если $n = 1$, то возвратить 0.
2. Если $n = 2$, то возвратить 1.
3. Если n — четное, то возвратить 0.
4. Если $a^{n-1} \not\equiv 1 \pmod n$, то возвратить 0.
5. Если $\prod_{i=1}^{k} q_i \neq n - 1$, то возвратить 0.
6. Для $i = 1, \ldots, k$:
   1) если $a^{(n-1)/q_i} \equiv 1 \pmod n$, то возвратить 0;
   2) если $M(q_i, \operatorname{cert}(q_i)) = 0$, то возвратить 0.
7. Возвратить 1.

Машина M работает за полиномиальное время (проверить самостоятельно), всегда дает ответ 1 на простых n и никогда не дает ответ 1 на составных n. Мы находимся в условиях определения класса **NP** и теорема доказана. □

Дополнительный к $PRIMES$ язык $\overline{PRIMES} = \{n \in \mathbb{N}\colon n \text{ — составное}\}$ также лежит в классе **NP** — сертификатом принадлежности n языку является любой нетривиальный делитель. Анализ вычислительных задач показывает, что если и язык L, и дополнительный язык $\overline{L}$ одновременно лежат в **NP**, то, как правило, $L \in \mathbf{P}$. Действительно, в 2002 году индийские математики М. Агравал, Н. Каяла и Н. Саксена (АКС) разработали полиномиальный алгоритм распознавания простоты, т. е. доказали, что $PRIMES \in \mathbf{P}$.

К сожалению, алгоритм АКС является довольно медленным, даже самые эффективные его редакции работают за время $O(\log^6 n)$. В криптографии для проверки простоты в основном применяют вероятностный алгоритм Рабина — Миллера, который работает за время $O(\log^3 n)$, допускает только односторонние ошибки (составное может быть признано простым), вероятность ошибки не превосходит $1/4$. Существование этого алгоритма даже без результатов АКС доказывает, что $PRIMES \in \mathbf{BPP}$.

2.11. Односторонние функции

Криптографические системы строятся по принципу «легко для легального пользователя (Алиса), трудно для противника (Виктор)». Дело сводится к построению функций, образы которых легко определить по прообразам, но прообразы трудно восстановить по образам. Формализуем интуитивное представление о таких функциях, отождествляя Алису и Виктора с машинами Тьюринга, ресурсы которых полиномиально ограничены.

Определение 2.11. *Функция* $f\colon \{0,1\}^* \to \{0,1\}^*$ *называется односторонней, если:*

1) *существует ПМТ, которая вычисляет* f*;*
2) *для любой ПВМТ* M *найдется пренебрежимо малая функция* $\boldsymbol{\nu}_M$ *такая, что*

$$\mathsf{P}\left\{f(M(1^l, f(x))) = f(x)\colon x \overset{R}{\leftarrow} \{0,1\}^l\right\} \leqslant \boldsymbol{\nu}_M(l).$$

Здесь вероятности определяются случайным выбором x *и случайной лентой, которую* M *использует при работе.*

Машина M, которая фигурирует в определении, получает на вход слово $y = f(x)$, полученное по случайному x. Машине надо обратить f, т. е. найти слово x', которое не обязательно совпадает с x, но для которого $f(x') = f(x)$. Требуется, чтобы любая полиномиальная машина находила искомое слово лишь с пренебрежимо малой вероятностью.

Обратим внимание на то, что на вход M вместе с $f(x)$ подается длина $l = |x|$. Число l представляется не словом длины $O(\log l)$, как обычно, а унарным кодом 1^l. Это значит, что M может работать за полиномиальное от $|x|$ время, даже если слово $f(x)$ будет значительно короче слова x. При этом односторонними не будут признаваться некоторые функции, интуитивно не подходящие на эту роль. Например, функция $f_{\text{len}}(x) = |x|$. Эту функцию нельзя обратить за полинимиальное от $|f(x)|$ время (просто не хватит времени выписать ответ), хотя она легко обращается за линейное от $|x|$ время.

Существование односторонних функций до настоящего времени не доказано. Как показывает следующая теорема, вопрос существования связан с известными нерешенными проблемами теории сложности.

Теорема 2.6. *Условия* $\mathbf{P} \neq \mathbf{NP}$, $\mathbf{NP} \not\subseteq \mathbf{BPP}$ *являются необходимыми для существования односторонних функций.*

Доказательство. Пусть $f\colon \{0,1\}^* \to \{0,1\}^*$ — односторонняя функция. Введем в рассмотрение язык L, составленный из троек $(1^l, y, S)$, в которых $y \in \{0,1\}^*$ и $S \subseteq \{1, 2, \ldots, l\}$. Тройка $(1^l, y, S) \in L$, если найдется слово $x \in \{0,1\}^l$ такое, что $f(x) = y$ и $x_i = 1$ для всех $i \in S$. Слово x является сертификатом принадлежности тройки языку, проверка принадлежности выполняется за полиномиальное время, следовательно, L лежит в $\mathbf{NP}$.

Предположим, что $\mathbf{P} = \mathbf{NP}$. Тогда $L \in \mathbf{P}$, т. е. существует машина A, которая проверяет принадлежность $(1^l, y, S) \in L$ за полиномиальное время без сертификата. По A построим машину B, которая берет на вход пару $(1^l, y)$. В этой паре $y = f(x)$ для некоторого $x \in \{0,1\}^l$. Машина B работает следующим образом.

1. Установить $S \leftarrow \varnothing$.
2. Для $i = 1, 2, \ldots, l$:
   (1) если $A(1^l, y, S \cup \{i\}) = 1$, то $S \leftarrow S \cup \{i\}$.
3. Построить $x' \in \{0,1\}^l$ такое, что $x'_i = 1$, только если $i \in S$.
4. Возвратить x'.

Машина B обращает f, делает это за полиномиальное время, и следовательно, f не является односторонней. Условие $\mathbf{P} \neq \mathbf{NP}$ действительно является необходимым для односторонности f.

Аналогично доказывается необходимость условия $\mathbf{NP} \not\subseteq \mathbf{BPP}$. Машины A и B становятся вероятностными, A распознает L с вероятностью ошибки $\leqslant 1/3$. Машина B на шаге 2.1 вызывает A не один, а несколько раз и обрабатывает ответы A по правилу большинства. Число обращений к A выбирается так, чтобы вероятность ошибки распознавания L не превосходила $1/l$. При этом машина B обратит f с вероятностью успеха не менее $1 - (1 - 1/l)^l \geqslant 1 - e^{-1}$, которая не является пренебрежимо малой. □

В доказательстве теоремы использована полиномиальная сводимость задачи обращения f к задаче распознавания языка $L \in \mathbf{NP}$. При построении f можно воспользоваться

сводимостью другого вида: задачи распознавания языка L' к задаче обращения f. Можно строить f так, чтобы язык L' было трудно распознать, например, чтобы $L' \in \mathbf{NPC}$. Можно ожидать при этом, что f будет трудно обратить. Оказывается, что это не так: сложность обращения f оценивается в среднем, в то время как сложность распознавания языков характеризуется наихудшим случаем (максимальная сложность на входах определенной длины). Поэтому обращение f является трудным в наихудшем случае (что подтверждается сведе́нием), но может быть простым в среднем (что не противоречит сведе́нию).

Пример 2.4. Пусть

$$f(x_1, \ldots, x_n, S) = \left(x_1, \ldots, x_n, \sum_{i \in S} x_i \right), \quad x_i \in \mathbb{Z}, \quad S \subseteq \{1, 2, \ldots, n\}.$$

Задача распознавания **NP**-полного языка `SubsetSum` сводится к задаче обращения f. Это, однако, не означает, что f является односторонней: почти для всех входов f множество S может быть найдено по выходу за полиномиальное время (с помощью LLL-алгоритма или его модификаций). На этом наблюдении базируются атаки на *рюкзачные криптосистемы* — криптосистемы, основанные на различных уточнениях задачи `SubsetSum`. □

2.12. Функции с лазейкой

Известна аналогия между односторонней функцией и телефонным справочником: по фамилии x легко определить номер телефона $f(x)$, однако определение фамилии по номеру — трудная задача. Функции с лазейкой, которые мы сейчас введем, являются хитро устроенными справочниками — обладая специальным секретом (лазейкой), поиск в справочнике нужного номера можно выполнить очень быстро.

Определение 2.12. *Функция* $f\colon \{0,1\}^* \to \{0,1\}^*$ *называется функцией с лазейкой, если:*

1) f *— односторонняя функция;*
2) *существует ПВМТ* I *и слова* $t_1, t_2, \ldots \in \{0,1\}^*$, $|t_l| = l^{O(1)}$, *такие, что* $f(x) = f(x')$ *для всех* $x \in \{0,1\}^l$ *и соответствующих* $x' = I(1^l, f(x), t_l)$.

Функции с лазейкой являются односторонними и соответствуют правилу «легко вычислить, трудно обратить». Дополнительно поддерживается правило «легко обратить с лазейкой». Имеется в виду, что кроме Алисы есть еще один легальный пользователь — Боб, который отождествляется с машиной I. Бобу известна *лазейка* t_l, с помощью которой он находит x' по $(1^l, f(x))$.

В криптографии функции с лазейкой могут использоваться для построения *систем ЭЦП* и *криптосистем с открытым ключом*. Эти системы будут подробно рассмотрены в соответствующих главах книги. Здесь мы остановимся на принципах построения. Сразу оговоримся, что упомянутые криптографические системы не обязательно должны строиться на основе функций с лазейкой (хотя на практике функции с лазейкой явно или неявно используются при построении).

В общем случае применяется не фиксированная функция с лазейкой, а семейство таких функций. Функции семейства имеют конечные области определений. С помощью вероятностного алгоритма `Gen`, который работает за полиномиальное в среднем время, Боб выбирает одну из таких функций. Алгоритм `Gen` берет на вход слово 1^l и возвращает описание f и лазейку t, нужную для обращения f. Описание f включает спецификацию области определения D и области значений E. Входной параметр l определяет размерности D и E.

Описание f называется *открытым ключом*, лазейка t — *личным*. Открытый ключ Боб делает общедоступным, личный ключ хранит в секрете.

В криптосистемах с открытым ключом функция f биективна. Алисе требуется передать Бобу конфиденциальное сообщение $x \in D$ — *открытый текст*. Алиса использует открытый ключ Боба и за полиномиальное время находит $y = f(x)$ — *шифртекст*. Боб за полиномиальное время определяет x по (y, t). Виктор не знает личный ключ t, ему требуется определить x по y, что является трудной задачей обращения f.

В системах ЭЦП Боб подтверждает подлинность документа $y \in E$. Зная t, Боб за полиномиальное время находит $x \in D$ такое, что $f(x) = y$. Слово x называется *электронной цифровой подписью* y. Алиса проверяет подпись, сравнивая $f(x)$ с y. Для определения подписи x без личного ключа t Виктору снова требуется обратить f.

Детали могут отличаться. Например, Алиса может зашифровывать открытый текст, дополненный случайными данными, или Боб может подписывать не сам документ, а его хэш-значение. Отличия важны, но не принципиальны. Суть использования функций с лазейкой сохраняется.

2.13. Функция Рабина

Функции, которые доказательно являются функциями с лазейкой, пока не известны. Тем не менее, построены функции, которые гипотетически (по мнению большинства специалистов) являются таковыми. Рассмотрим одну из них – *функцию Рабина*.

Функция Рабина f (точнее, представитель семейства функций Рабина) описывается натуральным $n = pq$, где p и q — различные простые, сравнимые с 3 по модулю 4. Такое n называется *числом Блюма*. Будем считать, что битовые длины p и q примерно равны, а битовая длина n равняется l: $\log_2 p \approx \log_2 q$, $\lceil \log_2 n \rceil = l$. Длина l выбирается Бобом при вызове алгоритма `Gen`. Этот алгоритм находит два случайных подходящих простых (сделать это можно за полиномиальное в среднем время), а затем их произведение.

Область определения $D = \mathbb{Z}_n$, область значений E — множество квадратов в $\mathbb{Z}_n$, лазейкой является пара (p, q). Действие f:

$$f(x) = x^2 \bmod n.$$

Рассмотрим задачи, которые решают Алиса, Боб и Виктор при использовании f как функции с лазейкой.

1. Алисе требуется вычислить $y = f(x)$. Сделать это можно за время $O(l^2)$.
2. Бобу по (y, p, q) требуется найти $x' \in \sqrt{y} \bmod n$. Модуль n выбран так, чтобы упростить извлечение корней по модулям его делителей:

$$x_1 = y^{(p+1)/4} \bmod p \in \sqrt{y} \bmod p, \quad x_2 = y^{(q+1)/4} \bmod q \in \sqrt{y} \bmod q.$$

Боб находит x_1, x_2, а затем решает китайскую систему

$$x' \equiv \pm x_1 \pmod{p}, \quad x' \equiv \pm x_2 \pmod{q}$$

с произвольной расстановкой знаков $\pm$. Искомый корень Боб найдет за время $O(l^3)$.

3. Виктору также требуется найти $x' \in \sqrt{y} \bmod n$, но уже по (n, y). Пусть $d = \gcd(y, n)$. Если $d > 1$, то $d \in \{p, q\}$, и Виктор узнает личный ключ Боба. После этого Виктор может определить x', действуя как Боб. Но при $x \xleftarrow{R} \mathbb{Z}_n$ вероятность

$$\mathsf{P}\{d > 1\} = \frac{n - \varphi(n)}{n} = \frac{1}{p} + \frac{1}{q} - \frac{1}{pq}$$

пренебрежимо мала. Если же $d = 1$, то Виктору требуется решить задачу `Sqrt`, вероятностно полиномиально эквивалентную `Factor` (см. следствие 2.1). Но `Factor` — задача, признаваемая трудной, следовательно задача `Sqrt` также трудна, и x' снова определяется лишь с пренебрежимо малой вероятностью.

Если f предполагается использовать для шифрования, то область определения D сужают так, чтобы получить биекцию. Например, зашифровывают слова x со специальным фиксированным префиксом, а после расшифрования используют тот из корней, который удовлетворяет выбранному формату.

2.14. Задания

Задание 2.1. Язык $L = \{0,1\}^*1$ состоит из непустых слов, которые заканчиваются единицей. Описать таким же образом следующие языки:
1) $\{0,1\}^*1\{0,1\}^{10}$;
2) $\{0,1\}^*1\{0,1\}^*1\{0,1\}^*$;
3) 0^*0100^*.

Задание 2.2. Доказать, что множество $\{0,1\}^*$ счетное. Доказать, что множество функций $\{0,1\}^* \to \{0,1\}$ — континуум.

Задание 2.3. Подтвердите оценки таблицы 2.1 (первые четыре строки).

Задание 2.4. Пусть $n_2(p)$ — минимальный квадратичный невычет по нечетному простому модулю p. Доказать, что $n_2(p)$ будет простым числом.

Задание 2.5 (неравенство Чернова). Пусть $\xi_1, \ldots, \xi_k$ — независимые одинаково распределенные бернуллиевские случайные величины, $\mathsf{P}\{\xi_1 = 1\} = \epsilon < 1/2$. Пусть $S = \xi_1 + \ldots + \xi_k$. Тогда справедливо неравенство Чернова:

$$\mathsf{P}\{S \geqslant (1+\delta)\mu\} \leqslant \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu, \quad \mu = \mathsf{E}S = k\epsilon.$$

Используя это неравенство, доказать, что

$$\mathsf{P}\{S \geqslant k/2\} \leqslant \exp\left(-k\frac{(1/2-\epsilon)^2}{(1/2+\epsilon)}\right)$$

(подсказка: воспользоваться оценкой $\ln(1+\delta) > \frac{2\delta}{2+\delta}$, справедливой для $\delta > 0$).

Задание 2.6. Доказать полиномиальную сводимость задачи распознавания $L_{\texttt{Factor}}$ к задаче `Factor`.

Задание 2.7. Доказать полиномиальную сводимость задачи факторизации чисел Блюма к задаче вычисления значений функции Эйлера.

Задание 2.8. Пусть f — предикат с языком

$$L = \{(p,g)\colon p \text{ — простое, } g \text{ — первообразный корень } \bmod p\}.$$

Доказать, что $f \leqslant_P \texttt{Factor}$.

Задание 2.9. Доказать, что $\mathbf{P} \subseteq \mathbf{NP}$, $\mathbf{P} \subseteq \mathbf{BPP}$.

Задание 2.10. Пусть $L = \{(p,g)\colon p \text{ — простое, } g \text{ — примитивный элемент } \mathbb{F}_p^*\}$. Доказать, что L и дополнительный язык $\bar{L}$ лежат в $\mathbf{NP}$.

Задание 2.11. Пусть $L = \{n\colon n \text{ свободно от квадратов}\}$. Доказать, что L, $\bar{L} \in \mathbf{NP}$.

Задание 2.12. Пусть $L = \{(p,a)\colon p \text{ — простое}, a \text{ — квадратичный вычет } \bmod p\}$. Доказать, что L, $\bar{L} \in \mathbf{NP}$.

Задание 2.13 (формула Тонелли). Пусть p — нечетное простое. Доказать, что корень $b \in \sqrt{a} \bmod p^e$ можно определить по корню $c \in \sqrt{a} \bmod p$ следующим образом:

$$b = c^{p^{e-1}} a^{(p^e - 2p^{e-1}+1)/2} \bmod p^e.$$

Задание 2.14. Обратить функцию Рабина: найти все решения сравнения $x^2 \equiv y \pmod n$ при $n = 19 \cdot 11$ и $y = 100$.

2.15. Комментарии

Развернутую информацию по различным (в том числе криптографическим) аспектам теории сложности можно найти в книгах [14, 24, 25]. Использованная в главе терминология в целом соответствует русскоязычным книгам [7, 8]. На сайте `http://www.cryptography.ru` размещен справочник по математической криптографии, в котором представлена подробная информация по вопросам, затронутым в последних параграфах.

Гамма-код, а также некоторые другие способы кодирования натуральных чисел введены П. Элиасом в [23].

Базовые теоретико-числовые алгоритмы описаны в книгах [4, 15, 19, 37].

Характеристика $s_M(x)$, которая используется в определении 2.2, учитывает как число воспомогательных ячеек памяти, так и длину входа и выхода. Чтобы учесть в $s_M(x)$ только вспомогательные ячейки (например, чтобы выделить класс машин, которые работают на логарифмической вспомогательной памяти), вводят машины с несколькими лентами. Одна лента используется для чтения, на нее записываются входные данные. Еще одна лента используется для записи выходных данных. Остальные ленты называются рабочими, именно они поддерживают вспомогательную память. На каждой ленте имеется свое управляющее устройство. Функция переходов усложняется — теперь она описывает управление не одной, а несколькими лентами. Характеристика $s_M(x)$ определяется как суммарное число ячеек рабочих лент, использованное при вычислениях на входе x.

Некоторые специалисты считают субэкспоненциальным не время $2^{o(l)}$, а время $2^{l^{o(1)}}$. При этом субэкспоненциальными не признаются многие алгоритмы (например, алгоритмы решета числового поля с временем работы $2^{(c+o(1))l^{1/3}(\log l)^{2/3}}$), которые другие специалисты классифицируют все-таки как субэкспоненциальные.

Имеются технические трудности использования понятия «полиномиальное в среднем время», если оно вводится в соответствии с определением 2.3. Можно подобрать пример, когда $\mathsf{E}t_M(x)$ полиномиально увеличивается с ростом $|x|$, но $\mathsf{E}t_M^2(x)$ растет уже с суперполиномиальной скоростью. Такая ситуация неприемлема при организации сведéний между задачами. Поэтому полиномиальное в среднем время определяют по-другому: требуют, чтобы существовали положительные константы C и ϵ, для которых

$$\mathsf{E}\frac{t_M(x)^\epsilon}{l} \leqslant C$$

при $x \xleftarrow{R} \{0,1\}^l$. Новое определение позволяет применить неравенство Маркова и получить оценку:

$$\mathsf{P}\left\{t_M(x) \geqslant (Cdl)^{1/\epsilon}\right\} = \mathsf{P}\left\{\frac{t_M(x)^\epsilon}{l} \geqslant Cd\right\} \leqslant 1/d.$$

Оценка означает, что машина M с высокой вероятностью работает за полиномиальное время.

Терминология, касающаяся алгоритмов Лас-Вегас, разнится. В некоторых источниках алгоритмам Лас-Вегас разрешается возвращать символ $\perp$ (ответ не найден). При этом алгоритмы, которые все-таки всегда возвращают правильный ответ, принято называть алгоритмами Шервуд.

Упомянутые результаты Кука и Левина представлены в работах [20], [5]. Кук рассматривал задачи распознавания, а Левин — задачи поиска. В [32] Р. Карп опубликовал знаменитый список, в который вошла 21 задача из класса **NPC**. Интересно, что Кук обратил внимание на затруднения при классификации $PRIMES \stackrel{?}{\in} \mathbf{NPC}$, а Карп — на затруднения при классификации $\overline{PRIMES} \stackrel{?}{\in} \mathbf{NPC}$. Теорема 2.5 о классификации $PRIMES$

доказана в [39].

Различные классы сложности подробно описаны и классифицированы в [30]. Интернет-каталог классов сложности поддерживает С. Аронзон (см. `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`).

Функции, соответствующие определению 2.11, иногда называют сильно односторонними. Выделяют еще слабо односторонние функции — условие «M обратит f с пренебрежимо малой вероятностью» меняется на «M не обратит f с обратно полиномиальной вероятностью». Точнее, второе условие в определении 2.11 принимает следующий вид:

2') найдется $c \in \mathbb{N}$ такое, что для любой ПВМТ M

$$\mathsf{P}\left\{f(M(1^l, f(x))) \neq f(x) \colon x \xleftarrow{R} \{0,1\}^l\right\} \geqslant l^{-c}$$

при всех достаточно больших $l \in \mathbb{N}$.

Доказано (см. напр. [25, предложение 2.3.1]), что если существуют слабо односторонние функции, то существуют и сильно односторонние.

Определение 2.12 соответствует [26]. Семейством функций с лазейкой называется множество функций $f \colon D_f \to E_f$, где $D_f, E_f \subseteq \{0,1\}^*$ — конечные множества. Семейство должно удовлетворять следующим ограничениям:

1) существует ПВМТ `Gen`, которая берет на вход слово 1^l и возвращает слова $\mathrm{descr}(f)$ и t, длины которых полиномиально ограничены (в зависимости от l);
2) существует ПМТ, которая берет на вход $\mathrm{descr}(f)$ и $x \in D_f$ и возвращает $f(x)$;
3) существует ПВМТ, которая берет на вход $\mathrm{descr}(f)$, t и $y \in E_f$ и возвращает $x' \in D_f$ такое, что $f(x') = y$;
4) существует ПВМТ, которая берет на вход $\mathrm{descr}(f)$ и возвращает реализацию случайной величины с равномерным распределением на D_f;
5) для любой ПВМТ M найдется пренебрежимо малая функция $\boldsymbol{\nu}_M$ такая, что

$$\mathsf{P}\left\{f(M(1^l, \mathrm{descr}(f), f(x))) = f(x) \colon (\mathrm{descr}(f), t) \leftarrow \mathtt{Gen}(1^l), x \xleftarrow{R} D_f\right\} \leqslant \boldsymbol{\nu}_M(l).$$

Функция Рабина введена в [40] как альтернатива функции RSA. М. Рабин предложил использовать новую функцию для построения систем ЭЦП.

Глава 3
БЛОЧНЫЕ КРИПТОСИСТЕМЫ

3.1. Блочное шифрование

Пусть Алиса и Боб обмениваются сообщениями по открытому каналу связи и пусть сообщениями являются слова в алфавите A. Алиса и Боб опасаются, что канал прослушивается противником Виктором и для обеспечения конфиденциальности своей переписки решают использовать шифрование с помощью блочных криптосистем. Термин «блочные» объясняется тем, что шифрование сообщения выполняется *блоками* — словами в алфавите A, которые имеют фиксированную длину n_b.

Прежде чем определить блочные криптосистемы, введем несколько понятий, связанных с подстановками. *Подстановкой* на множестве B называют биективное преобразование этого множества. Обозначим через $S(B)$ множество всех таких преобразований. *Композицией* $\sigma_1, \sigma_2 \in S(B)$ называется подстановка, которая обозначается $\sigma_2\sigma_1$ и действует по правилу

$$\sigma_2\sigma_1(x) = \sigma_2(\sigma_1(x)), \quad x \in B.$$

Множество $S(B)$ с операцией композиции является группой, которая называется *симметрической*. Единицей этой группы является тождественная подстановка id: $id(x) = x$. Если $\sigma_2\sigma_1 = id$, то подстановка σ_2 является *обратной* к σ_1, что записывается как $\sigma_2 = \sigma_1^{-1}$.

Определение 3.1. *Блочной криптосистемой называется семейство ключезависимых подстановок*

$$F = \{F_\theta \colon \theta \in \Theta\} \subseteq S(A^{n_b}).$$

Здесь Θ — множество ключей, A^{n_b} — множество блоков сообщений, F_θ — подстановка зашифрования, действие которой определяется ключом θ и которой соответствует подстановка расшифрования F_θ^{-1}.

Как правило, $|\Theta| \ll |S(A^{n_b})|$, т. е. криптосистема представляет собой лишь малое подмножество допустимых подстановок. Желательно, чтобы элементы подмножества были «разбросаны» по $S(A^{n_b})$ как можно более хаотично. Это цель, которая ставится при проектировании современных блочных криптосистем.

Действие подстановок F_θ, F_θ^{-1} задается алгоритмически. Высокое быстродействие алгоритмов зашифрования и расшифрования является еще одной целью при проектировании F.

Алиса и Боб используют криптосистему F в следующем протоколе (интерактивном алгоритме). Кроме открытого канала связи, по которому передаются сообщения, в протоколе используется также секретный канал, по которому доставляются ключи. Алисе и Бобу помогает третья доверенная сторона — Трент. Тренту доверяются генерация и доставка ключей.

Протокол Защищенная передача данных

Предназначен для конфиденциальной передачи сообщений $X \in A^*$

Стороны: Алиса, Боб, Трент.

Каналы: открытый канал связи (ОКС), секретный канал связи (СКС).

Генерация ключей:

1. Трент: $\theta \xleftarrow{R} \Theta$.

2. Трент $\overset{\text{СКС}}{\to}$ Алиса: θ.
3. Трент $\overset{\text{СКС}}{\to}$ Боб: θ.

Передача X (длина X кратна n_b):

1. Алиса:
   (1) разбивает X на блоки $X_1, \ldots, X_T \in A^{n_b}$;
   (2) выполняет зашифрование блоков: $Y_t \leftarrow F_\theta(X_t)$, $t = 1, \ldots, T$;
   (3) формирует шифртекст $Y \leftarrow Y_1 \parallel \ldots \parallel Y_T$.
2. Алиса $\overset{\text{ОКС}}{\to}$ Боб: Y.
3. Боб:
   (1) разбивает Y на блоки $Y_1, \ldots, Y_T \in A^{n_b}$;
   (2) выполняет расшифрование блоков: $X_t \leftarrow F_\theta^{-1}(Y_t)$, $t = 1, \ldots, T$;
   (3) собирает открытый текст $X \leftarrow X_1 \parallel \ldots \parallel X_T$.

Здесь и далее символ $\parallel$ обозначает конкатенацию (объединение) слов.

Если длина передаваемого сообщения X не обязательно кратна n_b, то Алиса и Боб должны предусмотреть обработку последнего (возможно, неполного) блока $X_T \in A^n$, $n \leqslant n_b$.

Простейшая схема обработки последнего блока состоит в следующем. Алиса дополняет X_T произвольными символами до слова длины n_b, зашифровывает X_T и дополнительно зашифровывает блок X_{T+1}, содержащий представление числа n словом из A^{n_b}. Боб расшифровывает последний блок, получает n, расшифровывает предпоследний блок и отбрасывает в нем $n_b - n$ заключительных символов.

Можно организовать обработку последнего блока так, чтобы дополнительный блок формировался не всегда. Пусть α и β — различные символы A. Алиса дописывает к X_T символ α (маркер), а затем минимальное число символов β до получения слова, длина которого кратна n. Алиса зашифровывает X_T и, если блок X_T был полным, дополнительный блок $X_{T+1} = \alpha\beta\beta\ldots\beta$. Боб после расшифрования и формирования открытого текста отбрасывает в нем заключительные символы β вплоть до маркера α, который также отбрасывается.

Третья схема обработки последнего блока может применяться при $T \geqslant 2$. Обработка организована так, что длина шифртекста совпадает с длиной открытого текста. Алиса зашифровывает два последних блока следующим образом (рис. 3.1):

$$Y_T \parallel Y^* \leftarrow F_\theta(X_{T-1}), \quad Y_{T-1} \leftarrow F_\theta(X_T \parallel Y^*),$$

а Боб следующим образом выполняет их расшифрование:

$$X_T \parallel Y^* \leftarrow F_\theta^{-1}(Y_{T-1}), \quad X_{T-1} \leftarrow F_\theta(Y_T \parallel Y^*).$$

Данную схему обработки называют *кражей блока*, имея в виду использование («похищение») Y^* для дополнения X_T.

3.2. Задачи криптоанализа

Посмотрим на предыдущий протокол глазами противника Виктора. При оценке надежности криптосистем предполагают, что противник обладает максимально возможным потенциалом. Поэтому, во-первых, считают, что Виктор точно знает устройство F, ему неизвестен только ключ θ, который доставляется по секретному каналу связи. Данное предположение есть известный в криптологии *принцип Керкгоффса* — надежность криптосистемы определяется лишь секретностью ключа Во-вторых, Виктор прослушивает открытый канал связи и, таким образом, перехватывает все блоки шифртекста Y_t. В-третьих, Виктор знает полностью или частично некоторые блоки открытого текста X_t или даже имеет возможность выбирать эти блоки.

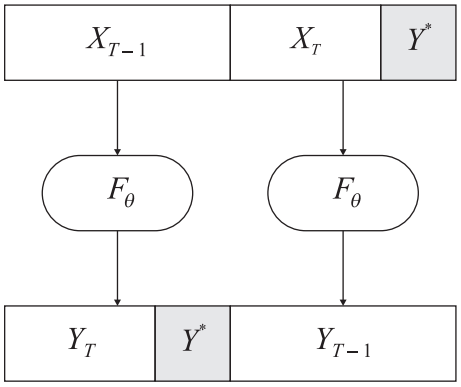


Рис. 3.1. Кража блока

По доступным данным, которые называются *шифрматериалом*, Виктору требуется решить задачу

C1: определить ключ θ подстановки F_θ.

Зная θ, Виктор может определить блок открытого текста $X_{T+1} = F_\theta^{-1}(Y_{T+1})$ по любому перехваченному блоку шифртекста $Y_{T+1} = F_\theta(X_{T+1})$. Вообще говоря, для определения X_{T+1} Виктору не обязательно находить ключ, ему достаточно решить задачу

C2: построить алгоритм нахождения X_{T+1} по заданному $Y_{T+1} = F_\theta(X_{T+1})$.

Чтобы решить поставленные задачи, Виктору требуется найти слабые места криптосистемы. Ими могут быть свойства, которыми, как правило, обладают подстановки F и, как правило, не обладают подстановки $S(A^{n_b})$. Всякое такое свойство позволяет отличить подстановку криптосистемы от других подстановок, и поэтому поиск свойств может быть сформулирован в виде задачи

C3: считая, что шифрматериал получен с помощью подстановки $\sigma \in S(A^{n_b})$, определить, является эта подстановка элементом F или нет.

Задачу **C3** можно интерпретировать как поддержку принципа Керкгоффса. Действительно, решив задачу Виктор узнает, используют ли Алиса и Боб криптосистему F или нет.

Задача **C1** является самой сложной (и практически значимой), задача **C3** – самой простой. Как правило, если найден способ решения одной задачи, то появляются подходы к решению всех остальных.

Алгоритмы решения задач криптоанализа называются *атаками*. Сложность атаки, как и любого другого алгоритма, характеризуется временем и памятью (см. § 2.5). Атака может быть вероятностной, и тогда появляется еще одна характеристика сложности — вероятность успеха (см. § 2.6). Криптографическую специфику атаки характеризует *объем шифрматериала* — количество T блоков шифртекста (и, возможно, открытого текста), требуемых для проведения атаки.

В зависимости от качества шифрматериала выделяют следующие типы атак:

1) известны $\{Y_t\}$ и свойства открытого текста $\{X_t\}$ (*атака при известном шифртексте*);
2) известны $\{X_t\}$ и $\{Y_t\}$ (*атака при известном открытом тексте*);
3) можно выбрать $\{X_t\}$ и получить $\{Y_t\}$ (*атака при выбранном открытом тексте*);
4) можно выбирать X_t, зная $Y_1, \ldots, Y_{t-1}$ (*атака при выбираемом открытом тексте*).

Условия атак последних типов кажутся искусственными. Это действительно так, если считать, что атаки применяются к протоколу из предыдущего пункта. Однако, как демонстрируют следующие примеры, атаки имеют практическое значение, если рассмотреть расширения этого протокола.

Пример 3.1 (GSM). В сетях связи GSM второго поколения речевые данные оцифровываются. Каждым $18,4$ мс разговора соответствует двоичное слово длиной 184. Для противодействия помехам в канале связи слово (как вектор-строка) умножается на двоичную матрицу размера 184×456. В результате получается кодовое слово X, которое обладает структурными особенностями: имеется $456 - 184$ независимых линейных комбинаций символов X, которые обязательно обращаются в $0 \bmod 2$. Кодовое слово X разбивается на 4 фрейма — слова длиной 114. Каждый фрейм зашифровывается перед отправкой в канал связи. Виктор, который перехватывает зашифрованные фреймы, знает о стуктурных особенностях соответствующего открытого текста X (хотя не располагает информацией о самих речевых данных). □

Пример 3.2 (формат). Сообщение X представляет собой файл определенного формата. В частности, X всегда начинается фиксированным заголовком X_1, известным Виктору. □

Пример 3.3 («Энигма»). Во время Второй мировой войны британская специальная служба обрабатывала шифрматериал немецкой шифровальной машины «Энигма». Для организации атаки при выбранном открытом тексте англичане по агентурным каналам доводили в немецкие подразделения информацию (ложную или правдивую) о наличии мин в тех или иных районах. Последующие сообщения немцев обязательно содержали слово «Minen» (мины, нем.). □

Пример 3.4 (пограничные шифраторы). Имеются два сегмента корпоративной сети, соединенные открытым каналом. На границах сегментов установлены шифраторы «Алиса» и «Боб», которые выполняют шифрование данных обмена. Виктор является пользователем сети, может выбрать любой открытый текст X_t для передачи в другой сегмент и перехватить соответствующий шифртекст Y_t. □

3.3. Блочно-итерационные криптосистемы

К. Шеннон предложил строить подстановки F_θ как многократные композиции преобразований усложнения и перемешивания. Преобразования усложнения отвечают за установление сложных зависимостей между отдельными символами шифруемых данных и ключа, преобразования перемешивания распространяют эти зависимости по всему блоку шифруемых данных. Принцип Шеннона реализован в *блочно-итерационных* криптосистемах. Принцип оказался чрезвычайно плодотворным — все современные блочные криптосистемы являются блочно-итерационными.

Блочно-итерационная криптосистема F задается следующими элементами:

1) число *тактов* d;
2) множество *тактовых ключей* K;
3) алгоритм $\mathtt{KS}$ (от англ. Key Schedule, *расписание ключей*), который по ключу $\theta \in \Theta$ строит тактовые ключи $\kappa_1, \kappa_2, \ldots, \kappa_d \in K$;
4) семейство *тактовых подстановок* $\Sigma = \{\Sigma_\kappa \colon \kappa \in K\} \subseteq S(A^{n_b})$.

Подстановки зашифрования и расшифрования определяются по правилам:

$$F_\theta = \Sigma_{\kappa_d} \ldots \Sigma_{\kappa_2} \Sigma_{\kappa_1}, \quad F_\theta^{-1} = \Sigma_{\kappa_1}^{-1} \ldots \Sigma_{\kappa_{d-1}}^{-1} \Sigma_{\kappa_d}^{-1}.$$

Правила означают, что зашифрование открытого текста $X \in A^{n_b}$ состоит в последовательном применении тактовых подстановок $\Sigma_{\kappa_1}, \Sigma_{\kappa_2}, \ldots, \Sigma_{\kappa_d}$. Наоборот, расшифрование шифртекста $Y \in A^{n_b}$ состоит в последовательном применении обратных тактовых подстановок $\Sigma_{\kappa_d}^{-1}, \Sigma_{\kappa_{d-1}}^{-1}, \ldots, \Sigma_{\kappa_1}^{-1}$.

Правила можно задать алгоритмически.

Алгоритм Блочно-итерационное зашифрование

Вход: $X \in A^{n_b}$ — открытый текст, $\theta \in \Theta$.
Выход: $Y \in A^{n_b}$ — шифртекст.
Шаги:
1. $(\kappa_1, \ldots, \kappa_d) \leftarrow \mathtt{KS}(\theta)$.
2. $Y \leftarrow X$.
3. Для $i = 1, \ldots, d$: $Y \leftarrow \Sigma_{\kappa_i}(Y)$.
4. Возвратить Y.

Алгоритм Блочно-итерационное расшифрование

Вход: $Y \in A^{n_b}$ — шифртекст, $\theta \in \Theta$.
Выход: $X \in A^{n_b}$ — открытый текст.
Шаги:
1. $(\kappa_1, \ldots, \kappa_d) \leftarrow \mathtt{KS}(\theta)$.
2. $X \leftarrow Y$.
3. Для $i = 1, \ldots, d$: $X \leftarrow \Sigma^{-1}_{\kappa_i}(X)$.
4. Возвратить X.

Тактовые подстановки Σ_κ имеют, как правило, простое строение и состоят в замене и перестановке символов подлежащего преобразованию слова. Однако многократная композиция таких подстановок определяет сложную зависимость между открытым текстом, шифртекстом и ключом.

Существует множество модификаций описанной блочно-итерационной конструкции. Приведем некоторые из них.

Неоднородные такты. Действие тактовых подстановок определяется не только ключом, но и номером такта: $F_\theta = \Sigma_{d,\kappa_d} \ldots \Sigma_{2,\kappa_2}\Sigma_{1,\kappa_1}$.

Дополнительные бесключевые подстановки. Перед первым тактом и после последнего применяются дополнительные бесключевые подстановки $\tau_1, \tau_2 \in S(A^{n_b})$: $F_\theta = \tau_2\Sigma_{\kappa_d} \ldots \Sigma_{\kappa_1}\tau_1$. Например, в криптосистемах Фейстеля $\tau_1 = id$, а τ_2 состоит в перестановке половинок блоков из A^{n_b} (n_b — четное).

Отбеливание. На A^{n_b} вводится групповая операция $+$, по θ строится дополнительный ключ $\kappa_{d+1} \in A^{n_b}$ и на последнем шаге алгоритма зашифрования вместо Y возвращается $Y + \kappa_{d+1}$.

При атаках на блочно-итерационные криптосистемы у Виктора имеется несколько перспективных возможностей. Во-первых, Виктору не обязательно определять ключ θ, достаточно найти тактовые ключи $\kappa_1, \kappa_2, \ldots, \kappa_d$, т. е. вместо задачи **C1** решить задачу **C2**. Во-вторых, тактовые ключи можно определить последовательно: сначала κ_d, затем κ_{d-1} и так далее. При этом реализуется важный криптоаналитический принцип: *divide et impera* (разделяй и властвуй, лат.). Определение каждого следующего тактового ключа является более простой задачей, чем предыдущего. Труднее всего определить κ_d. Для этого Виктор может решить задачу **C3** для подстановки $F'_\theta = \Sigma_{\kappa_{d-1}} \ldots \Sigma_{\kappa_2}\Sigma_{\kappa_1}$, т. е. найти некоторое отличительное свойство этой подстановки. Затем Виктор проверяет всевозможные ключи-кандидаты $\hat{\kappa}_d$ и в качестве искомой оценки κ_d выбирает тот из них, на котором для подстановки $\hat{F}'_\theta = \Sigma^{-1}_{\hat{\kappa}_d} F_\theta$ найденное свойство проявляется в наибольшей степени.

3.4. Операции над двоичными словами

В современных блочно-итерационных криптосистемах тексты и ключи являются двоичными словами. Откажемся от рассмотрения неиспользуемых на практике вариантов и случаев, и везде далее будем считать, что $A = \{0, 1\}$.

Построение криптосистемы сводится к организации преобразований усложнения и перемешивания над двоичными словами. Слова можно объединять, разбивать на блоки, блоки слов можно менять местами или заменять на другие блоки. Однако перечисленных операций может быть недостаточно. Для расширения набора преобразований удобно считать, что слова представляют элементы некоторых алгебраических структур, и задействовать операции этих структур. Рассмотрим распространенные представления.

Слова как векторы. Слово $a_1a_2\dots a_n \in \{0,1\}^n$ представляет вектор

$$a = (a_1, a_2, \dots, a_n) \in \mathbb{F}_2^n.$$

Векторы одинаковой размерности можно складывать: если $b = (b_1, b_2, \dots, b_n)$, то

$$a + b = (a_1 + b_1, a_2 + b_2, \dots, a_n + b_n).$$

Сложение выполняется в поле $\mathbb{F}_2$, т. е. по модулю 2. Такое сложение в криптографии часто обозначается знаком $\oplus$ («круглый плюс»).

Кроме этого, векторы можно умножать на матрицы над полем $\mathbb{F}_2$. Умножение на перестановочную матрицу задает перестановку координат вектора. Особый вид перестановки — циклический сдвиг. Через $a \lll r$ будем обозначать результат циклического сдвига вектора a на r позиций влево.

Через $a \cdot b$ будем обозначать скалярное произведение векторов a и b:

$$a \cdot b = a_1b_1 + a_2b_2 + \ldots + a_nb_n,$$

через $\mathrm{w}(a)$ — *вес Хэмминга* (число ненулевых координат) вектора a. Пусть $\mathbf{0}$ — нулевой вектор.

Слова как числа. Слово $a_1a_2\dots a_n$ представляет число

$$a = a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_n.$$

Наоборот, всякое число $a \in \{0, 1, \dots, 2^n - 1\}$ представляется двоичным словом длины n. Это слово будем обозначать через $\langle a \rangle_n$.

Число a интерпретируется как элемент кольца $\mathbb{Z}_{2^n}$. Это кольцо составлено из целых от 0 до $2^n - 1$, их сложение и умножение выполняется по модулю 2^n. Умножение в $\mathbb{Z}_{2^n}$ редко используется при построении блочных криптосистем, а вот сложение — достаточно часто. Операцию сложения принято обозначать знаком $\boxplus$ («квадратный плюс»), операцию вычитания — знаком $\boxminus$ («квадратный минус»).

Слова как элементы поля характеристики 2. Слово $a_1a_2\dots a_n$ представляет многочлен

$$a(\lambda) = a_1\lambda^{n-1} + a_2\lambda^{n-2} + \ldots + a_n \in \mathbb{F}_2[\lambda].$$

Этот многочлен интерпретируется как элемент факторкольца $\mathbb{F}_2[\lambda]/(f(\lambda))$, где $f(\lambda) \in \mathbb{F}_2[\lambda]$ — неприводимый многочлен степени n. Элементами факторкольца являются многочлены, степени которых меньше n. Эти элементы складываются и умножаются как обычные многочлены, произведение дополнительно приводится по модулю f. Поскольку f — неприводим, факторкольцо $\mathbb{F}_2[\lambda]/(f(\lambda))$ является полем. Это поле состоит из 2^n элементов. Все такие поля изоморфны друг другу и представляют одно и то же поле $\mathbb{F}_{2^n}$: $\mathbb{F}_2[\lambda]/(f(\lambda)) \cong \mathbb{F}_{2^n}$.

В поле появляется дополнительная операция умножения, которая обозначается знаком $*$ или, как обычно, опускается. Сложение слов как элементов $\mathbb{F}_{2^n}$ экивалентно сложению слов как элементов $\mathbb{F}_2^n$.

Элементы $\mathbb{F}_{2^n}$ — это в общем случае абстрактные объекты, не обязательно многочлены. Тем не менее всякий элемент $a \in \mathbb{F}_{2^n}$ может быть представлен вектором $(a_1, a_2, \dots, a_n) \in \mathbb{F}_2^n$ и далее словом $a_1a_2\dots a_n \in \{0,1\}^n$. Для этого можно выбрать базис $\alpha_1, \alpha_2, \dots, \alpha_n$ поля $\mathbb{F}_{2^n}$ как векторного пространства над $\mathbb{F}_2$ и записать a в виде

$$a = a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n.$$

Координаты a_i можно найти по формуле

$$a_i = \mathrm{Tr}(\alpha_i^* a), \quad i = 1, 2, \dots, n.$$

Здесь $\mathrm{Tr}\colon \mathbb{F}_{2^n} \to \mathbb{F}_2$, $x \mapsto x + x^2 + x^{2^2} + \ldots + x^{2^{n-1}}$ — функция абсолютного следа, а $\{\alpha_i^*\}$ — базис $\mathbb{F}_{2^n}$ над $\mathbb{F}_2$, *дуальный* к $\{\alpha_i\}$:

$$\mathrm{Tr}(\alpha_i \alpha_j^*) = \begin{cases} 1, & i = j, \\ 0, & \text{в противном случае.} \end{cases}$$

Функция Tr является линейным отображением $\mathbb{F}_{2^n} \to \mathbb{F}_2$, где $\mathbb{F}_{2^n}$ и $\mathbb{F}_2$ рассматриваются как векторные пространства над $\mathbb{F}_2$. Более того, любое линейное отображение $L\colon \mathbb{F}_{2^n} \to \mathbb{F}_2$ имеет вид

$$L(x) = \mathrm{Tr}(bx), \quad b \in \mathbb{F}_{2^n}.$$

В частности, если $a \in \mathbb{F}_2^n$, а x интерпретируется и как элемент поля, и как вектор, то

$$a \cdot x = \mathrm{Tr}(bx)$$

при подходящем выборе $b \in \mathbb{F}_{2^n}$.

Слова как элементы простого поля. Если $p = 2^n + 1$ — простое, то слова из $\{0,1\}^n$ представляют элементы мультипликативной группы простого поля $\mathbb{F}_p = \{1, 2, \ldots, p-1\}$. Ненулевые слова-как-числа представляют элементы $1, 2, \ldots, p-2$ этой группы, а нулевое слово — элемент $p - 1 = 2^n$. Умножение в $\mathbb{F}_p^*$ обозначается знаком $\odot$ («круглая точка») или опускается.

Числа p указанного выше вида называются *простыми Ферма*. На сегодняшний день известно 5 таких простых: 3, 5, 17, 257, 65537.

Обозначения для алгебраических структур, элементы которых представляют двоичные слова, будем переносить на сами слова, и от них распространять на все остальные структуры. Можно сказать, что множество двоичных является общей платформой для всех структур. Будем подчеркивать связь между структурами знаком $\sim$: $\{0,1\}^n \sim \mathbb{F}_2^n \sim \mathbb{Z}_{2^n} \sim \mathbb{F}_{2^n} \sim \mathbb{F}_{2^n+1}^*$.

Пример 3.5. В криптосистеме `AES` *октеты* (слова длины 8) отождествляются с векторами $\mathbb{F}_2^8$ и элементами поля $\mathbb{F}_{2^8} \cong \mathbb{F}_2[\lambda]/(\lambda^8 + \lambda^4 + \lambda^3 + \lambda + 1)$. Используемый здесь неприводимый многочлен даже получил именное название: *многочлен AES*. Октеты кодируются числами из $\mathbb{Z}_{2^8}$, представленными в шестнадцатеричной системе счисления. Например,

$$\begin{aligned} \text{слово } 01010111 \sim \text{вектор } (0,1,0,1,0,1,1,1) \sim \text{число } \mathtt{57}_{16} \sim \\ \sim \text{элемент поля } \lambda^6 + \lambda^4 + \lambda^2 + \lambda + 1. \end{aligned}$$

□

3.5. Булевы функции и отображения

3.5.1. S-блоки

Интересующие нас криптографические преобразования в конце концов оказываются отображениями $\{0,1\}^n \to \{0,1\}^m$ или $\mathbb{F}_2^n \to \mathbb{F}_2^m$. Обозначим через $\mathcal{F}_{n,m}$ множество всех таких отображений. Множество $\mathcal{F}_n = \mathcal{F}_{n,1}$ является множеством булевых функций $\mathbb{F}_2^n \to \mathbb{F}_2$, а всякое отображение $\sigma \in \mathcal{F}_{n,m}$ можно задать m координатными булевыми функциями $\sigma_1, \ldots, \sigma_m \in \mathcal{F}_n$ так, что

$$\sigma(x) = (\sigma_1(x), \ldots, \sigma_m(x)), \quad x \in \mathbb{F}_2^n.$$

Для $a = (a_1, \ldots, a_m) \in \mathbb{F}_2^m$ через $a \cdot \sigma$ будем обозначать линейную комбинацию координатных функций σ:

$$a \cdot \sigma(x) = a_1\sigma_1(x) + \ldots + a_m\sigma_m(x).$$

При малых n и m отображения $\sigma \in \mathcal{F}_{n,m}$, которые используются для усложнения при построении блочных криптосистем, принято называть *S-блоками*. S-блоки, как правило, задаются таблично. Вычисление значения $\sigma(x)$ сводится к обращению к ячейке таблицы по индексу x.

S-блоки могут быть фиксированными (DES), могут быть частью ключа θ (GOST) либо определяться по этому ключу (Blowfish). S-блоки могут быть сжимающими ($n > m$, DES), расширяющими ($n < m$, Blowfish) или сохраняющими размерность ($n = m$, GOST). Особый интерес представляют биективные S-блоки $\sigma \in S(\mathbb{F}_2^n) \subset \mathcal{F}_{n,n}$.

S-блоки должны обладать свойствами, затрудняющими применение тех или иных методов криптоанализа. Основными критериями выбора S-блоков являются: высокая нелинейность, большие степени координатных функций и их линейных комбинаций, малые значения в таблицах разностей. Эти критерии мы опишем в следующих параграфах, предварительно введя необходимый технический аппарат.

Существует три основных подхода к построению S-блоков:

1. *Случайная генерация.* Таблица значений S-блока заполняется случайно или псевдслучайно. Если криптографические характеристики полученного S-блока не являются удовлетворительными, то таблица генерируется заново. Как правило, с увеличением размерностей время генерации качественного S-блока быстро растет, что затрудняет применение данного подхода.
2. *Алгоритмические конструкции.* Задается алгоритм вычисления образов $\sigma(x)$, в котором используются эффективно реализуемые аппаратно и программно арифметические и логические операции, а также S-блоки меньшей размерности. Как правило, криптографические характеристики алгоритмических S-блоков не являются оптимальными.
3. *Алгебраические конструкции.* При построении S-блоков используются операции алгебраических структур, описанных в 3.4. Правила, определяющие действие S-блока, являются достаточно простыми, что позволяет провести теоретическое исследование криптографических характеристик. Правила выбирают так, чтобы характеристики были близки к оптимальным.

В пп. 3.5.9, 3.5.10 мы подробно рассмотрим две алгебраические конструкции S-блоков. Кроме этого, в следующем примере определяются модельные S-блоки, которые также построены с помощью алгебраических операций. Сразу скажем, что модельные S-блоки не являются оптимальными, это позволит нам впоследствии провести несколько атак на модельную криптосистему, в которых эти S-блоки используются.

Пример 3.6 (модельные S-блоки). S-блоки S_1 и S_2 действуют на $\{0,1\}^4 \sim \mathbb{F}_2^4 \sim \mathbb{Z}_{16} \sim \mathbb{F}_{17}^*$ по правилам:

$$S_1(x) = ((3^x \bmod 17) + 2) \bmod 16;$$
$$S_2(x) = ((5^x \bmod 17) + 7) \bmod 16, \quad x = 0, 1, \ldots, 15.$$

В правых частях выражений неявно используются операции $\odot$ и $\boxplus$. Поскольку 3 и 5 — примитивные элементы $\mathbb{F}_{17}$, модельные S-блоки являются биективными. □

3.5.2. Аддитивный характер

Введем в рассмотрение функцию $\chi\colon \mathbb{F}_2 \to \mathbb{R}$, $c \mapsto (-1)^c$. В теории конечных полей эту функцию принято называть *аддитивным характером*, имея в виду, что

$$\chi(c_1 + c_2) = \chi(c_1)\chi(c_2), \quad c_i \in \mathbb{F}_2.$$

Лемма 3.1 (тождество для характера). *Для $a \in \mathbb{F}_2^n$ выполняется*

$$\frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}\chi(a\cdot x)=\begin{cases}1, & a=\mathbf{0},\\ 0 & a\neq\mathbf{0}.\end{cases}$$

Доказательство. Для нулевого a равенство очевидно. При $a \neq \mathbf{0}$ имеем

$$\sum_{x\in\mathbb{F}_2^n}\chi(a\cdot x)=\sum_{x_1\in\mathbb{F}_2}\dots\sum_{x_n\in\mathbb{F}_2}\chi(a_1x_1)\dots\chi(a_nx_n)=\left(\sum_{x_1\in\mathbb{F}_2}\chi(a_1x_1)\right)\dots\left(\sum_{x_n\in\mathbb{F}_2}\chi(a_nx_n)\right)=0,$$

поскольку $\chi(a_i\cdot 0)+\chi(a_i\cdot 1)=1-1=0$, если $a_i=1$. □

Выражение в правой части доказанного тождества есть индикатор наступления события $\mathcal{E}=\{a=0\}$. Этот индикатор будем обозначать через $\mathsf{I}\{\mathcal{E}\}$, распространяя обозначение на произвольные события $\mathcal{E}$.

3.5.3. Преобразование Уолша – Адамара

Преобразование Уолша — Адамара ставит в соответствие функции $f \in \mathcal{F}_n$ вещественнозначную функцию

$$\hat{f}(u)=\sum_{x\in\mathbb{F}_2^n}\chi(f(x))\chi(u\cdot x),\quad u\in\mathbb{F}_2^n.$$

Функцию f можно восстановить по $\hat{f}$:

$$\begin{aligned}\frac{1}{2^n}\sum_{u\in\mathbb{F}_2^n}\hat{f}(u)\chi(u\cdot x)&=\frac{1}{2^n}\sum_{u\in\mathbb{F}_2^n}\chi(u\cdot x)\sum_{y\in\mathbb{F}_2^n}\chi(f(y)+u\cdot y)=\\&=\frac{1}{2^n}\sum_{u\in\mathbb{F}_2^n}\sum_{y\in\mathbb{F}_2^n}\chi(f(y)+u\cdot(x+y))=\\&=\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\chi(f(y))\sum_{u\in\mathbb{F}_2^n}\chi(u\cdot(x+y))=\chi(f(x)).\end{aligned}$$

Значения $\hat{f}(u)$ называются *коэффициентами Уолша — Адамара*. Коэффициенты удовлетворяют *равенству Парсеваля*:

$$\begin{aligned}\sum_{u\in\mathbb{F}_2^n}\hat{f}(u)^2&=\sum_{u\in\mathbb{F}_2^n}\sum_{x,y\in\mathbb{F}_2^n}\chi(f(x)+f(y))\chi(u\cdot(x+y))=\\&=\sum_{x,y\in\mathbb{F}_2^n}\chi(f(x)+f(y))\sum_{u\in\mathbb{F}_2^n}\chi(u\cdot(x+y))=\\&=2^n\sum_{x\in\mathbb{F}_2^n}\chi(0)=2^{2n}.\end{aligned}$$

Из этого равенства следует, что

$$\max_{u\in\mathbb{F}_2^n}|\hat{f}(u)|\geqslant 2^{n/2}.$$

3.5.4. Ортогональные системы

Функция $f \in \mathcal{F}_n$ называется *уравновешенной*, если она принимает значения 0 и 1 одинаковое количество раз. Другими словами, f — уравновешенная, если $\hat{f}(\mathbf{0}) = 0$.

Система булевых функций $\sigma_1, \ldots, \sigma_m \in \mathcal{F}_n$ называется *ортогональной*, если для каждого вектора $(b_1, \ldots, b_m) \in \mathbb{F}_2^m$ система уравнений

$$\sigma_1(x) = b_1, \ldots, \sigma_m(x) = b_m$$

имеет ровно 2^{n-m} решений относительно x.

Очевидно, ортогональные системы существуют только при $m \leqslant n$. При $m = n$ ортогональной системе соответствует подстановка

$$\sigma = (\sigma_1, \ldots, \sigma_n) \in S(\mathbb{F}_2^n).$$

Теорема 3.1 (критерий ортогональности). *Система $\sigma_1, \ldots, \sigma_m \in \mathcal{F}_n$ является ортогональной тогда и только тогда, когда для каждого ненулевого $a \in \mathbb{F}_2^m$ функция $a \cdot \sigma$ является уравновешенной.*

Доказательство. Для $b = (b_1, \ldots, b_m) \in \mathbb{F}_2^m$ через $N(b)$ обозначим число решений системы уравнений $\sigma_i(x) = b_i$, $i = 1, \ldots, m$.

Если система $\{\sigma_i\}$ ортогональна, то для ненулевого $a \in \mathbb{F}_2^m$ выполняется

$$\begin{aligned}\widehat{a \cdot \sigma}(\mathbf{0}) &= \sum_{x \in \mathbb{F}_2^n} \chi(a_1\sigma_1(x)) \ldots \chi(a_m\sigma_m(x)) = \\ &= \sum_{b \in \mathbb{F}_2^m} N(b)\chi(a_1b_1) \ldots \chi(a_mb_m) = \\ &= 2^{n-m} \sum_{b \in \mathbb{F}_2^m} \chi(a \cdot b) = 0\end{aligned}$$

и функция $a \cdot \sigma$ сбалансирована.

Обратно, если функция $a \cdot \sigma$ сбалансирована и $\widehat{a \cdot \sigma}(\mathbf{0}) = 0$ для всех ненулевых a, то для любого b выполняется:

$$\begin{aligned}N(b) &= \sum_{x \in \mathbb{F}_2^n} \left(\frac{1}{2} \sum_{a_1 \in \mathbb{F}_2} \chi(a_1(\sigma_1(x) + b_1)) \right) \ldots \left(\frac{1}{2} \sum_{a_m \in \mathbb{F}_2} \chi(a_m(\sigma_m(x) + b_m)) \right) = \\ &= \frac{1}{2^m} \sum_{a \in \mathbb{F}_2^m} \chi(a \cdot b) \sum_{x \in \mathbb{F}_2^n} \chi(a_1\sigma_1(x) + \ldots + a_m\sigma_m(x)) = 2^{n-m}.\end{aligned}$$

□

Теорема показывает, что при изучении биективных S-блоков важно исследовать свойства уравновешенных булевых функций, поскольку координатные функции S-блока и их невырожденные линейные комбинации являются таковыми.

3.5.5. Нелинейность

Расстоянием Хэмминга между функциями $f, g \in \mathcal{F}_n$ называется число несовпадений их значений:

$$\rho(f, g) = \sum_{x \in \mathbb{F}_2^n} \mathsf{I}\{f(x) \neq g(x)\}.$$

Пусть $\mathcal{A}_n = \{f \in \mathcal{F}_n \colon \deg f \leqslant 1\}$ — множество *аффинных функций* от n переменных. Всякая функция $l \in \mathcal{A}_n$ имеет вид $l(x) = a \cdot x + b$, где $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2$.

Нелинейностью функции $f \in \mathcal{F}_n$ называется расстояние от нее до множества аффинных функций:

$$\mathrm{nl}(f) = \rho(f, \mathcal{A}_n) = \min_{l \in \mathcal{A}_n} \rho(f, l).$$

Расстояние между f и l определяется через коэффициент Уолша — Адамара:

$$\begin{aligned}\rho(f,l) &= 2^n - \sum_{x \in \mathbb{F}_2^n} \mathsf{I}\{f(x) = l(x)\} = \\ &= 2^n - \frac{1}{2} \sum_{x \in \mathbb{F}_2^n} \sum_{c \in \mathbb{F}_2} \chi(c(f(x) + l(x))) = \\ &= 2^{n-1} - \chi(b) \frac{1}{2} \sum_{x \in \mathbb{F}_2^n} \chi(f(x) + a \cdot x) = \\ &= 2^{n-1} - \chi(b) \frac{1}{2} \hat{f}(a).\end{aligned}$$

Поэтому

$$\mathrm{nl}(f) = 2^{n-1} - \frac{1}{2} \max_{u \in \mathbb{F}_2^n} |\hat{f}(u)|$$

и

$$\mathrm{nl}(f) \leqslant 2^{n-1} - 2^{n/2-1}.$$

Функции f, для которых достигается последняя оценка, существуют при четных n и называются *бент-функциями*. Достижимые оценки сверху для $\mathrm{nl}(f)$ при нечетных $n \geqslant 9$ на сегодняшний день неизвестны. Неизвестны также максимальные значения нелинейности уравновешенных функций от $n \geqslant 11$ переменных. В таблице 3.1 представлены известные результаты, касающиеся максимальной нелинейности функций от малого числа переменных.

Таблица 3.1. Максимальные значения $\mathrm{nl}(f)$

Функция f	Число переменных n						
	4	5	6	7	8	9	10
Произвольная	6	12	28	56	120	от 242 до 244	496
Уравновешенная	4	12	24	56	112	240	480

Линейной аппроксимацией для $\sigma \in \mathcal{F}_{n,m}$ называется пара векторов $a \in \mathbb{F}_2^n$ и $b \in \mathbb{F}_2^m$. При случайном равновероятном выборе x из $\mathbb{F}_2^n$ определяется *вероятность аппроксимации*

$$\eta_{ab}(\sigma) = \mathsf{P}\{a \cdot x = b \cdot \sigma(x)\}.$$

Чем выше эта вероятность, тем с большей точностью можно предсказать линейную комбинацию $b \cdot \sigma(x)$ по линейной комбинации $a \cdot x$. Если вероятность близка к нулю, то можно изменить прогноз $b \cdot \sigma(x)$ на $b \cdot \sigma(x) + 1$, сделав вероятность прогнозирования близкой к 1. Таким образом, качество аппроксимации характеризует абсолютное значение величины $\epsilon_{ab}(\sigma) = \eta_{ab}(\sigma) - 1/2$, которую принято называть *преобладанием*.

Преобладание связано с коэффициентами Уолша — Адамара:

$$\begin{aligned}\epsilon_{ab}(\sigma) &= \frac{1}{2^n}\rho(a\cdot x, b\cdot\sigma(x)) - \frac{1}{2} = \\ &= \frac{1}{2^n}\left(2^{n-1} - \frac{1}{2}\widehat{b\cdot\sigma}(a)\right) - \frac{1}{2} = \\ &= \frac{1}{2^{n+1}}\widehat{b\cdot\sigma}(a).\end{aligned}$$

Отсюда

$$\max_{a,b\neq\mathbf{0}}|\epsilon_{ab}(\sigma)| = \frac{1}{2} - \frac{1}{2^n}\,\mathrm{nl}(\sigma),$$

где $\mathrm{nl}(\sigma)$ — *нелинейность* σ:

$$\mathrm{nl}(\sigma) = \min_{b\in\mathbb{F}_2^m\setminus\{\mathbf{0}\}}\mathrm{nl}(b\cdot\sigma).$$

При построении блочных криптосистем отображения σ выбирают так, чтобы их нелинейность была максимально большой. При этом преобладания всевозможных нетривиальных линейных аппроксимаций для σ будут близки к 0.

Пример 3.7 (нелинейность модельных *S*-блоков). Для модельных S-блоков, описанных в примере 3.6, выполняется: $\mathrm{nl}(S_1) = 2$, $\mathrm{nl}(S_2) = 0$. Зафиксируем следующие расстояния, которые нам пригодятся в дальнейшем:

$$\rho(0011\cdot S_1(x), 1110\cdot x + 1) = 2, \quad \rho(1100\cdot S_2(x), 0001\cdot x + 1) = 4.$$

3.5.6. Многочлены Жегалкина

Для $a, b \in \mathbb{F}_2^n$ обозначим $a^b = \prod_{i=1}^n a_i^{b_i}$. Здесь предполагается, что $0^0 = 1^0 = 1^1 = 1$ и $0^1 = 0$. Ясно, что $a^b = 1$ тогда и только тогда, когда $b_i \leqslant a_i$ для всех $i = 1,\ldots,n$ (сравнение в обычном арифметическом смысле, т. е. $0\leqslant 0$, $0\leqslant 1$, $1\leqslant 1$). Систему неравенств $b_i \leqslant a_i$ будем записывать просто как $b\leqslant a$. Если, дополнительно, $b\neq a$, то пишем $b < a$.

Функцию $f\in\mathcal{F}_n$ можно представить *многочленом Жегалкина* (в англоязычной литературе он называется *алгебраической нормальной формой*). Это многочлен от переменных $x = (x_1,\ldots,x_n)$ над $\mathbb{F}_2$, каждый его моном содержит любую из переменных x_i в степени не выше первой:

$$f(x) = \sum_{u\in\mathbb{F}_2^n} c_u x^u \in \mathbb{F}_2[x], \quad c_u\in\mathbb{F}_2.$$

Теорема 3.2 (Жегалкин, 1927). *Каждая булева функция представляется единственным многочленом Жегалкина.*

Доказательство. При подстановке в многочлене Жегалкина на места переменных x_i всевозможных значений из $\mathbb{F}_2$ получается некоторая функция $\mathbb{F}_2^n\to\mathbb{F}_2$. Имеется 2^{2^n} различных многочленов и столько же различных функций. Остается показать, что различные многочлены представляют различные функции.

Пусть, от противного, различные многочлены p_1 и p_2 представляют одну и ту же функцию f. Тогда ненулевой многочлен $p = p_1 - p_2$ представляет нулевую функцию. Пусть p содержит моном с k переменными и не содержит мономов с большим числом переменных. Считаем, не нарушая общности, что p содержит моном $x_1\ldots x_k$. Тогда

$$p(\underbrace{1,\ldots,1}_{k\text{ раз}},0,\ldots,0) = 1$$

и p не может представлять нулевую функцию. □

Коэффициенты многочлена Жегалкина можно найти по следующей формуле:

$$c_u = \sum_{x \leqslant u} f(x), \quad u \in \mathbb{F}_2^n.$$

Действительно,

$$\sum_{x \leqslant u} f(x) = \sum_{x \leqslant u} \sum_{v \leqslant x} c_v = \sum_{v \leqslant u} \sum_{x\colon v \leqslant x \leqslant u} c_v = c_u + \sum_{v < u} \underbrace{\sum_{x\colon v \leqslant x \leqslant u} c_v}_{\text{четное число слагаемых}} = c_u.$$

3.5.7. Алгебраическая степень

Алгебраической степенью функции f называется степень ее многочлена Жегалкина:

$$\deg(f) = \max_{u \in \mathbb{F}_2^n\colon c_u = 1} \mathrm{w}(u).$$

Если $f = 0$, то $\deg(f)$ полагается равной -1.

Если многочлен Жегалкина для f содержит моном $x_1 x_2 \dots x_m$, то функция $g(x_1, x_2, \dots, x_m) = f(x_1, x_2, \dots, x_m, 0, \dots, 0)$ принимает значение 1 нечетное число раз. Функция g называется *сужением* f на подпространство $L = \{(x_1, x_2, \dots, x_m, 0, \dots, 0)\}$ линейного пространства $\mathbb{F}_2^n$. Рассмотрим все подпространства, сужения f на которые принимают значения 1 нечетное число раз, и пусть r — максимальная размерность этих подпространств ($r = -1$, если f — нулевая функция). Тогда $\deg(f) = r$, что является альтернативным определением алгебраической степени.

Алгебраическая степень не изменяется при обратимом аффинном преобразовании переменных, т. е. является *аффинным инвариантом*: если $g \in \mathcal{F}_n$ получена из f по правилу

$$g(x) = f(xA + b), \quad A \in \mathrm{GL}_n(\mathbb{F}_2), \quad b \in \mathbb{F}_2^n,$$

то $\deg(f) = \deg(g)$. Действительно, при замене $x \mapsto y = xA + b$ степень многочлена $f(x)$ не увеличивается и $\deg(g) \leqslant \deg(f)$. Аналогично при замене $y \mapsto x = yA^{-1} - bA^{-1}$ не увеличивается степень $g(y)$ и $\deg(f) \leqslant \deg(g)$.

Криптографическое отображение $\sigma \in \mathcal{F}_{n,m}$ выбирают так, чтобы степени координатных функций были велики. Иногда требуют, чтобы величина

$$\min_{b \in \mathbb{F}_2^m \setminus \{\mathbf{0}\}} \deg(b \cdot \sigma)$$

также была большой. Эта величина называется *минимальной степенью* σ и обозначается через $\deg(\sigma)$.

Если $n \geqslant 2$ и f — уравновешенная, то $\deg(f) \leqslant n-1$. Действительно, если неравенство нарушается и $\deg(f) = n$, то f принимает значение 1 нечетное число раз и не может быть уравновешенной.

По теореме 3.1 все невырожденные линейные комбинации координатных функций подстановки $\sigma \in S(\mathbb{F}_2^n)$ уравновешенны. Поэтому при $n \geqslant 2$ справедлива оценка: $\deg(\sigma) \leqslant \leqslant n - 1$.

Пример 3.8 (степени модельных S-блоков). Для модельных S-блоков, описанных в примере 3.6, выполняется: $\deg(S_1) = 2$, $\deg(S_2) = 1$. В частности,

$$0010 \cdot S_1(x) = x_2x_3 + x_2x_4 + x_4 + 1,$$
$$1111 \cdot S_2(x) = x_3 + x_4 + 1. \qquad \square$$

3.5.8. Таблицы разностей

Пусть $\sigma \in \mathcal{F}_{n,m}$ и пусть на $\mathbb{F}_2^n$ и $\mathbb{F}_2^m$ заданы групповые операции $+$ и $+'$ соответственно. Событие $\{\sigma(x+a) = \sigma(x) +' b\}$ означает, что прообразам x и $x+a$ с *разностью* (сдвигом друг относительно друга) a соответствуют образы $\sigma(x)$ и $\sigma(x+a)$ с разностью b.

Таблица разностей для σ — это матрица размера $2^n \times 2^m$ с элементами

$$\mu_{ab}(\sigma) = \sum_{x \in \mathbb{F}_2^n} \mathsf{I}\{\sigma(x+a) = \sigma(x) +' b\}, \quad a \in \mathbb{F}_2^n, \quad b \in \mathbb{F}_2^m.$$

Элементам таблицы разностей можно придать вероятностный смысл. Пусть $x, \tilde{x}$ — случайные независимые векторы с равномерным распределением на $\mathbb{F}_2^n$. Тогда величина $2^{-n}\mu_{ab}(\sigma)$ есть вероятность перехода от пары $(x, \tilde{x})$ с разностью a к паре $(\sigma(x), \sigma(\tilde{x}))$ с разностью b:

$$\frac{1}{2^n}\mu_{ab}(\sigma) = \mathsf{P}\{\sigma(x+a) = \sigma(x) +' b\} = \mathsf{P}\{\sigma(\tilde{x}) = \sigma(x) +' b \mid \tilde{x} = x + a\}.$$

Понятно, как устроена первая строка таблицы разностей: $\mu_{\mathbf{00}}(\sigma) = 2^n$ и $\mu_{\mathbf{0}b}(\sigma) = 0$ для всех $b \in \mathbb{F}_2^m \setminus \{\mathbf{0}\}$. Вычеркнем эту строку и обозначим через $R(\sigma)$ максимальный элемент в оставшейся части таблицы:

$$R(\sigma) = \max_{\substack{a \in \mathbb{F}_2^n \\ a \neq \mathbf{0}}} \max_{b \in \mathbb{F}_2^m} \mu_{ab}(\sigma).$$

Криптографические отображения σ строят так, чтобы их разностные характеристики относительно определенных операций были невелики. Если $R(\sigma) \leqslant r$, то отображение σ называют *r-равномерным*.

В качестве $+$ и $+'$, как правило, выбирают операции $\oplus$ и $\boxplus$, которые часто используются при построении блочных криптосистем. При этом получают разностные характеристики четырех типов: $R_{\oplus\oplus}$, $R_{\oplus\boxplus}$, $R_{\boxplus\oplus}$ и $R_{\boxplus\boxplus}$. Здесь в индексе указывается сначала операция, выбранная в качестве $+$, а затем операция, выбранная в качестве $+'$.

Пусть $n = m$ и $\sigma \in S(\mathbb{F}_2^n)$. Построение r-равномерных подстановок с минимальным значением r является важной и не до конца решенной задачей. Ни одна из подстановок не может быть 1-равномерной. Действительно, если, например, $R_{\oplus\boxplus}(\sigma) = 1$, то уравнение $\sigma(x \oplus a) = \sigma(x) \boxplus b$ имеет ровно одно решение относительно x для любых ненулевых a и b. Это значит, что при фиксированном a отображение $x \mapsto \sigma(x \oplus a) \boxminus \sigma(x)$ является биекцией и, в частности, принимает нулевое значение, что невозможно. Подстановки с $r = 2$ известны при любом сочетании операций $\oplus$ и $\boxplus$, но для определенных n. Например, неизвестно, существуют ли при четных n подстановки σ, для которых $R_{\oplus\oplus}(\sigma) = 2$.

Пример 3.9 (таблицы разностей модельных *S*-блоков). Для модельных S-блоков, описанных в примере 3.6, выполняется:

$$R_{\oplus\oplus}(S_1) = R_{\oplus\boxplus}(S_1) = 8, \quad R_{\boxplus\oplus}(S_1) = 4, \quad R_{\boxplus\boxplus}(S_1) = 2;$$

$$R_{\oplus\oplus}(S_2) = R_{\boxplus\oplus}(S_2) = 16, \quad R_{\oplus\boxplus}(S_2) = 4, \quad R_{\boxplus\boxplus}(S_2) = 2.$$

Зафиксируем следующий факт, который нам потребуется в дальнейшем:

$$\mathsf{P}\{S_1(x \oplus 1000) = S_1(x) \oplus 0001\} = \frac{1}{4}.$$

3.5.9. Инверсные S-блоки

В 1993 году К. Нюберг и, независимо от нее, Т. Бес и К. Динг предложили строить биективные S-блоки, используя обращение (инверсию) в поле $\mathbb{F}_{2^n}$, $n \geqslant 2$. *Инверсный S-блок* $s \in S(\mathbb{F}_{2^n})$ определяется следующим образом:

$$s(x) = x^{2^n-2} = \begin{cases} x^{-1}, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

Элементы $\mathbb{F}_{2^n}$, как обычно, отождествляются с двоичными векторами или словами, при этом действие s естественным образом переносится на $\mathbb{F}_2^n$ или $\{0,1\}^n$.

Инверсные S-блоки обладают рядом замечательных свойств, и в связи с этим часто используются в блочных криптосистемах. Например, S-блок `AES` — это модификация инверсного S-блока при $n = 8$. Модификация состоит в применении дополнительных обратимых аффинных преобразований к прообразам и образам. Эти преобразования не изменяют криптографические характеристики $R_{\oplus\oplus}(s)$, $\deg(s)$ и $\mathrm{nl}(s)$, которые мы рассмотрим ниже.

Теорема 3.3. *Пусть $s \in S(\mathbb{F}_{2^n})$ — инверсный S-блок. Тогда $R_{\oplus\oplus}(s) = 4$ при четном n и $R_{\oplus\oplus}(s) = 2$ при нечетном n.*

Доказательство. Пусть $a, b \in \mathbb{F}_{2^n}$ и $a \neq 0$. Рассмотрим уравнение

$$s(x + a) = s(x) + b \tag{3.1}$$

относительно x. Характеристика $R_{\oplus\oplus}(s)$ есть максимальное число решений данного уравнения при всевозможных фиксированных a, b. Подсчитаем максимальное число решений. Для этого рассмотрим два случая.

1. Пусть $b \neq a^{-1}$. Тогда $x = 0$ и $x = a$ не могут являться решениями (3.1). Считая, что $x \notin \{0, a\}$, можно записать (3.1) в следующем виде:

$$\frac{1}{x+a} + \frac{1}{x} + b = \frac{bx^2 + abx + a}{(x+a)x} = 0.$$

Последнее равенство равносильно уравнению $bx^2 + abx + a = 0$, которое имеет не более двух решений в поле $\mathbb{F}_{2^n}$.

2. Пусть $b = a^{-1}$. В этом случае решениями (3.1) являются $x = 0$ и $x = a$, а также, дополнительно, решения уравнения

$$a^{-1}x^2 + x + a = 0.$$

Умножим обе его части на a, заменим x на ay и получим уравнение

$$y^2 + y + 1 = 0,$$

все корни которого лежат в поле $\mathbb{F}_{2^2}$. (Действительно, корнями являются элементы $\lambda, \lambda + 1 \in \mathbb{F}_2[\lambda]/(\lambda^2 + \lambda + 1) \cong \mathbb{F}_{2^2}$.)

При четном n поле $\mathbb{F}_{2^2}$ является подполем $\mathbb{F}_{2^n}$, а при нечетном n — нет. Таким образом, у исходного уравнения (3.1) имеется не более 4 решений при четном n и не более 2 решений при нечетном, причем верхние границы достижимы. □

Теорема 3.4. *Если $s \in S(\mathbb{F}_{2^n})$ — инверсный S-блок, то $\deg(s) = n - 1$.*

Доказательство. 1. Рассмотрим функцию $f \in \mathcal{F}_n$, которая определяется по правилу $f(x) = \mathrm{Tr}(bx^d)$, $b \in \mathbb{F}_{2^n}^*$, $d \in \{1, 2, \ldots, 2^n - 1\}$. Пусть число d представлено двоичным словом и $r = \mathrm{w}(d)$ — вес Хэмминга этого слова. Докажем, что $\deg(f) \leqslant r$.

Пусть $d = 2^{k_1}+2^{k_2}+\ldots+2^{k_r}$, где $0 \leqslant k_1 < k_2 < \ldots < k_r \leqslant n-1$, и пусть $\alpha_1, \alpha_2, \ldots, \alpha_n$ — базис $\mathbb{F}_{2^n}$ над $\mathbb{F}_2$, который отвечает за представление элементов $\mathbb{F}_{2^n}$ векторами. Тогда

$$f(x) = \operatorname{Tr}\left(b\left(\sum_{i=1}^{n} x_i\alpha_i\right)^d\right) = \operatorname{Tr}\left(b\prod_{j=1}^{r}\left(\sum_{i=1}^{n} x_i\alpha_i\right)^{2^{k_j}}\right) =$$

$$= \operatorname{Tr}\left(b\prod_{j=1}^{r}\left(\sum_{i=1}^{n} x_i\alpha_i^{2^{k_j}}\right)\right) = \sum_{1\leqslant i_1,i_2,\ldots,i_r\leqslant n} x_{i_1}x_{i_2}\ldots x_{i_r}\operatorname{Tr}\left(b\prod_{j=1}^{r}\alpha_{i_j}^{2^{k_j}}\right)$$

и, следовательно, $\deg(f) \leqslant r$.

2. Любую функцию $g \in \mathcal{F}_n$ можно представить в виде $g(x) = \operatorname{Tr}(P(x))$, где $P(x) \in \mathbb{F}_{2^n}[x]$, $\deg(P) \leqslant 2^n - 1$. Действительно, можно подобрать преобразование $\sigma\colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ такое, что $g(x) = \operatorname{Tr}(\sigma(x))$, а затем задать действие σ интерполяционным многочленом Лагранжа P.

3. Пусть $\deg(g) = n-1$. Многочлен $P(x)$ представляет собой сумму членов bx^d, каждому из которых соответствует функция $\operatorname{Tr}(bx^d)$ степени $\leqslant \mathrm{w}(d)$. Найдется член cx^e с $\mathrm{w}(e) = n-1$, которому соответствует функция $\operatorname{Tr}(cx^e)$ степени $n-1$. Действительно, противное означает, что в многочлене Жегалкина для g мономы степени $n-1$ индуцируются членом bx^{2^n-1}. Но этому члену соответствует функция

$$\operatorname{Tr}\left(bx^{2^n-1}\right) = \begin{cases} 0, & x = 0, \\ \operatorname{Tr}(b), & x \neq 0, \end{cases}$$

которая либо нулевая, либо имеет степень n. В обоих случаях получаем противоречие.

4. Показатель e представляется словом $11\ldots1011\ldots1$ из $n-1$ единичных и одного нулевого символов. Пусть нулевой символ находится в k-й слева позиции. Умножение e на 2^k по модулю 2^n-1 состоит в циклическом сдвиге слова-представления на k позиций влево. После умножения будет получено слово $11\ldots10$, которому соответствует число 2^n-2. Поэтому функция

$$h(x) = \operatorname{Tr}\left(c^{2^k}x^{2^n-2}\right) = \operatorname{Tr}\left((cx^e)^{2^k}\right) = \operatorname{Tr}(cx^e)$$

имеет степень $n-1$.

5. Невырожденные линейные комбинации координатных функций s имеют вид $s_b(x) = \operatorname{Tr}(bx^{2^n-2})$, $b \in \mathbb{F}_{2^n}^*$. Заменив в любой из таких функций x на $bc^{-2^k}y$, получаем функцию $h(y)$. Замена переменных является обратимой линейной, и поэтому $\deg(s_b) = \deg(h) = n-1$. Следовательно, $\deg(s) = \min_b \deg(s_b) = n-1$. □

Следующая теорема, которую мы приводим без доказательства, касается нелинейности инверсного S-блока. Теорема является следствием оценок для сумм Клостермана, полученных Ж. Лашо и Ж. Волфманом в 1990 году.

Теорема 3.5. *Пусть $s \in S(\mathbb{F}_{2^n})$ — инверсный S-блок. Если n — четное, то $\operatorname{nl}(s) = 2^{n-1} - 2^{n/2}$. Если n — нечетное, то $\operatorname{nl}(s)$ есть минимальное четное $\geqslant 2^{n-1} - 2^{n/2}$.*

3.5.10. Экспоненциальные S-блоки

Выберем примитивный элемент $\alpha \in \mathbb{F}_{2^n}$ и построим отображение $s\colon \mathbb{Z}_{2^n} \to \mathbb{F}_{2^n}$,

$$s(x) = \begin{cases} 0, & x = 0, \\ \alpha^x, & x \neq 0. \end{cases}$$

Поскольку $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2^n-1}$ суть все ненулевые элементы $\mathbb{F}_{2^n}$, s является биекцией. Заменяя прообразы и образы s векторами из $\mathbb{F}_2^n$, получаем *экспоненциальный S-блок* $s \in S(\mathbb{F}_2^n)$.

Пусть $f(\lambda) \in \mathbb{F}_2[\lambda]$ — минимальный многочлен элемента α. Этот многочлен имеет степень n и является примитивным. Через $L(f)$ обозначим множество всех двоичных последовательностей длиной 2^n, устроенных следующим образом: первый элемент каждой последовательности — 0, остальные элементы — отрезок линейной рекуррентной последовательности с характеристическим многочленом $f(\lambda)$. Используя свойства линейных рекуррентных последовательностей, легко проверить, что $L(f)$ – векторное пространство размерностью n над полем $\mathbb{F}_2$, и всякая ненулевая последовательность $L(f)$ является уравновешенной — содержит одинаковое количество 0 и 1.

Невырожденные линейные комбинации координатных функций s имеют вид

$$s_b(x) = \begin{cases} 0, & x = 0, \\ \operatorname{Tr}(b\alpha^x), & x \neq 0, \end{cases}$$

где $b \in \mathbb{F}_{2^n}^*$, а x интерпретируется как вектор в левой части и как число — в правой. Пусть $s_{b,0}, s_{b,1}, \ldots, s_{b,2^n-1}$ — таблица истинности s_b, т. е. последовательность значений функции на упорядоченных по возрастанию аргументах x (как числах). Эта последовательность является элементом $L(f)$. В частности, таблица истинности уравновешенна, откуда с использованием теоремы 3.1 получаем еще одно доказательство биективности s.

Альтернативный способ построения подстановки s состоит в выборе в качестве таблиц истинности ее координатных функций некоторого базиса векторного пространства $L(f)$. Имеется $\varphi(2^n-1)/n$ различных примитивных многочленов степени n над полем $\mathbb{F}_2$ (здесь φ — функция Эйлера) и, следовательно, столько же различных множеств $L(f)$. Базис $L(f)$ можно выбрать $(2^n-1)(2^n-2)\ldots(2^n-2^{n-1})$ способами. Таким образом, имеется

$$\frac{\varphi(2^n-1)}{n}(2^n-1)(2^n-2)\ldots(2^n-2^{n-1})$$

различных экспоненциальных S-блоков, действующих на $\mathbb{F}_2^n$.

Экспоненциальные S-блоки обладают достаточно высокой нелинейностью, а их характеристики $R_{\boxplus\oplus}(s)$ и $\deg(s)$, как показывают следующие теоремы, близки к оптимальным.

Теорема 3.6. *Пусть $s \in S(\mathbb{F}_2^n)$ — экспоненциальный S-блок, построенный с помощью примитивного элемента α. Если минимальный многочлен $f(\lambda)$ этого элемента не делит ни один из многочленов вида*

$$\lambda^{2^{n-1}} + \lambda^t + 1, \quad t = 1, \ldots, 2^{n-1} - 1,$$

то $R_{\boxplus\oplus}(s) \leqslant 3$. В противном случае $R_{\boxplus\oplus}(s) = 4$.

Доказательство. Пусть a — ненулевой вектор $\mathbb{F}_2^n$ и τ — число, которое соответствует этому вектору. Множество значений $s(x) \oplus s(x \boxplus a)$, $x \in \mathbb{F}_2^n$, является объединением $\{\alpha^\tau\} \cup \{\alpha^{2^n-\tau}\} \cup A \cup B$, где

$$A = \{\alpha^t \oplus \alpha^{t+\tau} \colon t = 1, \ldots, 2^n - 1 - \tau\}, \quad B = \{\alpha^t \oplus \alpha^{2^n-\tau+t} \colon t = 1, \ldots, \tau - 1\}.$$

Все элементы, определяющие каждое из множеств A и B различны, следовательно, среди значений $s(x) \oplus s(x \boxplus a)$ не может быть более 4 одинаковых. Это означает, что $R_{\boxplus\oplus}(s) \leqslant 4$.

Более того, $R_{\boxplus\oplus}(s) = 4$ тогда и только тогда, когда для $\tau = 2^{n-1}$ (в этом случае $\alpha^\tau = \alpha^{2^n-\tau}$) и некоторого $t = 1, \ldots, 2^{n-1} - 1$ выполняется равенство

$$\alpha^\tau = \alpha^t \oplus \alpha^{t+\tau}.$$

Но это означает, что минимальный многочлен $f(\lambda)$ делит некоторый многочлен, указанный в формулировке теоремы. □

Теорема 3.7. *Если $s \in S(\mathbb{F}_2^n)$ — экспоненциальный S-блок, то*

$$\deg(s) \geqslant n - \lceil \log_2(n+1) \rceil.$$

Доказательство. Обозначим $\alpha_i = \alpha^{2^{i-1}}$, $i = 1, 2, \ldots$. Пусть s_b — невырожденная линейная комбинация координатных функций s, соответствующая элементу $b \in \mathbb{F}_{2^n}^*$:

$$s_b(x_1, \ldots, x_n) = \mathrm{Tr}\left(b \prod_{i=1}^{n} \alpha_i^{x_i}\right) + \mathrm{Tr}(b) \prod_{i=1}^{n} (1 + x_i) = \begin{cases} 0, & x_1 = \ldots = x_n = 0, \\ \mathrm{Tr}(b\alpha^x) & \text{в противном случае.} \end{cases}$$

Введем оператор Δ_j взятия разности по переменной x_j:

$$\Delta_j s_b(x_1, \ldots, x_n) = s_b(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n) + s_b(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) =$$

$$= \mathrm{Tr}\left(b(1 + \alpha_j) \prod_{\substack{1 \leqslant i \leqslant n \\ i \neq j}} \alpha_i^{x_i}\right) + \mathrm{Tr}(b) \prod_{\substack{1 \leqslant i \leqslant n \\ i \neq j}} (1 + x_i). \tag{3.2}$$

Многочлен $\Delta_j s_b$, домноженный на x_j, входит в качестве слагаемого в многочлен s_b. Если $\Delta_j s_b$ ненулевой, то

$$\deg(s_b) \geqslant \deg(\Delta_j s_b) + 1.$$

Пусть $g(x_1, \ldots, x_k)$ — функция, полученная после применения к $\sigma(x_1, \ldots, x_n)$ последовательно операторов $\Delta_{k+1}, \ldots, \Delta_n$, $1 \leqslant k < n$. Используя (3.2), получаем

$$g(x_1, \ldots, x_k) = \mathrm{Tr}\left(bc \prod_{i=1}^{k} \alpha_i^{x_i}\right) + \mathrm{Tr}(b) \prod_{i=1}^{k} (1 + x_i),$$

где

$$c = \prod_{j=k+1}^{n} (1 + \alpha_j) = \sum_{t=0}^{2^{n-k}-1} \alpha_{k+1}^t = (1 + \alpha)(1 + \alpha_{k+1})^{-1} \neq 0.$$

После удаления в таблице истинности функции g первого элемента $\mathrm{Tr}(bc+b)$ остается отрезок

$$\mathrm{Tr}(bc\alpha), \mathrm{Tr}(bc\alpha^2), \ldots, \mathrm{Tr}(bc\alpha^{2^k-1})$$

ненулевой линейной рекуррентной последовательности с примитивным характеристическим многочленом степени n. Если $2^k - 1 \geqslant n$, то данный отрезок не может быть нулевым. Поэтому при $k = \lceil \log_2(n+1) \rceil$

$$\deg(s_b) \geqslant \deg(g) + n - k \geqslant n - k,$$

откуда и следует требуемый результат. □

Следующая теорема дает критерий выбора α, при котором степени всех ненулевых координатных функций s достигают максимального значения.

Теорема 3.8. *Пусть $s \in S(\mathbb{F}_{2^n})$ — экспоненциальный S-блок, построенный с помощью примитивного элемента α. Тогда $\deg(s) = n - 1$ только и только если элементы*

$$a = \alpha(1 + \alpha)^{-1}, a^2, \ldots, a^{2^{n-1}}$$

образуют базис $\mathbb{F}_{2^n}$ над $\mathbb{F}_2$.

Доказательство. Используем обозначения из доказательства предыдущей теоремы. Рассмотрим функции

$$\begin{aligned} g_j(x_j) &= \Delta_n \dots \Delta_{j+1}\Delta_{j-1} \dots \Delta_1 s_b(x_1, \dots, x_n) = \\ &= \mathrm{Tr}\left(b\alpha_j^{x_j} \prod_{\substack{1 \leqslant i \leqslant n \\ i \neq j}} (1+\alpha_i) \right) + \mathrm{Tr}(b)(1+x_j) = \\ &= \mathrm{Tr}(b\alpha_j^{x_j}(1+\alpha_j)^{-1}) + \mathrm{Tr}(b)(1+x_j). \end{aligned}$$

Последнее равенство справедливо в силу того, что

$$\prod_{i=1}^{n}(1+\alpha_i) = \sum_{t=0}^{2^n-1} \alpha^t = 1 + \sum_{c \in \mathbb{F}_{2^n}} c = 1.$$

Для функции g_j выполняется:

$$g_j(0) = \mathrm{Tr}(b((1+\alpha_j)^{-1}+1)) = \mathrm{Tr}(b\alpha_j(1+\alpha_j)^{-1}) = g_j(1).$$

Обозначим $a_j = \alpha_j(1+\alpha_j)^{-1} = a^{2^j}$. Теперь $\deg(s_b) = n-1$ тогда и только тогда, когда $g_j(0) = \mathrm{Tr}(ba_j) \neq 0$ для некоторого j, $1 \leqslant j \leqslant n$. Следовательно, $\deg(s) = n-1$ тогда и только тогда, когда ядро гомоморфизма $\mathbb{F}_{2^n} \to \mathbb{F}_2^n$, $b \mapsto (\mathrm{Tr}(ba_1), \dots, \mathrm{Tr}(ba_n))$ состоит из единственного (нулевого) элемента. Но последнее означает, что $a_1, \dots, a_n$ является базисом $\mathbb{F}_{2^n}$ над $\mathbb{F}_2$. □

3.6. Криптосистемы подстановки-перестановки

Вернемся к блочно-итерационным криптосистемам и обсудим строение тактовых подстановок $\Sigma_\kappa \in S(\{0,1\}^{n_b})$, $\kappa \in K$. В *криптосистемах подстановки-перестановки* множество тактовых ключей K совпадает с $\{0,1\}^{n_b}$, длина блока n_b является произведением rm, где r и m — натуральные числа, $r \geqslant 2$. Используются S-блоки $S_1, \dots, S_r \in S(\{0,1\}^m)$ и преобразование $P \in S(\{0,1\}^{n_b})$. Действие P состоит в обратимой перестановке символов слов-прообразов:

$$P(x_1x_2 \dots x_{n_b}) = x_{\pi(1)}x_{\pi(2)} \dots x_{\pi(n_b)}, \quad \pi \in S(\{1,2,\dots,n_b\}).$$

В соответствии с английскими терминами Substitution (подстановка) и Permutation (перестановка) криптосистемы подстановки-перестановки часто называют *SP-криптосистемами.*

Образы $\Sigma_\kappa(X)$ определяются в три этапа (см. рис. 3.2).

1. Прообраз X суммируется с тактовым ключом κ по правилу $\oplus$.
2. К последовательным m-фрагментам суммы применяются S-блоки $S_1, \dots, S_r$.
3. К объединению выходов S-блоков применяется преобразование перестановки P.

Действие Σ_κ, а также обратной подстановки Σ_κ^{-1}, можно задать алгоритмически.

Алгоритм Применение Σ_κ

Вход: $X, \kappa \in \{0,1\}^{n_b}$ $(n_b = rm)$.
Выход: $Y = \Sigma_\kappa(X)$.
Шаги:
1. $Y \leftarrow X \oplus \kappa$.
2. Представить Y в виде $Y_1 \parallel \ldots \parallel Y_r$.
3. Для $i = 1, \ldots, r$: $Y_i \leftarrow S_i(Y_i)$.
4. $Y \leftarrow P(Y)$.
5. Возвратить Y.

Алгоритм Применение Σ_κ^{-1}

Вход: $Y, \kappa \in \{0,1\}^{n_b}$ $(n_b = rm)$.
Выход: $X = \Sigma_\kappa^{-1}(Y)$.
Шаги:
1. $X \leftarrow P^{-1}(Y)$.
2. Представить X в виде $X_1 \parallel \ldots \parallel X_r$.
3. Для $i = 1, \ldots, r$: $X_i \leftarrow S_i^{-1}(X_i)$.
4. $X \leftarrow X \oplus \kappa$.
5. Возвратить X.

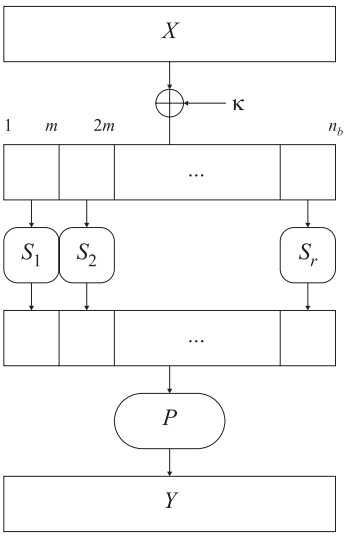


Рис. 3.2. Тактовая подстановка SP-криптосистемы

Современные подходы к построению SP-криптосистем заключаются в замене перестановки P на преобразование A, состоящее в умножении слова-как-вектора на обратимую матрицу над полем $\mathbb{F}_2$ (не обязательно перестановочную). Например в криптосистеме `Serpent` преобразование A реализовано с помощью многократных сложений, сдвигов и перестановок частей исходного вектора. Криптосистемы, полученные заменой P на A, принято обозначать аббревиатурой SA.

Частным случаем SA-криптосистем являются *SQUARE-криптосистемы*. Здесь m-фрагменты X записываются в ячейки квадратной матрицы (отсюда SQUARE — квадрат, англ.). После сложения с ключом и подстановки на S-блоках выполняются линейные преобразования сначала всех строк матрицы (по отдельности), а затем всех столбцов.

3.7. Криптосистема AES

Криптосистема `AES` была разработана бельгийскими криптографами В. Рэйменом и Й. Дэменом в рамках проводимого в США конкурса на разработку стандарта шифрования XXI в. Разработка Рэймена и Дэмена, которая первоначально называлась `Rijndael`, победила в конкурсе и была стандартизирована в 2002 году. Сегодня это самая распространенная блочная криптосистема в мире.

Криптосистема построена по схеме SQUARE, длина блока $n_b = 128$, число тактов $d \in \{10, 12, 14\}$, множество ключей $\Theta \in \{\{0,1\}^{128}, \{0,1\}^{192}, \{0,1\}^{256}\}$, множество тактовых ключей $K = \{0,1\}^{128}$.

В `AES` октеты отождествляются с векторами $\mathbb{F}_2^8$, элементами $\mathbb{F}_{2^8}$ и числами $\mathbb{Z}_{2^8}$ так, как это описано в примере 3.6. Для обработки блоков данных используется матрица размером 4×4. В ячейки матрицы записываются октеты x_{ij}, $0 \leqslant i, j \leqslant 3$. К матрице применяются преобразования `AddRoundKey`, `SubBytes`, `ShiftRows`, `MixColumns`.

Действие преобразования `AddRoundKey` определяется тактовым ключом и состоит в сложении октетов матрицы с соответствующими октетами тактового ключа. Сложение выполняется по правилу $\oplus$.

Преобразование `SubBytes` состоит в одновременной замене всех октетов x_{ij} на $s(x_{ij})$. Здесь $s \in S(\{0,1\}^8)$ — модифицированный инверсный S-блок (см. 3.5.9).

Под действием `ShiftRows` строка матрицы с номером $i = 0, 1, 2, 3$ сдвигается циклически влево на i позиций.

Самым сложным алгоритмически является преобразование `MixColumns`. Каждый из столбцов $(b_0, b_1, b_2, b_3)^{\mathrm{T}}$ матрицы связывается с многочленом

$$b(\lambda) = b_3\lambda^3 + b_2\lambda^2 + b_1\lambda + b_0 \in \mathbb{F}_{2^8}[\lambda],$$

который умножается на фиксированный многочлен

$$a(\lambda) = \mathtt{03}_{16}\lambda^3 + \mathtt{01}_{16}\lambda^2 + \mathtt{01}_{16}\lambda + \mathtt{02}_{16} \in \mathbb{F}_{2^8}[\lambda].$$

Затем полученное произведение делится на многочлен $\lambda^4 + 1$:

$$a(\lambda)b(\lambda) = g(\lambda)(\lambda^4 + 1) + b'(\lambda), \quad g, b' \in \mathbb{F}_{2^8}[\lambda], \quad \deg b' < 4.$$

Коэффициенты остатка $b'(\lambda)$ определяют новый столбец матрицы — результат преобразования `MixColumns`.

Важно, что $a(\lambda)$ взаимно прост с $\lambda^4 + 1$: имеется многочлен $a^{-1}(\lambda) \in \mathbb{F}_{2^8}[\lambda]$ такой, что $a(\lambda)a^{-1}(\lambda) \equiv 1 \pmod{\lambda^4 + 1}$. Поэтому

$$b'(\lambda)a^{-1}(\lambda) = b(\lambda)a(\lambda)a^{-1}(\lambda) \equiv 1 \pmod{\lambda^4 + 1},$$

т. е. обратное преобразование определяется умножением на $a^{-1}(\lambda)$.

При зашифровании выполняются следующие преобразования:

Σ_{1,κ_1}	`AddRoundKey`
	`SubBytes`
	`ShiftRows`
	`MixColumns`

.....	

$\Sigma_{d-1,\kappa_{d-1}}$	`AddRoundKey`
	`SubBytes`
	`ShiftRows`
	`MixColumns`

Σ_{d,κ_d}	`AddRoundKey`
	`SubBytes`
	`ShiftRows`

Отбеливание	`AddRoundKey`

При расшифровании преобразования меняются на композиционно обратные и применяются в обратном порядке.

3.8. τ-инволютивные подстановки

Как мы только что видели, при расшифровании с помощью SP- или SA-криптосистем итерационные преобразования должны применяться в обратном порядке, прямые преобразования S_i, P или A должны заменяться на композиционно обратные. Это оказывается не всегда удобным. Например, при аппаратной реализации криптосистемы приходится хранить в устройстве как код программы зашифрования, так и код расшифрования; как таблицы S_i, так и таблицы S_i^{-1}.

Покажем, как организовать переключение между зашифрованием и расшифрованием, измененяя только порядок следования тактовых ключей.

Определение 3.2. *Пусть* τ *и* σ *— подстановки, которые действуют на одном и том же множестве. Подстановка* σ *называется* τ*-инволютивной, если* $\sigma\tau\sigma = \tau$.

Вместо «id-инволютивная» будем говорить просто «инволютивная», как это принято в алгебре.

Теорема 3.9. *Пусть преобразования зашифрования блочно-итерационной криптосистемы* F *имеют вид*

$$F_\theta = \tau\Sigma_{\kappa_d} \dots \Sigma_{\kappa_2}\Sigma_{\kappa_1},$$

где $\tau \in S(\{0,1\}^{n_b})$ *— инволютивна, а тактовые подстановки* $\Sigma_\kappa \in S(\{0,1\}^{n_b})$ *—* τ*-инволютивны при любом* $\kappa \in K$. *Тогда*

$$F_\theta^{-1} = \tau\Sigma_{\kappa_1}\Sigma_{\kappa_2} \dots \Sigma_{\kappa_d}.$$

Доказательство. Имеем

$$\begin{aligned} F_\theta\tau\Sigma_{\kappa_1}\Sigma_{\kappa_2} \dots \Sigma_{\kappa_d} &= \tau\Sigma_{\kappa_d} \dots \Sigma_{\kappa_2}(\Sigma_{\kappa_1}\tau\Sigma_{\kappa_1})\Sigma_{\kappa_2} \dots \Sigma_{\kappa_d} = \\ &= \tau\Sigma_{\kappa_d} \dots (\Sigma_{\kappa_2}\tau\Sigma_{\kappa_2}) \dots \Sigma_{\kappa_d} = \dots = \\ &= \tau\tau = id, \end{aligned}$$

что и требовалось доказать. □

Для реализации подстановок F_θ и F_θ^{-1}, описанных в теореме, требуется использовать семейство τ-инволютивных тактовых подстановок. Рассмотрим две распространенные конструкции таких семейств. Будем считать, что $n_b = 2m$ — четное число, и использовать преобразования

$$f_\kappa \colon \{0,1\}^m \to \{0,1\}^m, \quad \kappa \in K,$$

которые назовем *тактовыми функциями*. Тактовые функции не обязательно биективны.

Подстановки Фейстеля. При разработке криптосистемы `DES` было решено использовать следующие подстановки, предложенные ранее Х. Фейстелем (рис. 3.3, а):

$$\Sigma_\kappa(X_1 \parallel X_2) = X_2 \parallel (X_1 \oplus f_\kappa(X_2)), \quad X_i \in \{0,1\}^m.$$

Пусть преобразование τ состоит в перестановке половинок слов из $\{0,1\}^{2m}$: $\tau(X_1 \parallel X_2) = X_2 \parallel X_1$. Тогда

$$\begin{aligned} \Sigma_\kappa\tau\Sigma_\kappa(X_1 \parallel X_2) &= \Sigma_\kappa\tau(X_2, X_1 \oplus f_\kappa(X_2)) = \\ &= \Sigma_\kappa(X_1 \oplus f_\kappa(X_2) \parallel X_2) = \\ &= X_2 \parallel X_1 = \tau(X_1 \parallel X_2). \end{aligned}$$

Следовательно, Σ_κ является τ-инволютивной подстановкой.

Подстановки Лай — Мэсси. При разработке криптосистемы `IDEA` С. Лай и Дж. Мэсси использовали следующие подстановки (рис. 3.3, б):

$$\Sigma_\kappa(X_1 \parallel X_2) = (X_1 \oplus f_\kappa(X_1 \oplus X_2) \parallel X_2 \oplus f_\kappa(X_1 \oplus X_2)), \quad X_i \in \{0,1\}^m.$$

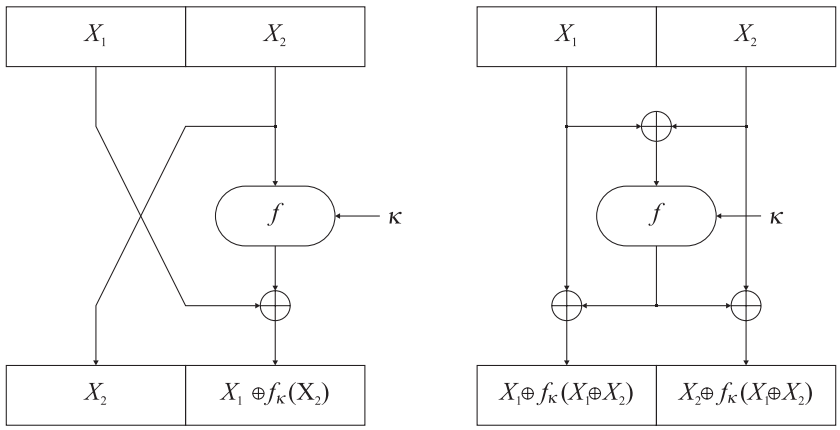
Подстановка Σ_κ является id-инволютивной (т. е. просто инволютивной) при любом выборе тактовой функции f_κ.

Вместо $\oplus$ можно использовать другие операции, например, перейти к подстановке

$$\Sigma'_\kappa(X_1 \parallel X_2) = (X_1 \boxplus f_\kappa(X_1 \boxplus X_2) \parallel X_2 \boxminus f_\kappa(X_1 \boxplus X_2)),$$

которая также является инволютивной.

Подстановка Σ_κ сохраняет сумму половинок: если $\Sigma(X_1 \parallel X_2) = Y_1 \parallel Y_2$, $Y_i \in \{0,1\}^m$, то $Y_1 \oplus Y_2 = X_1 \oplus X_2$. Поэтому Σ_κ целесообразно использовать в совокупности с другими преобразованиями, как это сделано в криптосистемах `IDEA` и `Belt`.



а *б*

Рис. 3.3. τ-инволютивные подстановки

3.9. Криптосистемы Фейстеля

В криптосистемах, основанных на подстановках Фейстеля, шифрование выполняется с помощью следующего алгоритма.

Алгоритм Шифрование в криптосистемах Фейстеля

Вход: $X \in \{0,1\}^{2m}$, $\kappa_1, \ldots, \kappa_d \in K$.
Выход: $Y \in \{0,1\}^{2m}$.
Шаги:
1. $Y_1 \parallel Y_2 \leftarrow X$.
2. Для $i = 1, \ldots, d$: $Y_1 \parallel Y_2 \leftarrow Y_2 \parallel (Y_1 \oplus f_{\kappa_i}(Y_2))$.
3. Возвратить $Y_2 \parallel Y_1$.

При зашифровании на вход алгоритма подаются тактовые ключи $(\kappa_1, \ldots, \kappa_d) = \mathtt{KS}(\theta)$, при расшифровании — ключи $(\kappa_d, \ldots, \kappa_1)$. Для окончательного определения криптосистемы Фейстеля требуется уточнить расписание ключей `KS` и действие тактовых функций f_κ.

В таблице 3.2 приводятся характеристики некоторых известных криптосистем Фейстеля. У всех криптосистем длина блока $n_b = 64$. Тактовые функции $f_\kappa\colon \{0,1\}^{32} \to \{0,1\}^{32}$ криптосистем схематически представлены на рис. 3.4. На схеме, соответствующей `DES`, отображение $E\colon \{0,1\}^{32} \to \{0,1\}^{48}$ состоит в повторе и перестановке координат слова-прообраза, P — преобразование перестановки.

Таблица 3.2. Криптосистемы Фейстеля

Криптосистема	n_b	d	Θ	K	S-блоки
DES (стандарт США в 1979–2005 гг.)	64	16	$\{0,1\}^{56}$	$\{0,1\}^{48}$	$S_i\colon \{0,1\}^6 \to \{0,1\}^4$ (постоянные)
GOST (стандарт СССР ГОСТ 28147, принят в 1989 г.)	64	32	$\{0,1\}^{256}$	$\{0,1\}^{32}$	$S_i\colon \{0,1\}^4 \to \{0,1\}^4$ (долговременный ключ)
Blowfish (США)	64	16	$\{0,1\}^{40\div448}$	$\{0,1\}^{32}$	$S_i\colon \{0,1\}^8 \to \{0,1\}^{32}$ (рассчитываются по θ)
CAST (Канада)	64	16	$\{0,1\}^{64}$	$\{0,1\}^{16}$	$S_i\colon \{0,1\}^8 \to \{0,1\}^{32}$ (постоянные)

Пример 3.10 (модельная криптосистема G). Методы криптоанализа мы будем иллюстрировать примерами с модельной криптосистемой Фейстеля, названной нами G (рис. 3.5). В этой криптосистеме $n_b = 16$, $d = 8$, $\Theta = \{0,1\}^{32}$, $K = \{0,1\}^8$. Алгоритм KS ставит в соответствие ключу $\theta = \theta_1 \parallel \theta_2 \parallel \theta_3 \parallel \theta_4$, $\theta_i \in \{0,1\}^8$, последовательность тактовых ключей

$$\kappa_1 = \theta_1, \quad \kappa_2 = \theta_2, \quad \kappa_3 = \theta_3, \quad \kappa_4 = \theta_4, \quad \kappa_5 = \theta_1, \quad \kappa_6 = \theta_2, \quad \kappa_7 = \theta_3, \quad \kappa_8 = \theta_4.$$

В тактовой функции $f_\kappa\colon \{0,1\}^8 \to \{0,1\}^8$ используются модельные S-блоки S_1 и S_2, описанные в примере 3.6. Тактовые функции действуют следующим образом:

$$f_\kappa(x) = (S_1(z_1) \parallel S_2(z_2)) \lll 3, \quad z_i \in \{0,1\}^4, \quad z_1 \parallel z_2 = x \oplus \kappa.$$

3.10. Криптосистема Belt

Криптосистема Belt, разработанная в 2001 году, определена в государственном стандарте Республики Беларусь СТБ 34.101.31-2011 «Информационные технологии и безопасность. Криптографические алгоритмы шифрования и контроля целостности». Этот стандарт был введен как предварительный в 2007 году и получил статус окончательного 4 годами позже.

Belt не является ни SP-криптосистемой, ни криптосистемой Фейстеля, хотя и содержит конструктивные элементы этих систем. Длина блока $n_b = 128$, число тактов $d = 8$, множество ключей $\Theta = \{0,1\}^{256}$.

В Belt используются операции $\oplus$, $\boxplus$, $\boxminus$, циклические сдвиги, подстановки на S-блоках. При представлении чисел двоичными словами применяются соглашения «от младших к старшим» (little endian), действующие для большинства современных микропроцессоров. Согласно этим соглашениям, первый октет слова представляет младший байт числа, последний октет — старший. Например, слову

10110001 10010100 10111010 11001000

соответствуют байты $\mathtt{B1}_{16}$, 94_{16}, $\mathtt{BA}_{16}$, $\mathtt{C8}_{16}$ и число $\mathtt{C8BA94B1}_{16}$ (в шестнадцатеричной записи). Удобно считать, что при переходе от слов к числам и назад октеты слов неявно разворачиваются. Неявный разворот октетов выполняется до и после операций $\boxplus$, $\boxminus$. Операция циклического сдвига слова на r позиций влево, которая обозначается через $\mathtt{RotHi}^r$, также выполняется с разворотом октетов. Например,

$$\mathtt{RotHi}^5(\mathtt{C8BA94B1}_{16}) = \mathtt{17529639}_{16} \sim 00111001\ 10010110\ 01010010\ 00010111.$$

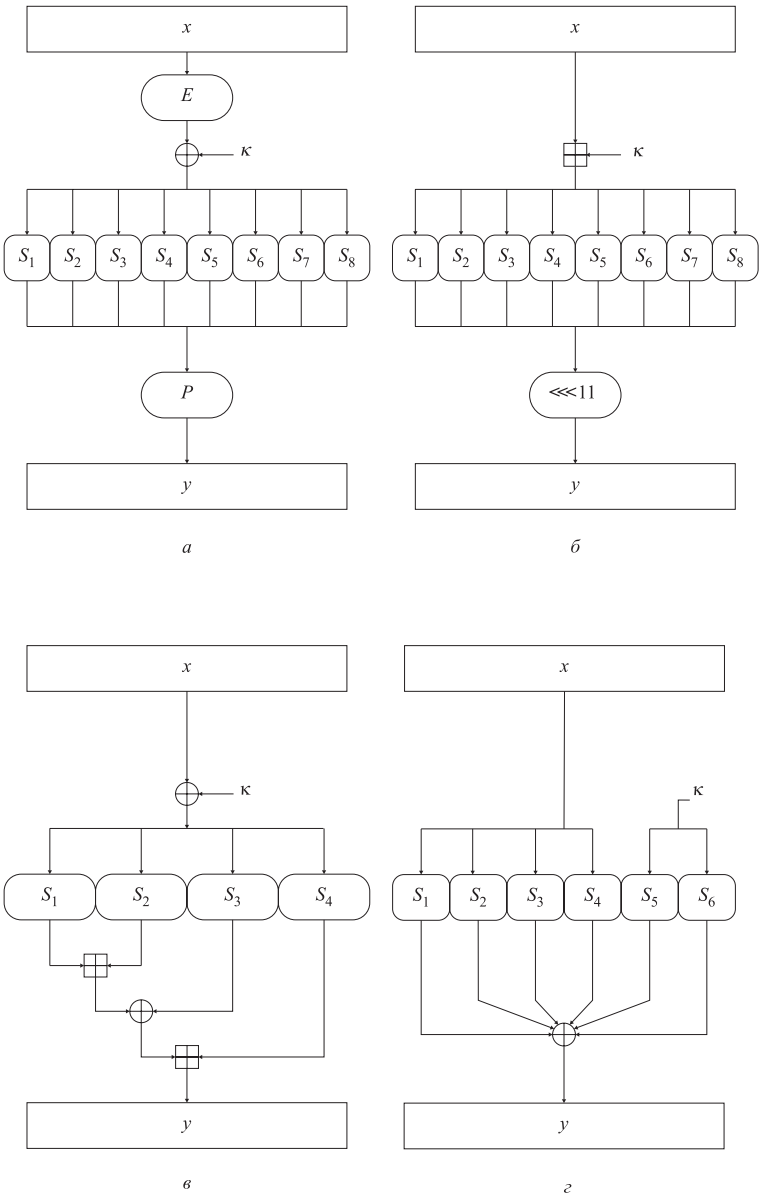


Рис. 3.4. Тактовые функции криптосистем Фейстеля: `DES` (*а*), `GOST` (*б*), `Blowfish` (*в*) `CAST` (*г*)

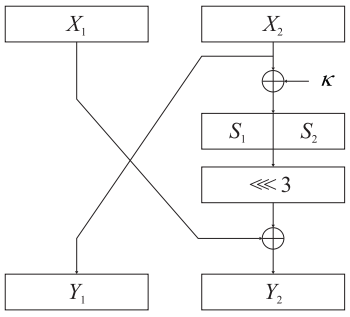


Рис. 3.5. Тактовая подстановка криптосистемы `G`

Расписание ключей `Belt` очень простое. Тактовые ключи $\mathsf{к}_1, \mathsf{к}_2, \ldots, \mathsf{к}_{56} \in \{0,1\}^{32}$ стро-

ятся по ключу θ следующим образом:

$$\kappa_1 \parallel \kappa_2 \parallel \ldots \parallel \kappa_{56} \leftarrow \underbrace{\theta \parallel \theta \parallel \ldots \parallel \theta}_{7 \text{ раз}}.$$

Используется модифицированный экспоненциальный S-блок $H \in S(\{0,1\}^8)$. Модификация состоит в предварительном сдвиге прообраза по правилу $\boxplus$. По S-блоку строятся подстановки $G_r \in S(\{0,1\}^{32})$, $r = 5, 13, 21$:

$$G_r(x_1 \parallel x_2 \parallel x_3 \parallel x_4) = \mathtt{RotHi}^r(H(x_1) \parallel H(x_2) \parallel H(x_3) \parallel H(x_4)).$$

В алгоритме зашифрования и расшифрования открытый текст X и шифртекст Y разбиваются на слова $a, b, c, d \in \{0,1\}^{32}$. Алгоритмы имеют следующий вид.

Алгоритм Зашифрование Belt

Вход: $X \in \{0,1\}^{128}$, $\theta \in \{0,1\}^{256}$.
Выход: $Y \in \{0,1\}^{128}$.
Шаги:
1. $\kappa_1 \parallel \ldots \parallel \kappa_{56} \leftarrow \theta \parallel \ldots \parallel \theta$.
2. $a \parallel b \parallel c \parallel d \leftarrow X$.
3. Для $t = 1, 2, \ldots, 8$ выполнить:
   (1) $b \leftarrow b \oplus G_5(a \boxplus \kappa_{7t-6})$;
   (2) $c \leftarrow c \oplus G_{21}(d \boxplus \kappa_{7t-5})$;
   (3) $a \leftarrow a \boxminus G_{13}(b \boxplus \kappa_{7t-4})$;
   (4) $e \leftarrow G_{21}(b \boxplus c \boxplus \kappa_{7t-3}) \oplus \langle t \rangle_{32}$;
   (5) $b \leftarrow b \boxplus e$;
   (6) $c \leftarrow c \boxminus e$;
   (7) $d \leftarrow d \boxplus G_{13}(c \boxplus \kappa_{7t-2})$;
   (8) $b \leftarrow b \oplus G_{21}(a \boxplus \kappa_{7t-1})$;
   (9) $c \leftarrow c \oplus G_5(d \boxplus \kappa_{7t})$;
   (10) $a \leftrightarrow b$;
   (11) $c \leftrightarrow d$;
   (12) $b \leftrightarrow c$.
4. $Y \leftarrow b \parallel d \parallel a \parallel c$.
5. Возвратить Y.

Алгоритм Расшифрование Belt

Вход: $Y \in \{0,1\}^{128}$, $\theta \in \{0,1\}^{256}$.
Выход: $X \in \{0,1\}^{128}$.
Шаги:
1. $\kappa_1 \parallel \ldots \parallel \kappa_{56} \leftarrow \theta \parallel \ldots \parallel \theta$.
2. $a \parallel b \parallel c \parallel d \leftarrow Y$.
3. Для $t = 8, 7, \ldots, 1$ выполнить:
   (1) $b \leftarrow b \oplus G_5(a \boxplus \kappa_{7t})$;
   (2) $c \leftarrow c \oplus G_{21}(d \boxplus \kappa_{7t-1})$;
   (3) $a \leftarrow a \boxminus G_{13}(b \boxplus \kappa_{7t-2})$;
   (4) $e \leftarrow G_{21}(b \boxplus c \boxplus \kappa_{7t-3}) \oplus \langle t \rangle_{32}$;
   (5) $b \leftarrow b \boxplus e$;
   (6) $c \leftarrow c \boxminus e$;
   (7) $d \leftarrow d \boxplus G_{13}(c \boxplus \kappa_{7t-4})$;
   (8) $b \leftarrow b \oplus G_{21}(a \boxplus \kappa_{7t-5})$;
   (9) $c \leftarrow c \oplus G_5(d \boxplus \kappa_{7t-6})$;
   (10) $a \leftrightarrow b$;
   (11) $c \leftrightarrow d$;
   (12) $a \leftrightarrow d$.
4. $X \leftarrow c \parallel a \parallel d \parallel b$.
5. Возвратить X.

Обратим внимание, что такты зашифрования и расшифрования структурно близки. Переключение между $\mathtt{Belt}_\theta$ и $\mathtt{Belt}_\theta^{-1}$ реализуется изменением порядка следования тактовых ключей, а также перестановками переменных a, b, c, d на шагах 3.12 и 4.

3.11. Атака «грубой силой»

Мы достаточно подробно обсудили принципы построения блочных криптосистем, перейдем теперь к атакам на них. Начнем с *атаки «грубой силой»* — самой простой и в некоторых случаях единственно подходящей. Термин «грубой силой» — это калька с английского brute force. (Некоторые начинающие последователи Виктора так и говорят: *брутфорс*.)

Атака «грубой силой» проводится при известном открытом тексте и направлена на решение задачи **C1**. Атака является *универсальной* в том смысле, что может быть применена к любой криптосистеме.

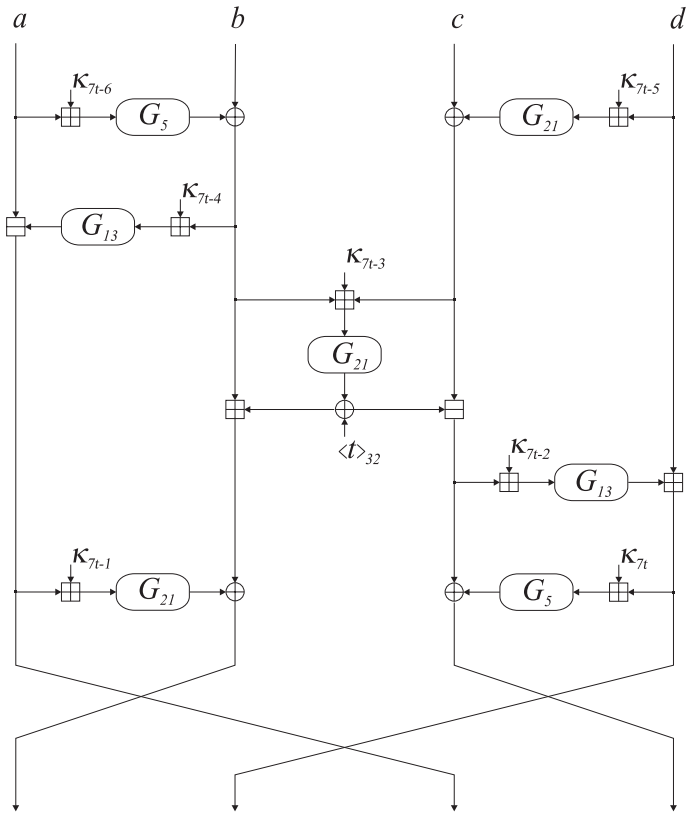


Рис. 3.6. Вычисления на t-м такте зашифрования `Belt`

Пусть Виктор располагает T блоками открытого текста $X_t \in \{0,1\}^{n_b}$ и соответствующими блоками шифртекста $Y_t \in \{0,1\}^{n_b}$. По этому шифрматериалу Виктор строит систему уравнений

$$Y_t = F_\theta(X_t), \quad t = 1, \ldots, T,$$

относительно θ. Как правило, это система имеет единственное решение уже при небольшом T.

Виктор проводит атаку «*грубой силой*», просто выбирая последовательно кандидатов $\hat{\theta} \in \Theta$ и проверяя равенства

$$Y_t \stackrel{?}{=} F_{\hat{\theta}}(X_t), \quad t = 1, \ldots, T.$$

При выполнении всех равенств принимается решение о том, что $\theta = \hat{\theta}$ и поиск прекращается.

Сложность атаки «грубой силой» характеризуется величиной τ — числом использованных кандидатов $\hat{\theta}$. Пусть искомый ключ θ был выбран Трентом из Θ случайно равновероятно, и пусть $\theta_1, \ldots, \theta_{|\Theta|}$ — последовательные ключи-кандидаты, проверяемые Виктором. Тогда среднее значение

$$\mathsf{E}\tau = \sum_{t=1}^{|\Theta|-1} t\mathsf{P}\{\theta = \theta_t\} = \frac{(|\Theta|-1)|\Theta|}{2|\Theta|} = \frac{1}{2}(|\Theta|-1).$$

При оценке мы опустили проверку последнего кандидата $\theta_{|\Theta|}$ — его проверять не надо, он заведомо будет правильным.

Реализация атаки. Для проведения атаки «грубой силой» используются специализированные микропроцессорные устройства или распределенные сетевые вычисления.

Например, в 1999 году с использованием специализированного компьютера Deep Crack стоимостью 250 тыс. долл. была проведена атака «грубой силой» на DES ($|\Theta| = 2^{56}$). Deep Crack проверял около 80 млрд ключей-кандидатов в секунду. Поиск ключа занял 22 ч и 15 мин. Для сравнения, в 2010 г. устройство RIVYERA, разработанное компанией SciEngines, проверяло в секунду уже 292 млрд ключей и его стоимость составляла всего 15 тыс. долл. В сентябре 2002 г. методом «грубой силы» был найден ключ криптосистемы RC5 ($|\Theta| = 2^{64}$). Поиск ключа занял около 4 лет и проводился на 331 252 компьютерах сети Интернет.

В рамках проекта ECRYPT (http://www.ecrypt.eu.org), выполняемого под эгидой Евросоюза, разработаны и периодически обновляются оценки стойкости к атаке «грубой силой» в зависимости от длины ключа. В таблице 3.3 представлены оценки 2012 г.

Таблица 3.3. Стойкость к атаке «грубой силой» (оценки ECRYPT)

Длина ключа (в битах)	Уровень защиты
32	Защита от атак «реального времени» отдельных лиц
64	Краткосрочная защита от атак малой организации (бюджет — 10 тыс. долл.)
72	Краткосрочная защита от атак средней организации (бюджет — 300 тыс. долл.)
80	Краткосрочная защита от атак государственного агентства (бюджет — 300 млн. долл.)
112	Среднесрочная защита (на 20 лет) от атак государственного агентства
128	Долгосрочная защита (на 30 лет) от атак государственного агентства
256	Защита на все обозримое будущее

Простые соотношения. *Простым соотношением* для F называются равенства

$$g_2 F_{h(\theta)} g_1(X) = F_\theta(X), \quad g_1, g_2 \in S(\{0,1\}^{n_b}), \quad h \in S(\Theta), \quad h \neq id,$$

которые выполняются для всех $X \in \{0,1\}^{n_b}$ и $\theta \in \Theta$.

Простое соотношение можно использовать для снижения сложности атаки «грубой силой». Покажем как это сделать. Пусть известны пары (X, Y_1), $(g_1(X), Y_2)$ — «открытый текст — шифртекст» преобразования F_θ. Виктор выбирает ключ-кандидат $\hat{\theta}$ и выполняет зашифрование $\hat{Y} = F_{\hat{\theta}}(X)$. Если $\hat{\theta} = \theta$, то $\hat{Y} = Y_1$, а если $h(\hat{\theta}) = \theta$, то

$$Y_2 = F_\theta g_1(X) = F_{h(\hat{\theta})} g_1(X) = g_2^{-1} g_2 F_{h(\hat{\theta})} g_1(X) = g_2^{-1} F_{\hat{\theta}}(X) = g_2^{-1}(\hat{Y})$$

и $\hat{Y} = g_2(Y_2)$. Таким образом, выполнив одно зашифрование, Виктор может проверить сразу два ключа: $\hat{\theta}$ и $h(\hat{\theta})$.

Пример 3.11 (простые соотношения для G). Имеется 255 простых соотношений для криптосистемы G, описанной в примере 3.10. Действительно, для любого $a \in \{0,1\}^8$ выполняется

$$f_{\kappa \oplus a}(x \oplus a) = f_\kappa(x) \Rightarrow \Sigma_{\kappa \oplus a}(X \oplus (a \parallel a)) \oplus (a \parallel a) = \Sigma_\kappa(X) \Rightarrow$$
$$\Rightarrow F_{\theta \oplus (a \| a \| a \| a)}(X \oplus (a \parallel a)) \oplus (a \parallel a) = F_\theta(X).$$

Последнее тождество является простым соотношением при $a \neq \mathbf{0}$. Используя все эти соотношения, Виктор за одно зашифрование может проверять не один, а сразу 256 ключей криптосистемы G. □

Баланс «время — память». Пусть известна пара «открытый текст X, шифртекст $Y = F_\theta(X)$», и по ней требуется найти ключ θ. Будем считать, что $N = |\Theta| = 2^{n_b}$ и уравнение $Y = F_{\hat\theta}(X)$ имеет малое число решений относительно $\hat\theta$ (одно из решений совпадает с θ).

Описанная выше атака «грубой силой» проводится за время $O(N)$ на памяти $O(1)$. Возможна модификация атаки. На подготовительном этапе Виктор для всевозможных кандидатов $\hat\theta$ вычисляет $\hat Y = F_{\hat\theta}(X)$ и помещает в ячейку памяти по адресу $\hat Y$ значение $\hat\theta$ (вообще говоря, ячейки могут содержать несколько значений). На оперативном этапе Виктор получает Y, обращается к ячейке по адресу Y и определяет все $\hat\theta$, которые переводят X в Y. Виктор делает это за время $O(1)$ на памяти $O(N)$. Сложность подготовительного этапа при этом не учитывается. Виктору важно провести с минимальными издержками именно оперативный этап атаки, как можно быстрее обработать полученную пару (X, Y). К сожалению Виктора, память объема $O(N)$ может быть ему недоступна.

В 1980 г. М. Хеллман предложил метод организации атаки «грубой силой», промежуточный между двумя описанными выше. Этот метод, названный *балансом «время — память»*, позволяет найти θ за время $O(N^{2/3})$ на памяти $O(N^{2/3})$.

Суть метода Хеллмана состоит в следующем. Пусть r — некоторая просто вычисляемая и близкая к биективной функция $\{0,1\}^{n_b} \to \Theta$ (например, перестановка битов) и пусть

$$h_r\colon \Theta \to \Theta, \quad h_r(\theta) = r(F_\theta(X)).$$

Тогда задача определения ключа θ по заданному Y сводится к решению уравнения $h_r(\theta) = r(Y)$, т. е. к обращению функции h_r.

На подготовительном этапе атаки Виктор выбирает ключи-кандидаты $\theta_0 \in \Theta$ и для каждого из них строит траекторию

$$\theta_0, \quad \theta_1 = h_r(\theta_0), \quad \theta_2 = h_r(\theta_1), \ldots, \quad \theta_T = h_r(\theta_{T-1}).$$

Виктор сохраняет начало θ_0 и конец θ_T траектории, а остальные элементы отбрасывает. При этом вместо $O(T)$ ячеек памяти требуется использовать только $O(1)$. Сжатие данных является обратимым — отброшенные элементы траектории могут быть восстановлены по θ_0 за время $O(T)$.

На оперативном этапе Виктор получает Y, вычисляет траекторию

$$Y_0 = r(Y), \quad Y_t = h_r(Y_{t-1}), \quad t = 1, \ldots, T,$$

и на каждом шаге вычислений проверяет совпадение $Y_t \stackrel{?}{=} \theta_T$. Если $\theta = \theta_{T-\tau-1}$ (траектория ключей-кандидатов покрыла θ), то

$$\begin{aligned}
&Y_0 = r(Y) = r(F_{\theta_{T-\tau-1}}(X)) = h_r(\theta_{T-\tau-1}) = \theta_{T-\tau};\\
&Y_1 = h_r(Y_0) = h_r(\theta_{T-\tau}) = \theta_{T-\tau+1};\\
&\quad \ldots\\
&Y_\tau = \theta_T,
\end{aligned}$$

т. е. искомое совпадение будет найдено. Располагая номером итерации τ, на которой совпадение произошло, и началом траектории θ_0, Виктор определяет искомый ключ $\theta_{T-\tau-1}$.

Хеллман предложил использовать R таблиц, в каждой из которых выбирать S различных значений θ_0 и применять разные функции r при построении h_r. Тогда время подготовительного этапа (игнорируется) – $O(RST)$, память – $O(RS)$, время оперативного этапа – $O(RT)$. Оказывается, что вероятность успеха атаки (т. е. покрытия θ хотя бы одной траекторией ключей-кандидатов) максимальна, если $R = S = T = O(N^{1/3})$.

3.12. Разностная атака

Пусть F — d-тактовая блочно-итерационная криптосистема. *Разностная атака* на F проводится при выбираемом открытом тексте. Целью атаки является определение последнего тактового ключа $\varkappa_d$. Определив $\varkappa_d$, можно перейти к $(d-1)$-тактовой криптосистеме, определить ее последний тактовый ключ $\varkappa_{d-1}$ и так далее, вплоть до определения всех тактовых ключей и решения таким образом задачи **C2**. Сразу оговоримся, что существует множество модификаций разностной атаки (о некоторых мы поговорим в конце параграфа). В частности, $\varkappa_d$ может определяться не полностью, а только частично.

Разностная атака была предложена в 1991 году Э. Бихамом и А. Шамиром применительно к криптосистеме `DES`. Как выяснилось впоследствии (эти сведения были первоначально секретными), разработчики `DES` уже знали о технике разностного криптоанализа и предусмотрели в `DES` соответствующие защитные механизмы.

Разностные атаки сегодня являются одними из наиболее эффективных. Так, сразу после появления работы Бихама и Шамира с помощью предложенной ими техники были успешно атакованы криптосистемы `Lucifer`, `FEAL`, `REDOC-II`, `Khafre` и др. В настоящее время при разработке блочных криптосистем стойкость к разностным атакам проверяется, как правило, в первую очередь.

Характеристики и дифференциалы. Для описания разностной атаки нам потребуется несколько построений. Рассмотрим зашифрование пары открытых текстов $X, \tilde{X} \in \{0,1\}^{n_b}$. Пусть $\Delta X = X \oplus \tilde{X}$ — разность между ними. Эта разность порождает последовательность разностей

$$\Delta Y(1) = Y(1) \oplus \tilde{Y}(1), \ldots, \Delta Y(d) = Y(d) \oplus \tilde{Y}(d).$$

Здесь $Y(i), \tilde{Y}(i)$ — результаты выполнения i тактов преобразования $F_\theta = \Sigma_{\varkappa_d} \ldots \Sigma_{\varkappa_1}$ над X и $\tilde{X}$ соответственно. В частности, $Y(d) = F_\theta(X)$ и $\tilde{Y}(d) = F_\theta(\tilde{X})$.

Последовательность $(\alpha, \beta(1), \ldots, \beta(r))$ возможных значений разностей $(\Delta X, \Delta Y(1), \ldots, \Delta Y(r)$ называется *r-тактовой характеристикой*, а пара $(\alpha, \beta(r))$ значений $(\Delta X, \Delta Y(r))$ — *r-тактовым дифференциалом*, $1 \leqslant r \leqslant d$. Характеристика $(\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0})$ и дифференциал $(\mathbf{0}, \mathbf{0})$ являются тривиальными и далее не рассматриваются.

В предположении, что $X, \tilde{X}$ — случайные независимые слова с равномерным на $\{0,1\}^{n_b}$ распределением, а ключ θ выбран из Θ случайно равновероятно, введем *вероятность характеристики*

$$\mathsf{P}\left\{\Delta Y(1) = \beta(1), \ldots, \Delta Y(r) = \beta(r) \mid \Delta X = \alpha\right\}$$

и *вероятность дифференциала*

$$\mathsf{P}\left\{\Delta Y(r) = \beta(r) \mid \Delta X = \alpha\right\}.$$

Подготовительный этап. На подготовительном этапе Виктор находит $(d-1)$-тактовый дифференциал $(\alpha, \beta(d-1))$ с максимально возможной вероятностью p. Сразу скажем, что для успеха атаки требуется выполнение условия $p \gg 2^{-n_b}$.

Поиск высоковероятного дифференциала — непростая задача, успешное решение которой зависит от квалификации криптоаналитика. Виктор должен тщательно изучить криптосистему, проанализировать таблицы разностей S-блоков, исследовать характеристики перемешивания преобразований перестановки и т. д.

Обычно вместо высоковероятного дифференциала $(\alpha, \beta(d-1))$ Виктор ищет высоковероятную характеристику $(\alpha, \beta(1), \ldots, \beta(d-1))$ и использует оценки:

$$p \geqslant \mathsf{P}\left\{\Delta Y(1) = \beta(1), \ldots, \Delta Y(r) = \beta(d-1) \mid \Delta X = \alpha\right\} \approx$$

$$\approx \mathsf{P}\left\{\Delta Y(1) = \beta(1) \mid \Delta X = \alpha\right\} \prod_{i=2}^{d-1} \mathsf{P}\left\{\Delta Y(i) = \beta(i) \mid \Delta Y(i-1) = \beta(i-1)\right\}.$$

При этом Виктор идеализирует расписание ключей F, считая, что при случайном равновероятном выборе θ последовательность $(\kappa_1, \dots, \kappa_{d-1})$ равномерно распределена на K^{d-1}. Как правило, такая идеализация не сказывается на адекватности окончательных выводов.

Оперативный этап. Выбирая ключ-кандидат $\hat{\kappa}_d \in K$, Виктор может построить прогноз

$$Z(Y, \tilde{Y}; \hat{\kappa}_d) = \Sigma^{-1}_{\hat{\kappa}_d}(Y) \oplus \Sigma^{-1}_{\hat{\kappa}_d}(\tilde{Y})$$

предпоследней разности $\Delta Y(d-1)$.

На оперативном этапе разностной атаки Виктор случайным образом выбирает открытые тексты $X_t \in \{0,1\}^{n_b}$ и определяет шифртексты $Y_t = F_\theta(X_t)$, $\tilde{Y}_t = F_\theta(X_t \oplus \alpha)$, $t = 1, \dots, T$. В качестве оценки искомого ключа κ_d Виктор выбирает значение $\hat{\kappa}_d$, доставляющее максимум сумме

$$W(\hat{\kappa}_d) = \sum_{t=1}^{T} \mathsf{I}\left\{ Z(Y_t, \tilde{Y}_t; \hat{\kappa}_d) = \beta(d-1) \right\}.$$

Проведем грубый анализ сложности атаки (более тонкий анализ предполагает учет свойств конкретной криптосистемы). При $\hat{\kappa}_d = \kappa_d$ среднее значение

$$\mathsf{E}W(\hat{\kappa}_d) = \sum_{t=1}^{T} \mathsf{P}\left\{ Z(Y_t, \tilde{Y}_t; \hat{\kappa}_d) = \beta(d-1) \right\} = Tp.$$

Если же $\hat{\kappa}_d \neq \kappa_d$, то можно предположить (это еще одно допущение, которое в определенных случаях оказывается адекватным), что слова $Z(Y_t, \tilde{Y}_t; \hat{\kappa}_d)$ равномерно распределены на $\{0,1\}^{n_b} \setminus \{\mathbf{0}\}$. При этом

$$\mathsf{E}W(\hat{\kappa}_d) = T/(2^{n_b} - 1).$$

Потребуем, чтобы среднее значение (целочисленной) целевой функции W на верном ключе было больше, чем на остальных:

$$\mathsf{E}W(\kappa_d) \geqslant \mathsf{E}W(\hat{\kappa}_d) + 1, \quad \hat{\kappa}_d \neq \kappa_d.$$

Отсюда

$$T \geqslant \frac{1}{p - 1/(2^{n_b} - 1)}.$$

Модификации. Перечислим основные модификации разностной атаки.

Обобщенные разности. Разность определяется не с помощью операции $\oplus$, а с помощью другой алгебраической операции. Например, $\Delta X = X \boxminus \tilde{X}$.

Усеченные разности. Контролируется не все, а только выборочные символы разностей $\Delta Y(i)$.

Запрещенные дифференциалы. Вместо высоковероятных дифференциалов $(\alpha, \beta(r))$ используются запрещенные дифференциалы, т. е. дифференциалы с нулевой вероятностью. Если $(\alpha, \beta(d-1))$ — запрещенный дифференциал, то для определения κ_d нужно не максимизировать $\sum Z(Y_t, \tilde{Y}_t; \hat{\kappa}_d)$, а браковать те $\hat{\kappa}_d$, для которых $Z(Y_t, \tilde{Y}_t; \hat{\kappa}_d) = \beta(d-1)$.

Атака на несколько тактовых ключей. Определяется не один, а несколько последних тактовых ключей. Для определения $(\kappa_{r+1}, \dots, \kappa_d)$ Виктор находит высоковероятный r-тактовый дифференциал $(\alpha, \beta(r))$, просматривает ключи-кандидаты $(\hat{\kappa}_{r+1}, \dots, \hat{\kappa}_d)$ и составляет по ним прогнозы

$$Z(Y, \tilde{Y}; \hat{\kappa}_{r+1}, \dots, \hat{\kappa}_d) = \Sigma^{-1}_{\hat{\kappa}_{r+1}} \dots \Sigma^{-1}_{\hat{\kappa}_d}(Y) \oplus \Sigma^{-1}_{\hat{\kappa}_{r+1}} \dots \Sigma^{-1}_{\hat{\kappa}_d}(\tilde{Y})$$

разности $\Delta Y(r)$. Как и ранее, в качестве искомой оценки последних тактовых ключей выбирается тот набор ключей-кандидатов, на котором прогноз разности чаще всего совпадает с $\beta(r)$.

Частичная информация о ключах. В некоторых случаях различные наборы ключей-кандидатов могут давать одинаковые прогнозы $Z(Y, \tilde{Y}; \hat{\kappa}_{r+1}, \ldots, \hat{\kappa}_d)$. В таких случаях Виктор может определить только частичную информацию $\mathcal{I} = \mathcal{I}(\kappa_{r+1}, \ldots, \kappa_d)$ о последних тактовых ключах. Эту информацию часто называют *эффективными битами ключа*. Эффективные биты выбираются так, чтобы при любой их оценке $\hat{\mathcal{I}} = \mathcal{I}(\hat{\kappa}_{r+1}, \ldots, \hat{\kappa}_d)$ для пары шифртекстов $Y, \tilde{Y}$ можно было построить прогноз разности $\Delta Y(r)$. Частичная информация о ключах определяется по той же схеме, что и полная. После определения $\mathcal{I}$ недостающие данные о последних тактовых ключах Виктор может получить с помощью атаки «грубой силой» или с помощью той же разностной атаки, но уже с другим дифференциалом.

Пример 3.12 (разностная атака на DES). В атаке Бихама и Шамира на DES используются два 13-тактовых дифференциала, каждый с вероятностью $\approx 2^{-47{,}2}$. С помощью этих дифференциалов можно в совокупности определить 52 бита двух последних тактовых ключей. При этом требуется использовать $T = 2^{47}$ пар «открытый текст — шифртекст». Вероятность успеха атаки ≈ 0.58. □

Пример 3.13 (разностная атака на G). Проведем разностную атаку на модельную криптосистему G, описанную в примере 3.10. Для простоты будем опускать при зашифровании заключительную перестановку половинок, которая выполняется в криптосистемах Фейстеля.

Рассмотрим тактовую функцию f_κ. Пусть на вход f_κ подана пара случайных октетов x, $\tilde{x}$ с разностью $\gamma = 10000000$. После сложения с κ разность не изменится, и на вход S_1 попадут тетрады с разностью 1000, а на вход S_2 — одинаковые тетрады. В соответствии с расчетами, проведенными в примере 3.9, на выходе S_1 с вероятностью $1/4$ будет получена разность 0001. Выходы S_2 будут одинаковыми. Вычисление значений f_κ заканчивается объединением выходов S-блоков в октеты и циклическим сдвигом этих октетов на 3 позиции влево. Окончательная разность будет равняться γ. Таким образом,

$$\mathsf{P}\{f_\kappa(x) \oplus f_\kappa(\tilde{x}) = \gamma \mid x \oplus \tilde{x} = \gamma\} = \frac{1}{4},$$

или, другими словами, f_κ сохраняет разность γ с вероятностью $1/4$.

Рассуждения можно продолжить и построить следующую 7-тактовую разностную характеристику для G:

$$\begin{aligned}
\alpha &= \gamma \parallel \mathbf{0} \\
\beta(1) &= \mathbf{0} \parallel \gamma \qquad \text{(с вероятностью 1)} \\
\beta(2) &= \gamma \parallel \gamma \qquad \text{(с вероятностью 1/4)} \\
\beta(3) &= \gamma \parallel \mathbf{0} \qquad \text{(с вероятностью 1/4)} \\
\beta(4) &= \mathbf{0} \parallel \gamma \qquad \text{(с вероятностью 1)} \\
\beta(5) &= \gamma \parallel \gamma \qquad \text{(с вероятностью 1/4)} \\
\beta(6) &= \gamma \parallel \mathbf{0} \qquad \text{(с вероятностью 1/4)} \\
\beta(7) &= \mathbf{0} \parallel \gamma \qquad \text{(с вероятностью 1).}
\end{aligned}$$

Вероятность характеристики и соответствующего дифференциала $(\alpha, \beta(7))$ оценим как произведение однотактовых характеристик: $p \geqslant 2^{-8}$. Поскольку $n_b = 16$ и $p \gg 2^{-16}$, дифференциал можно использовать для определения информации $\mathcal{I} = \mathcal{I}(\kappa_8)$ о последнем тактовом ключе. Выясним, какую информацию $\mathcal{I}$ мы можем восстановить, и обсудим детали восстановления.

Разобъем слова $Y(i)$ на тетрады:

$$Y(i) = Y_1(i) \parallel Y_2(i) \parallel Y_3(i) \parallel Y_4(i), \quad Y_k(i) \in \{0,1\}^4.$$

Аналогичные разбиения введем для $\tilde{Y}(i)$. Пусть $\kappa_8 = a \parallel b$, $a, b \in \{0,1\}^4$.

Пару $(X, \tilde{X})$ открытых текстов назовем верной, если она удовлетворяет дифференциалу, т. е. $\Delta X = \alpha$ и $\Delta Y(7) = \beta(7)$. Для верной пары

$$\tilde{Y}_1(7) = Y_1(7), \quad \tilde{Y}_2(7) = Y_2(7), \quad \tilde{Y}_3(7) = Y_3(7) \oplus 1000, \quad \tilde{Y}_4(7) = Y_4(7),$$

и поэтому

$$\Delta Y(8) = \gamma \parallel {*}0000{***}. \tag{3.3}$$

Здесь знак * соответствует символам, которые могут принимать любые значения, только не все нулевые одновременно.

Пару $(X, \tilde{X})$, для которой выполняется соотношение (3.3), назовем подходящей. Существуют подходящие пары, которые не являются верными. Впрочем вероятность их появления $\approx 2^{-12} \ll p$. Поэтому будем упрощать рассуждения и предполагать далее, что всякая пара, удовлетворяющая (3.3), является верной.

По верной паре построим слово $Z \in \{0,1\}^4$. Это слово составим из символов $\Delta Y(8)$ с номерами 14, 15, 16 и 9, т. е. из символов, помеченных знаком *. Тетрады a и b последнего тактового ключа участвуют в следующих разностных соотношениях:

$$S_1(Y_3(7) \oplus a) \oplus S_1(Y_3(7) \oplus 1000 \oplus a) = Z,$$
$$S_2(Y_4(7) \oplus b) \oplus S_2(Y_4(7) \oplus b) = \mathbf{0}.$$

Второе соотношение является тождеством и не позволяет определить символы b, а вот с помощью первого можно получить информацию об a.

Перепишем первое соотношение:

$$S_1(Y_1(8) \oplus a) \oplus S_1(Y_1(8) \oplus 1000 \oplus a) = Z. \tag{3.4}$$

Нам известны все используемые здесь слова, за исключением a. S-блок S_1 устроен так, что при изменении 1-го символа a равенство не нарушится. Поэтому информация $\mathcal{I}$ о последнем тактовом ключе, которую можно получить, касается 2-го, 3-го и 4-го символов a. Для определения $\mathcal{I}$ следует ожидать появления верных пар, проверять для них выполнение (3.4) при всевозможных значениях $\hat{\mathcal{I}}$ и принимать решение в пользу того значения, на котором (3.4) выполнилось наибольшее количество раз.

Пусть y — случайное слово с равномерным распределением на $\{0,1\}^4$. Прямые расчеты показывают, что слово $z = S_1(y) \oplus S_1(y \oplus 1000)$ имеет следующее распределение вероятностей:

$$\mathsf{P}\{z = 0001\} = \frac{1}{4}, \quad \mathsf{P}\{z = 1001\} = \frac{1}{4}, \quad \mathsf{P}\{z = 0101\} = \frac{1}{2}$$

(по первому равенству и рассчитывалась вероятность разностной характеристики). Поэтому случайное значение $\hat{\mathcal{I}}$ приведет к выполнению (3.4) с вероятностью

$$\left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{3}{8}.$$

С другой стороны, при $\hat{\mathcal{I}} = \mathcal{I}$ равенство будет выполняться с вероятностью 1. Различие вероятностей и является основанием результативности разностной атаки.

В ходе вычислительных экспериментов 3 бита последнего тактового ключа гарантированно восстанавливались при анализе ≈ 8 верных пар. При этом требовалось сгенерировать $T \approx 2000$ пар открытого текста. □

3.13. Линейная атака

Линейная атака проводится при известном открытом тексте. В самом простом случае, который мы и рассмотрим, объектом атаки является d-тактовая блочно-итерационная криптосистема F с множеством тактовых ключей $K = \{0,1\}^{n_b}$ и тактовыми подстановками следующего вида:

$$\Sigma_{\kappa}\colon X \mapsto s(X \oplus \kappa), \quad s \in S(\mathbb{F}_2^n).$$

Целью атаки является определение одного бита информации о $(\kappa_1, \ldots, \kappa_d)$. Другие биты могут быть определены с помощью той же линейной атаки, но с другими настройками.

Линейная атака была предложена в 1993 г. М. Мацуи применительно к DES. Мацуи довел атаку до практической стадии — в 1994 году в ходе трудоемкого вычислительного эксперимента была впервые решена задача **C1** для DES.

Линейные аппроксимации. Прежде чем описывать линейную атаку, проведем подготовительные построения. Будем предполагать, что ключ θ выбран из Θ случайно равновероятно. Как и в разностной атаке, будем идеализировать расписание ключей F, считая, что последовательность $(\kappa_1, \ldots, \kappa_d)$ равномерно распределена на K^d.

Зашифрование $Y = F_\theta(X)$ состоит в выполнении тактовых преобразований

$$Y(1) = s(X \oplus \kappa_1),\ Y(2) = s(Y(1) \oplus \kappa_2),\ \ldots,\ Y = s(Y(d-1) \oplus \kappa_d).$$

Для подстановки s, которая применяется на i-м такте, используем линейную аппроксимацию $(\beta(i-1), \beta(i))$, пусть q_i — ее вероятность. Обозначим

$$b(\kappa_1, \ldots, \kappa_d) = \beta(0) \cdot \kappa_1 \oplus \ldots \oplus \beta(d-1) \cdot \kappa_d.$$

Объединение аппроксимаций $(\beta(i-1), \beta(i))$, $i = 1, \ldots, d$, называется *d-тактовой аппроксимацией* и обозначается через $(\beta(0), \beta(1), \ldots, \beta(d))$. Вероятность d-тактовой аппроксимации определяется как

$$p = \mathsf{P}\left\{\beta(0) \cdot X \oplus \beta(d) \cdot F_\theta(X) = b(\kappa_1, \ldots, \kappa_d)\right\}.$$

Тривиальные аппроксимации $(\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0})$ с вероятностью 1 договоримся далее не рассматривать.

Введем случайные величины

$$\begin{aligned}
\xi_1 &= \beta(0) \cdot (X \oplus \kappa_1) \oplus \beta(1) \cdot Y(1);\\
\xi_2 &= \beta(1) \cdot (Y(1) \oplus \kappa_2) \oplus \beta(2) \cdot Y(2);\\
&\ldots\\
\xi_d &= \beta(d-1) \cdot (Y(d-1) \oplus \kappa_d) \oplus \beta(d) \cdot Y.
\end{aligned}$$

Эти величины независимы, для них $\mathsf{P}\{\xi_i = 0\} = q_i$ и $\mathsf{P}\{\xi_1 \oplus \ldots \oplus \xi_d = 0\} = p$. Определить p по q_i можно с помощью следующей леммы.

Лемма 3.2 (о набегании знаков). *Пусть $\xi_1, \ldots, \xi_d$ — независимые бернуллиевские случайные величины и $\mathsf{P}\{\xi_i = 0\} = q_i$. Тогда*

$$\mathsf{P}\{\xi_1 \oplus \xi_2 \ldots \oplus \xi_d = 0\} = \frac{1}{2} + 2^{d-1} \prod_{i=1}^{d} \left(q_i - \frac{1}{2}\right).$$

Доказательство. Введем преобладания $\epsilon_i = \mathsf{P}\{\xi_1 \oplus \ldots \oplus \xi_i = 0\} - \frac{1}{2}$, $i = 1, \ldots, d$. Формула леммы справедлива при $d = 1$. Для $d \geqslant 2$ имеем

$$\begin{aligned}\epsilon_d &= \sum_{c\in\{0,1\}} \mathsf{P}\{\xi_1 \oplus \ldots \oplus \xi_{d-1} = c,\ \xi_d = c\} - \frac{1}{2} = \\ &= \left(\epsilon_{d-1} + \frac{1}{2}\right) q_d + \left(\frac{1}{2} - \epsilon_{d-1}\right)(1 - q_d) - \frac{1}{2} = \\ &= 2\epsilon_{d-1}\left(q_d - \frac{1}{2}\right),\end{aligned}$$

откуда и следует требуемый результат. □

Подготовительный этап. На подготовительном этапе линейной атаки Виктор находит d-тактовую линейную аппроксимацию $(\beta(0), \ldots, \beta(d))$ с максимально отличной от $\frac{1}{2}$ вероятностью p. Если $(p - 1/2)^2 \gg 2^{-n_b}$, то с помощью этой аппроксимации Виктор сможет определить бит $b(\kappa_1, \ldots, \kappa_d)$.

Как и в разностной атаке, подготовительный этап является творческим. Виктор должен изучить структуру криптосистемы, проанализировать нелинейность S-блоков, исследовать характеристики перемешивания и т. д.

Оперативный этап. На оперативном этапе Виктор получает открытые тексты X_t и перехватывает соответствующие шифртексты $Y_t = F_\theta(X_t)$, $t = 1, 2, \ldots, T$. Сделаем еще одно допущение: пусть открытые тексты X_t являются реализациями независимых случайных величин с равномерным распределением на $\{0,1\}^{n_b}$. Это допущение упрощает рассуждения, но не сказывается на адекватности окончательных выводов в практически важных случаях.

По полученному шифрматериалу Виктор определяет сумму

$$W = \sum_{t=1}^{T} \mathsf{I}\{\beta(0) \cdot X_t = \beta(d) \cdot Y_t\}.$$

Если $p > 1/2$, то оценка бита $b = b(\kappa_1, \ldots, \kappa_d)$ строится следующим образом:

$$\hat{b} = \begin{cases} 0, & W \geqslant \frac{T}{2}, \\ 1 & \text{в противном случае.}\end{cases}$$

Если $p < 1/2$, то эта оценка инвертируется.

Проведем анализ требуемого для атаки числа пар T. Пусть, не нарушая общности, $p > 1/2$ и $b = 0$. Величина W представляет собой сумму независимых бернуллиевских случайных величин $\delta_t = \mathsf{I}\{\beta(0) \cdot X_t = \beta(d) \cdot Y_t\}$ со средним $\mathsf{E}\delta_t = p$. Поэтому вероятность ошибки при оценивании b можно оценить по теореме Муавра — Лапласа:

$$\begin{aligned}\mathsf{P}\left\{\hat{b} \neq b\right\} &= \mathsf{P}\left\{W < \frac{T}{2}\right\} = \\ &= \mathsf{P}\left\{\frac{W - pT}{\sqrt{Tp(1-p)}} < \frac{(1/2 - p)T}{\sqrt{Tp(1-p)}}\right\} \approx \\ &\approx \Phi\left(-2\sqrt{T}(p - 1/2)\right).\end{aligned}$$

Здесь $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-x^2/2} dx$ — функция распределения стандартного нормального закона. Для того чтобы вероятность ошибки была мала, должно использоваться порядка $|p - 1/2|^{-2}$ пар «открытый текст — шифртекст».

Модификации. Методы линейного криптоанализа можно применить не только к криптосистемам с тактовыми подстановками $X \mapsto s(X \oplus \kappa)$, но и к другим криптосистемам, например Фейстеля. При этом используются различные модификации базовой линейной атаки, описанной выше. Перечислим основные из них.

Несколько аппроксимаций. Используется не одна, а несколько линейных аппроксимаций. Разные аппроксимации позволяют определить разные биты ключа, или разные аппроксимации повышают качество определения одного и того же бита.

Атака на последние тактовые ключи. Определяется информация $\mathcal{I} = \mathcal{I}(\kappa_{r+1}, \ldots, \kappa_d)$ о последних тактовых ключах. Для этого используется r-тактовая линейная аппроксимация $(\beta(0), \ldots, \beta(r))$. Предполагается, что по всякой оценке $\hat{\mathcal{I}}$ можно построить прогноз бита $\beta(r) \cdot Y(r)$. В ходе атаки выбирается та оценка, для которой эти прогнозы максимально часто совпадают с битами $\beta(0) \cdot X$ (случай А) или еще чаще отличаются от них (случай Б). Одновременно строится оценка для бита $b(\kappa_1, \ldots, \kappa_r)$. Это либо 0 в случае А, либо 1 в случае Б.

Аппроксимации с нулевым преобладанием. Вместо аппроксимаций, вероятности которых существенно отличаются от $1/2$, используются аппроксимации с вероятностью $1/2$. Среди оценок $\hat{\mathcal{I}}$ выбираются те, для которых корреляции между $\beta(0) \cdot X$ и прогнозом $\beta(r) \cdot Y(r)$ не проявляются.

Пример 3.14 (линейная атака на DES). В своей второй линейной атаке на DES, проведенной в 1994 г., М. Мацуи использовал две 14-тактовые линейные аппроксимации с вероятностью $\frac{1}{2} + 1,19 \cdot 2^{-22}$ каждая. На оперативном этапе было обработано $T = 2^{43}$ пар «открытый текст – шифртекст». По полученному шифрматериалу были определены 26 битов ключа θ. Оставшиеся 30 битов ключа были найдены «грубой силой». □

Пример 3.15 (линейная атака на G). Проведем линейную атаку на криптосистему G. Как и в предыдущем примере для G, будем игнорировать заключительную перестановку половинок при зашифровании.

Тактовую функцию f_κ можно представить в виде $f_\kappa(x) = s(x \oplus \kappa)$, где

$$s(x) = (S_1(x_1) \parallel S_2(x_2)) \lll 3, \quad x = x_1 \parallel x_2, \quad x_i \in \{0,1\}^4.$$

Пусть $\gamma = 11100001$. С учетом вычислений, проведенных в примере 3.7,

$$\mathsf{P}\{\gamma \cdot x = \gamma \cdot s(x)\} = \mathsf{P}\{1110 \cdot x_1 \oplus 0001 \cdot x_2 = 0011 \cdot S_1(x_1) \oplus 1100 \cdot S_2(x_2)\} = \frac{11}{16}.$$

Перейдем к тактовым подстановкам. Рассмотрим следующие соотношения, связывающие прообраз $X = X_1 \parallel X_2$, образ $Y_1 \parallel Y_2$ и тактовый ключ κ:

$$\begin{aligned} &\gamma \cdot (X_2 \oplus Y_1) = 0; \\ &\gamma \cdot (X_1 \oplus Y_1 \oplus Y_2) = \gamma \cdot \kappa; \\ &\gamma \cdot (X_1 \oplus X_2 \oplus Y_2) = \gamma \cdot \kappa. \end{aligned}$$

Поскольку $Y_1 = X_2$ и $Y_2 = X_1 \oplus s(X_2 \oplus \kappa)$, для случайного X с равномерным распределением на $\{0,1\}^{16}$ первое равенство выполняется с вероятностью 1, а второе и третье — с вероятностью $\frac{11}{16}$.

Используя данные факты, можно построить следующую 7-тактовую линейную ап-

проксимацию для $\mathtt{G}$:

$$\begin{array}{ll}
\beta(0) = \mathbf{0} \parallel \gamma & \\
\beta(1) = \gamma \parallel \mathbf{0} & \text{(с вероятностью 1)} \\
\beta(2) = \gamma \parallel \gamma & \text{(с вероятностью 11/16)} \\
\beta(3) = \mathbf{0} \parallel \gamma & \text{(с вероятностью 11/16)} \\
\beta(4) = \gamma \parallel \mathbf{0} & \text{(с вероятностью 1)} \\
\beta(5) = \gamma \parallel \gamma & \text{(с вероятностью 11/16)} \\
\beta(6) = \mathbf{0} \parallel \gamma & \text{(с вероятностью 11/16)} \\
\beta(7) = \gamma \parallel \mathbf{0} & \text{(с вероятностью 1).}
\end{array}$$

Вероятность аппроксимации:

$$q = \frac{1}{2} + 2^3 \left(\frac{3}{16}\right)^4 \approx \frac{1}{2} + 2^{-6,6}.$$

Аппроксимацию можно использовать для определения последнего тактового ключа κ_8 (частично) и бита $\gamma \cdot (\kappa_2 \oplus \kappa_3 \oplus \kappa_5 \oplus \kappa_6)$. В вычислительных экспериментах удавалось определять 5 битов информации о κ_8. □

3.14. Режимы шифрования

В протоколе, описанном в начале главы, Алиса и Боб выполняют зашифрование и расшифрование следующим образом:

$$Y_t = F_\theta(X_t), \quad X_t = F_\theta^{-1}(Y_t), \quad t = 1, 2, \ldots, T.$$

При этом говорят о шифровании в режиме *простой замены*. Этот режим принято обозначать аббревиатурой ECB (Electronic CodeBook).

В режиме ECB каждый блок открытого текста обрабатывается отдельно от остальных. Поэтому возникают следующие угрозы:

Перестановка блоков. Виктор располагает форматами банковских документов и переставляет местами блоки с младшими и старшими цифрами суммы платежа.

Анализ повторов блоков. Виктор располагает форматами банковских документов и располагает информацией о повторе их блоков. Анализ повторов блоков перехваченного шифртекста позволяет Виктору установить тип документа.

Для защиты от этих угроз используются другие режимы шифрования. В режимах *сцепления блоков* (CBC, Cipher Block Chaining) и *гаммирования с обратной связью* (CFB, Cipher FeedBack) результат зашифрования блока X_t зависит от всех предыдущих блоков открытого текста $X_1, \ldots, X_{t-1}$. Правила шифрования в этих режимах соответственно имеют вид:

$$\begin{array}{ll}
\text{CBC:} & Y_t = F_\theta(X_t \oplus Y_{t-1}), \quad X_t = Y_{t-1} \oplus F_\theta^{-1}(Y_t); \\
\text{CFB:} & Y_t = X_t \oplus F_\theta(Y_{t-1}), \quad X_t = Y_t \oplus F_\theta(Y_{t-1}).
\end{array}$$

Здесь Y_0 – некоторое наперед заданное слово, называемое *синхропосылкой*.

Синхропосылка обеспечивает уникальность результатов криптографического преобразования на одном и том же ключе. Синхропосылка является несекретным объектом и может передаваться вместе с зашифрованными данными.

В режимах *обратной связи по выходу* (OFB, Output FeedBack) и счетчика (CTR, Counter) зависимость от блоков $X_1, \ldots, X_{t-1}$ отсутствует. В данных режимах вырабатывается гамма — последовательность векторов $\Gamma_1, \ldots, \Gamma_T \in \{0,1\}^{n_b}$, которая используется как для зашифрования, так и для расшифрования:

$$Y_t = X_t \oplus \Gamma_t, \quad X_t = Y_t \oplus \Gamma_t, \quad t = 1, \ldots, T.$$

Гамма вырабатывается по правилам:

$$\begin{aligned} &\text{OFB}: \quad \Gamma_t = F_\theta(\Gamma_{t-1});\\ &\text{CTR}: \quad \Gamma_t = F_\theta(S_t), \quad S_t = \varphi(S_{t-1}). \end{aligned}$$

Здесь Γ_0, S_0 — синхропосылки, $\varphi\colon \{0,1\}^{n_b} \to \{0,1\}^{n_b}$ — *функция инкремента*. Функция инкремента выбирается так, чтобы обеспечить большой период последовательности S_t. Часто инкремент состоит в добавлении числа 1 (представленного двоичным словом) по правилу $\boxplus$.

Сравнительные характеристики режимов приведены в таблице 3.4.

Таблица 3.4. Режимы шифрования

Свойства	Режимы				
	ECB	CBC	CFB	OFB	CTR
Зависимость от $X_1, \ldots, X_{t-1}$	–	+	+	–	–
Использование F_θ^{-1}	+	+	–	–	–
Восстановление после ошибки в шифртексте[1]	+	+	+	+	+
Восстановление после ошибки в синхропосылке[2]	не исп.	+	+	–	–
распараллеливание[3]	+	–	–	–	+

[1] даже если один из блоков Y_t изменен при передаче, при расшифровании начиная с некоторого $\tau > t$ будут получены корректные блоки X_τ;
[2] даже если синхропосылка изменена при передаче, при расшифровании начиная с некоторого τ будут получены корректные блоки X_τ;
[3] шифрование различных блоков может выполняться одновременно на нескольких процессорах.

3.15. Имитозащита

Блочные криптосистемы могут использоваться не только для обеспечения конфиденциальности, но и для контроля целостности сообщений. Для этого по блочной криптосистеме строится система имитозащиты.

Определение 3.3. *Системой имитозащиты называется семейство*

$$I = \{I_\theta\colon \theta \in \Theta\}$$

ключезависимых функций $I_\theta\colon \{0,1\}^* \to \{0,1\}^{n_m}$. *Значение* $I_\theta(X)$ *называется имитовставкой* X *(на ключе* θ*).*

В англоязычной литературе системы имитозащиты называются также *MAC-системами* (от Message Authentication Codes). Отсюда индекс m в размерности имитовставки.

Для контроля целостности сообщения X Алиса вместе с шифртекстом Y отправляет Бобу имитовставку $Z = I_\theta(X)$. Боб получает шифртекст Y' (который может отличаться от Y), находит открытый текст X', вычисляет имитовставку $Z' = I_\theta(X')$ и сравнивает ее с Z. Если имитовставки различаются, то Боб принимает решение о том, что $Y' \neq Y$, т. е. шифртекст был изменен в канале связи. Если имитовставки совпадают, то Боб принимает X'.

Детали могут отличаться. Например, имитовставка может вычисляться не от открытого текста X, а от шифртекста Y. При этом Боб сначала проверяет имитовставку, а только затем, при успешной проверке выполняет расшифрование Y.

При атаках на систему имитозащиты I Виктор получает полную или частичную информацию о сообщениях X (или даже выбирает эти сообщения) и перехватывает соответствующие имитовставки $Z = I_\theta(X)$. Виктору требуется решить одну из следующих задач:

M1: определить ключ θ;

M2: найти имитовставку заданного сообщения, отличного от предыдущих;

M3: найти имитовставку произвольного сообщения, отличного от предыдущих.

Ясно, что решение первой задачи приводит к решению второй, а решение второй — к решению третьей. Таким образом, третья задача является самой простой и на ней концентрируется основное внимание при оценке стойкости систем имитозащиты.

Если ключ имитозащиты θ используется также для шифрования $X \mapsto Y$, то у Виктора появляется дополнительный шифрматериал Y и, как следствие, дополнительный потенциал при решении **M1**, **M2**, **M3**. Известный криптографический принцип — *ключ должен использоваться по одному назначению* — запрещает совмещение ключей шифрования и имитозащиты. Тем не менее такое совмещение может быть разрешено, если оно тщательно проанализировано и слабости не обнаружены. Например, в СТБ 34.101.31 определены алгоритмы, которые на одном и том же ключе θ выполняют одновременно и шифрование, и имитозащиту данных.

Рассмотрим два распространенных подхода к построению систем имитозащиты.

Схема CBC-MAC. Для имитозащиты сообщения

$$X = X_1 \,\|\, X_2 \,\|\, \ldots \,\|\, X_T, \quad X_t \in \{0,1\}^{n_b},$$

выполняются вычисления, аналогичные шифрованию в режиме CBC с нулевой синхропосылкой:

$$Y_t = F_\theta(X_t \oplus Y_{t-1}), \quad t = 1, 2, \ldots, T, \quad Y_0 = 0^{n_b}.$$

Имитовставкой объявляется слово Y_T, если $n_m = n_b$, или выборочные символы этого слова, если $n_m < n_b$. Построенные таким образом системы имитозащиты принято обозначать аббревиатурой CBC-MAC.

Алгоритм выработки имитовставки, определенный в ГОСТ 28147, соответствует схеме CBC-MAC. В этом алгоритме $T \geqslant 2$, $n_b = 64$, $n_m \leqslant 32$, имитовставкой объявляются последние символы Y_T.

Существует несколько модификаций схемы CBC-MAC. Одна из них, известная как OMAC, была предложена Т. Иватой и К. Куросавой в 2003 году. Эта схема использована в следующем алгоритме имитозащиты, определенном в СТБ 34.101.31.

Алгоритм Выработка имитовставки (СТБ 34.101.31)

Вход: $X \in \{0,1\}^*$ — сообщение, $\theta \in \{0,1\}^{256}$ — ключ.

Выход: $Z \in \{0,1\}^{64}$ — имитовставка.

Шаги:

1. Непустое сообщение X представить в виде $X_1 \,\|\, \ldots \,\|\, X_{T-1} \,\|\, X_T$, где $|X_1| = \ldots = |X_{T-1}| = 128$, $0 < |X_T| \leqslant 128$. Для пустого X считать, что $T = 1$ и $|X_1| = 0$.
2. Установить $s \leftarrow 0^{128}$, $r \leftarrow \mathtt{Belt}_\theta(s)$.
3. Для $t = 1, 2, \ldots, T-1$ выполнить: $s \leftarrow \mathtt{Belt}_\theta(s \oplus X_i)$.
4. Если $|X_T| = 128$, то $s \leftarrow s \oplus X_T \oplus \varphi_1(r)$, иначе $s \leftarrow s \oplus \mathtt{Pad}_{128}(X_T) \oplus \varphi_2(r)$.
5. Записать в Z первые 64 символа слова $\mathtt{Belt}_\theta(s)$.
6. Возвратить Z.

В этом алгоритме $\mathtt{Pad}_{128}$ — описанное в начале главы расширение неполного блока до полного (с выбором $\alpha = 1$ и $\beta = 0$),

$$\varphi_1(x_1 \parallel x_2 \parallel x_3 \parallel x_4) = x_2 \parallel x_3 \parallel x_4 \parallel (x_1 \oplus x_2),$$
$$\varphi_2(x_1 \parallel x_2 \parallel x_3 \parallel x_4) = (x_1 \oplus x_4) \parallel x_1 \parallel x_2 \parallel x_3, \quad x_i \in \{0,1\}^{32}.$$

Отметим, что φ_i выбраны так, что каждое из преобразований

$$x \mapsto \varphi_1(x) \oplus x, \quad x \mapsto \varphi_2(x) \oplus x, \quad x \mapsto \varphi_1(x) \oplus \varphi_2(x), \quad x \mapsto \varphi_1(x) \oplus \varphi_2(x) \oplus x$$

является биекцией.

Схема Вигмана — Картера. Пусть определено инъективное отображение, которое ставит в соответствие сообщениям X многочлены f_X над некоторым конечным полем k. Пусть H — случайный равновероятный элемент k. Если $\deg(f_X) \leqslant D \ll |k|$, то значения $f_X(H)$ могут совпасть только с контролируемо малой вероятностью:

$$\mathsf{P}\{f_X(H) = f_{X'}(H)\} = \mathsf{P}\{H \text{ — корень } f_X - f_{X'}\} \leqslant \frac{\deg(f_X - f_{X'})}{|k|} \leqslant \frac{D}{|k|} \ll 1.$$

Это наблюдение было использовано в 1981 г. М. Вигманом и Дж. Картером для организации имитозащиты. Существует несколько вариантов реализации базовой схемы Вигмана — Картера. Мы рассмотрим один из них, использованный в системе GHASH.

Пусть $n_m = n_b$ и $k = \mathbb{F}_{2^{n_b}}$. Сообщение X разобъем на блоки $X_1, \ldots, X_n \in \{0,1\}^{n_b}$, которые будем интерпретировать как элементы k. Если длина X не кратна n_b, то дополним X до границы блока нулевыми символами. Сформируем дополнительный блок X_{T+1}, который представляет первоначальную (до дополнения) длину X. По X строится многочлен

$$f_X(\lambda) = X_1\lambda^{T+1} + X_2\lambda^T + \ldots + X_T\lambda^2 + X_{T+1}\lambda.$$

Различным X соответствуют различные последовательности $(X_1, \ldots, X_T, X_{T+1})$, и отображение $X \mapsto f_X$ действительно является инъективным.

Для $H \in k$ значение $Y = f_X(H)$ можно найти по схеме Горнера. Для этого следует установить $Y \leftarrow 0$ и выполнить следующие итерации:

$$Y \leftarrow (Y \oplus X_t) * H, \quad t = 1, 2, \ldots, T+1.$$

Эти итерации похожи на вычисления в режиме CBC, только вместо шифрования выполняется умножение на H.

Имитовставка сообщения X определяется следующим образом:

$$Z = f_X(H) \oplus F_\theta(S).$$

Здесь $H = F_\theta(0^{n_b})$ (интерпретируется как случайный секретный элемент k), $S \in \{0,1\}^{n_b}$.

В выражении для имитовставки зависимое от сообщения значение $f_X(H)$ «зашумляется» независимым от сообщения значением $F_\theta(S)$. Доказано, что если F — надежная криптосистема, а синхропосылки S не повторяются, то система имитозащиты также криптографически надежна. Однако, как только происходит повтор синхропосылок, и Виктор получает две имитовставки

$$Z = f_X(H) \oplus F_\theta(S), \quad Z' = f_{X'}(H) \oplus F_\theta(S), \quad X \neq X',$$

он может решить полиномиальное уравнение $f_X(H) \oplus f_{X'}(H) = Z \oplus Z'$ относительно секретного значения H. Поэтому требование неповторяемости синхропосылок критически важно в системах типа GHASH.

3.16. Задания

Задание 3.1. Разработать способ представления числа $n \in \{1, 2, \ldots, n_b\}$ словом из A^{n_b} (A — произвольный алфавит).

Задание 3.2. Пусть a и b – случайные независимые слова с равномерным на $\{0,1\}^n$ распределением, $c = a \boxplus b$. Доказать, что

$$\mathsf{P}\{c_i = a_i \oplus b_i\} = \frac{1}{2} + \frac{1}{2^{n-i+1}}, \quad i = 1, \ldots, n.$$

Задание 3.3. Доказать, что всякое простое число Ферма имеет вид $p = 2^{2^k} + 1$, где k — неотрицательное целое.

Задание 3.4. Доказать, что для любых $f, g, h \in \mathcal{F}_n$ выполняется неравенство треугольника: $\rho(f,g) \leqslant \rho(f,h) + \rho(g,h)$.

Задание 3.5. Доказать, что всякая отличная от константы аффинная функция уравновешенна.

Задание 3.6. Доказать, что функция $f(x) = g(x_1, \ldots, x_m) \oplus h(x_{m+1}, \ldots, x_n)$ является уравновешенной, если таковой является g или h.

Задание 3.7. Найти число уравновешенных функций от n переменных.

Задание 3.8. Определить числа $C_d = |\{f \in \mathcal{F}_n \colon \deg(f) = d\}|$, $d = 1, 2, \ldots, n$.

Задание 3.9 (существенные переменные). Переменная x_i является *существенной* для $f \in \mathcal{F}_n$, если найдется аргумент x, для которого

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).$$

Доказать, что число функций f, существенных от всех своих переменных, равняется

$$\sum_{i=0}^{n} (-1)^i \binom{n}{i} 2^{2^{n-i}}.$$

Задание 3.10. Пусть $f(x) = g(x_1) + h(x_2)$, $x = (x_1, x_2)$, $x_i \in \mathbb{F}_2^{n_i}$. Выразить коэффициент Уолша — Адамара $\hat{f}(u)$ через коэффициенты $\hat{g}(u_1)$ и $\hat{h}(u_2)$, $u = (u_1, u_2)$, $u_i \in \mathbb{F}_2^{n_i}$.

Задание 3.11 (симметрические функции). Пусть g — функция с аргументом $x \in \mathbb{F}_2^n$. Если перестановка координат x не изменяет значения g, то g — *симметрическая* функция. Доказать, что если $f \in \mathcal{F}_n$ — симметрическая функция, то и $\hat{f}$ — симметрическая функция.

Задание 3.12. Найти число симметрических функций $f \in \mathcal{F}_n$.

Задание 3.13. Доказать, что для $\sigma \in \mathcal{F}_{n,m}$ характеристики $\mathrm{nl}(\sigma)$ и $R_{\oplus\oplus}(\sigma)$ не изменятся, если к прообразам или к образам σ применить обратимое аффинное преобразование.

Задание 3.14. Пусть $s, s^{-1} \in S(\mathbb{F}_2^n)$ — взаимно обратные подстановки. Доказать, что $R_{\oplus\oplus}(s) = R_{\oplus\oplus}(s^{-1})$, $\mathrm{nl}(s) = \mathrm{nl}(s^{-1})$.

Задание 3.15. Пусть $\mu_{ab}(\sigma)$ — элементы таблицы разностей для отображения $\sigma \in \mathcal{F}_{n,m}$ при выборе операций $+ = \oplus$ и $+' = \oplus$. Доказать:

1) $\mu_{ab}(\sigma)$ — четные числа;

2) $\sum_b \mu_{ab}(\sigma) = 2^n$;

3) если $n = m$ и σ — подстановка, то $\sum_a \mu_{ab} = 2^n$.

(Последние два свойства означают, что матрица $(2^{-n}\mu_{ab}(\sigma))$ — дважды стохастическая.)

Задание 3.16. Пусть $s \in S(\mathbb{F}_2^n)$ и $\epsilon_{ab}(s)$ — преобладание линейной аппроксимации (a, b) для s. Доказать, что матрица $(4\epsilon_{ab}(s)^2)$ — дважды стохастическая.

Задание 3.17. Пусть $s(x) = x^d$, $d \in \mathbb{N}$, $x \in \mathbb{F}_{2^n}$. Доказать, что s — биекция тогда и только тогда, когда $(d, 2^n - 1) = 1$. Доказать, что $R_{\oplus\oplus}(s) \leqslant d - 1$.

Задание 3.18. Пусть $s \in S(\mathbb{F}_2^2)$. Доказать, что $\mathrm{nl}(s) = 0$.

Задание 3.19. Пусть $F = \{F_\theta \colon \theta \in \Theta\} \subseteq S(\mathbb{F}_2^n)$ — блочная криптосистема. Пусть $F_\theta(X) = XA_\theta + b_\theta$, где A_θ — обратимая $n \times n$ матрица над полем $\mathbb{F}_2$, $b_\theta \in \mathbb{F}_2^n$. Провести на F атаку по решению задачи **C2** при известном открытом тексте.

Задание 3.20. Пусть в SP-криптосистеме используются S-блоки $S_1, \ldots, S_r \in S(\mathbb{F}_2^m)$, которые являются аффинными преобразованиями, т. е. $S_i(x) = xA_i + b_i$, где A_i — обратимая матрица порядка m, $b_i \in \mathbb{F}_2^m$. Провести на C атаку по решению задачи **C2**. Разрешается выбирать открытые тексты.

Задание 3.21. Пусть тактовая подстановка имеет вид

$$\Sigma_\kappa(X_1 \parallel X_2 \parallel X_3) = f_\kappa(X_2) \parallel (X_3 \boxplus f_\kappa(X_1 \boxminus X_3)) \parallel (X_1 \boxplus X_2 \boxplus f_\kappa(X_1 \boxminus X_3)),$$

где $X_i \in \{0,1\}^n$, f_κ — некоторое преобразование $\{0,1\}^n$. Какие условия следует наложить на f_κ, чтобы Σ_κ действительно являлась биекцией? Как действует обратная подстановка Σ_κ^{-1}?

Задание 3.22. В блочной криптосистеме `Skipjack` по тактовой функции $G_\kappa \in S(\{0,1\}^{16})$ строится тактовая подстановка

$$A_\kappa(X_1 \parallel X_2 \parallel X_3 \parallel X_4) = (X_4 \oplus G_\kappa(X_1)) \parallel G_\kappa(X_1) \parallel X_2 \parallel X_3, \quad X_i \in \{0,1\}^{16}.$$

Описать обратную подстановку A_κ^{-1}. Найти неподвижные точки A_κ, т. е. такие прообразы $X \in \{0,1\}^{64}$, что $A_\kappa(X) = X$.

Задание 3.23. В блочной криптосистеме `Skipjack` по тактовой функции $G_\kappa \in S(\{0,1\}^{16})$ строится тактовая подстановка

$$B_\kappa(X_1 \parallel X_2 \parallel X_3 \parallel X_4) = (X_4 \parallel G_\kappa(X_1) \parallel (G_\kappa(X_1) \oplus X_2) \parallel X_3, \quad X_i \in \{0,1\}^{16}.$$

Описать обратную подстановку B_κ^{-1}. Найти неподвижные точки B_κ.

Задание 3.24. В блочной криптосистеме `SMS4` (Китай) по тактовой функции $G_\kappa \in S(\{0,1\}^{32})$ строится тактовая подстановка

$$\Sigma_\kappa(X_1 \parallel X_2 \parallel X_3 \parallel X_4) = X_2 \parallel X_3 \parallel X_4 \parallel X_1 \oplus (G_\kappa(X_2 \oplus X_3 \oplus X_4)), \quad X_i \in \{0,1\}^{32}.$$

Описать обратную подстановку Σ_κ^{-1}. Найти τ, при котором Σ_κ является τ-инволютивной подстановкой. Найти неподвижные точки Σ_κ.

Задание 3.25. Пусть тактовая подстановка блочно-итерационной криптосистемы имеют вид:

$$\Sigma_\kappa(X_1 \parallel X_2 \parallel X_3) = (X_3 \oplus f_\kappa(X_1 \oplus X_2)) \parallel X_1 \parallel X_2, \quad X_i \in \mathbb{F}_2^n, \quad f_\kappa \in \mathcal{F}_{n,n}.$$

Найти подстановку τ, при которой Σ_κ является τ-инволютивной.

Задание 3.26. Пусть $f_\kappa \in S(\{0,1\}^m)$ — тактовая функция и Σ_κ — соответствующая подстановка Фейстеля. Описать все неподвижные точки Σ_κ.

Задание 3.27. В SA-криптосистеме `Serpent` $n_b = 128$, $m = 4$. Действие линейное преобразования A задается следующими вычислениями над вектором $X_0 \parallel X_1 \parallel X_2 \parallel X_3$, $X_i \in \{0,1\}^{32}$:

1) $X_0 \leftarrow X_0 \lll 13$;
2) $X_2 \leftarrow X_2 \lll 3$;
3) $X_1 \leftarrow X_1 \oplus X_0 \oplus X_2$;
4) $X_3 \leftarrow X_3 \oplus X_2 \oplus (X_0 \ll 3)$;
5) $X_1 \leftarrow X_1 \lll 1$;
6) $X_3 \leftarrow X_3 \lll 7$;
7) $X_0 \leftarrow X_0 \oplus X_1 \oplus X_3$;
8) $X_2 \leftarrow X_2 \oplus X_3 \oplus (X_1 \ll 7)$;
9) $X_0 \leftarrow X_0 \lll 5$;
10) $X_2 \leftarrow X_2 \lll 22$.

Здесь $a \ll d$ — обычный (не циклический) сдвиг слова a с записью 0 в освобождающиеся разряды. Построить матрицу преобразования A и определить среднее число нулевых элементов в ее столбцах, т. е. среднее число несущественных переменных координатных функций A. Построить алгоритм вычисления образов обратного преобразования L^{-1}.

Задание 3.28 (время размножения ошибки). Временем размножения ошибки блочно-итерационной криптосистемы F называется минимальное r такое, что при некотором выборе тактовых ключей $\kappa_1, \ldots, \kappa_r$ ни одна из координатных булевых функций подстановки $s = \Sigma_{\kappa_r} \ldots \Sigma_{\kappa_1} \in \mathcal{F}_{n,n}$ не имеет несущественных переменных. Другими словами, при использовании r-тактового криптопреобразования изменение (ошибка) в любом символе открытого текста может привести к изменению любого символа шифртекста. Оценить время размножения ошибки криптосистемы $\mathtt{G}$ при различных величинах циклического сдвига в тактовой функции. Является ли сдвиг на 3 оптимальным?

Задание 3.29 (слабые ключи). Ключ θ криптосистемы F называется слабым, если $F_\theta = F_\theta^{-1}$ или, другими словами, F_θ – инволютивная подстановка. Пусть F – d-тактовая криптосистема Фейстеля и пусть ключу θ соответствуют тактовые ключи κ_i такие, что $\kappa_i = \kappa_{d+1-i}$, $i = 1, \ldots, d$. Доказать, что θ – слабый ключ.

Задание 3.30 (полуслабые ключи). Пара ключей θ, θ' криптосистемы F называется полуслабой, если $F_\theta = F_{\theta'}^{-1}$. Пусть F – d-тактовая криптосистема Фейстеля и пусть ключам θ и θ' соответствуют тактовые ключи κ_i и κ'_i такие, что $\kappa_i = \kappa'_{d+1-i}$, $i = 1, \ldots, d$. Доказать, что θ, θ' — пара полуслабых ключей.

Задание 3.31. В криптосистеме Фейстеля $\mathtt{CAST}$ используется ключ $\theta = \theta_1 \parallel \ldots \parallel \theta_8$, $\theta_i \in \{0,1\}^8$. Расписание ключей:

$$\mathtt{KS}(\theta) = (\theta_1 \parallel \theta_2,\ \theta_3 \parallel \theta_4,\ \theta_5 \parallel \theta_6,\ \theta_7 \parallel \theta_8,\ \theta_4 \parallel \theta_3,\ \theta_2 \parallel \theta_1,\ \theta_8 \parallel \theta_7,\ \theta_6 \parallel \theta_5).$$

Найти слабые и пары полуслабых ключей $\mathtt{CAST}$.

Задание 3.32. В криптосистеме $\mathtt{GOST}$ используется ключ $\theta = K_1 \parallel \ldots \parallel K_8$, $K_i \in \{0,1\}^{32}$. Расписание ключей:

$$\mathtt{KS}(\theta) = (K_1, K_2, \ldots, K_8, K_1, K_2, \ldots, K_8, K_1, K_2, \ldots, K_8, K_8, \ldots, K_2, K_1).$$

Найти слабые и пары полуслабых ключей $\mathtt{GOST}$.

Задание 3.33. В криптосистеме $\mathtt{TripleDES}$ используется ключ $\theta_1 \parallel \theta_2 \parallel \theta_3$, $\theta_i \in \{0,1\}^{56}$, преобразование зашифрования имеет вид $\mathtt{DES}_{\theta_3}\mathtt{DES}^{-1}_{\theta_2}\mathtt{DES}_{\theta_1}$. Оценить среднее число лет, которое понадобится для атаки «грубой силой» на $\mathtt{TripleDES}$, предполагая, что:

1) используется специализированное устройство RIVYERA, которое проверяет 292 млрд ключей $\mathtt{DES}$ в секунду;
2) согласно эвристическому закону Мура, описывающему темпы роста производительности вычислительной техники, мощность RIVYERA будет удваиваться каждые 18 месяцев.

Задание 3.34. Расписание ключей $\mathtt{DES}$ обладает следующим свойством: если $\mathtt{KS}(\theta) = (\kappa_1, \ldots, \kappa_{16})$, то $\mathtt{KS}(\bar{\theta}) = (\bar{\kappa}_1, \ldots, \bar{\kappa}_{16})$, где черта обозначает инверсию символов двоичных слов (замена 0 на 1 и 1 на 0). Доказать простое соотношение для $\mathtt{DES}$: $F_\theta(X) = \bar{F}_{\bar{\theta}}(\bar{X})$.

Задание 3.35. Найти простое соотношение для $\mathtt{GOST}$ (расписание ключей описано выше).

Задание 3.36. Пусть в криптосистеме Фейстеля, действующей на $\{0,1\}^{2m}$, используются биективные тактовые функции. Доказать, что для любого ненулевого $\gamma \in \{0,1\}^m$ вероятность 5-тактового дифференциала $(\gamma \parallel \mathbf{0}, \mathbf{0} \parallel \gamma)$ равняется 0.

Задание 3.37. Пусть в криптосистеме Фейстеля, действующей на $\{0,1\}^{2m}$, используются биективные тактовые функции. Доказать, что для любого ненулевого $\gamma \in \{0,1\}^m$ преобладание 5-тактовой линейной аппроксимации $(\mathbf{0} \parallel \gamma, \gamma \parallel \mathbf{0})$ равняется 0.

Задание 3.38. В каких из режимов шифрования (ECB, CBC, CFB, OFB, CTR) биективность преобразования F_θ не обязательна для однозначного расшифрования?

Задание 3.39. В режиме OFB используемая подстановка $F_\theta \in S(\{0,1\}^{n_b})$ должна обеспечивать большой период последовательности $\Gamma_t = F_\theta(\Gamma_{t-1})$, $t = 1, 2, \ldots$. Максимально большой период обеспечивает полноцикловая подстановка F_θ, для которой все элементы $\Gamma_0, \ldots, \Gamma_{2^{n_b}-1}$ различаются. Доказать, что среди элементов $S(\{0,1\}^{n_b})$ имеется $(2^{n_b}-1)!$ полноцикловых подстановок.

Задание 3.40. В ГОСТ 28147 для шифрования в режиме счетчика определена следующая функция инкремента:

$$\varphi(S_{t,1} \parallel S_{t,2}) = (S_{t,1} \boxplus' C_1) \parallel (S_{t,2} \boxplus C_2), \quad S_{t,i}, C_i \in \{0,1\}^{32}.$$

Здесь $\boxplus'$ — операция сложения слов-как-чисел по модулю $2^{32}-1$ (вычет 0 по этому модулю представляется не числом 0, как обычно, а числом $2^{32}-1$). Найти период последовательности (S_t) в зависимости от выбора констант C_1, C_2.

Задание 3.41. В режиме CBC синхропосылка должна быть не только уникальной, но и непредсказуемой. Обосновать данное требование. Предположить, что Виктор может выбирать открытый текст и до своего выбора знает, какая будет использоваться синхропосылка. Виктору требуется проверить, что блок открытого текста X_t, соответствующий перехваченному блоку шифртекста Y_t, совпадает с определенным значением a.

Задание 3.42. Проанализировать характеристики следующего режима шифрования: $Y_t = F_\theta(X_i) \oplus Y_{t-1}$, $t = 1, 2, \ldots$. Найти недостатки.

Задание 3.43. Имеется смарт-карта, которая реализует зашифрование блоков открытого текста X_1, X_2 на ключе θ и синхропосылке S следующим образом:

$$Y_1 = X_1 \oplus F_\theta(S);$$
$$Y_2 = X_1 \oplus X_2 \oplus F_\theta(X_1) \oplus F_\theta(S).$$

Виктор получает тройку (S, Y_1, Y_2) и смарт-карту, с помощью которой он может зашифровывать любые данные. Требуется определить X_1, X_2 (ключ θ Виктору неизвестен).

Задание 3.44. Пусть блоки открытого текста $X_1, X_2, \ldots, X_T$ зашифровываются по правилам:

$$Y_t = F_\theta(X_1 \boxplus X_2 \boxplus \ldots \boxplus X_t) \oplus F_\theta(Y_0 \boxminus Y_1 \boxminus \ldots \boxminus Y_{t-1}), \quad t = 1, 2, \ldots, T$$

(Y_0 — синхропосылка). Как выполнить расшифрование?

Задание 3.45. Для проверки подлинности друг друга Алиса и Боб используют общий секретный ключ θ блочной криптосистемы F, действующей на $\{0,1\}^{n_b}$. Стороны выполняют следующий протокол:

$$\begin{aligned}
\text{Алиса: } & R_A \xleftarrow{R} \{0,1\}^{n_b};\\
\text{Алиса} \to \text{Боб: } & R_A;\\
\text{Боб: } & R_B \xleftarrow{R} \{0,1\}^{n_b};\\
\text{Алиса} \leftarrow \text{Боб: } & E_\theta(R_B \parallel R_A);\\
\text{Алиса} \to \text{Боб: } & E_\theta(R_A \parallel R_B)
\end{aligned}$$

Здесь E_θ — зашифрование в режиме CBC на основе криптосистемы F. При зашифровании используется нулевая синхропосылка. После получения зашифрованных сообщений каждая из сторон расшифровывает их и проверяет, что полученное слово R_A (для Алисы) или R_B (для Боба) совпадает с словом, первоначально сгенерированным стороной. Если это так, то стороны признают подлинность друг друга. Провести атаку на протокол.

Задание 3.46 (атаки по времени). В криптосистеме IDEA при зашифровании блока данных 34 раза выполняется операция $\odot$ (умножение в $\mathbb{F}^*_{65537} \sim \{0,1\}^{16}$). Пусть для вычисления $a \odot b$ используется следующая программа на языке Си:

```
uint32 Mul(uint32 a, uint32 b)
{ int32 p;
  uint32 q;
  if (a == 0)                       // (*)
    p = 0x10001 - b;
  else if (b == 0)                  // (*)
    p = 0x10001 - a;
  else {                            // (**)
    q = a * b;
    p = (q & 0xFFFF) - (q >> 16);
    if (p <= 0)
      p += 0x10001;
  }
  return (uint32)(p & 0xFFFF);
}
```

В этой программе компоненты операндов хранятся в младших разрядах 4-байтовых целых переменных беззнаковых / знаковых типа uint32 / int32. Для вычисления результата в условиях (*) и (**) программы требуется выполнить различное число инструкций. Если Виктор имеет возможность измерять время зашифрования, то он может оценить число встретившихся нулевых операндов и упростить тем самым решение задачи криптоанализа. Предложить модификацию программы, которая позволяет защититься от описанной атаки.

3.17. Комментарии

Блочные криптосистемы описываются в книгах [2, 37, 41, 43]. Книга [11] рассчитана не только на студентов и преподавателей, но и на школьников старших классов, в ней имеется много интересных исторических подробностей и фотографий.

Огюст Керкгоффс (Kerkhoffs), голландский криптограф, написал в 1883 году книгу «Военная криптография». В этой книге введено правило: *компрометация (криптографической) системы не должна причинять неудобства корреспондентам*, которое впоследствии трансформировалось в принцип Керкгоффса.

Русскоязычная терминология в области блочных криптосистем не до конца устоялась. Вместо «такты» и «тактовый» говорят «раунды» и «раундовый», что кажется автору не очень удачным. Вместо «S-блоки» говорят «S-боксы», что кажется совсем неприемлемым.

Булевы функции и отображения — это отдельная область исследования, подробно рассмотренная в [6]. Теория конечных полей изложена в классической книге [9].

Инверсные S-блоки введены в работах [17, 38]. Теорема 3.5 доказана в [33]. Пункт 3.5.10 основан на материалах статьи [1].

Некоторые исследователи считают оценки таблицы 3.3 весьма пессимистичными (для Алисы и Боба). Имеются расчеты, согласно которым для перебора уже 2^{128} вариантов ключа требуется практически недостижимое количество энергии.

Баланс «время – память» введен в работе [28]. Разностная атака предложена в работе [18], линейная — в [35]. Подробные примеры 3.13, 3.15 принципиальны. Автор считает, что методы криптоанализа должны обязательно иллюстрироваться исчерпывающими «историями успеха».

В США при введении DES был выпущен стандарт FIPS PUB 81, в котором определялись режимы CBC, CFB и OFB (Output Feedback), лишенные недостатков ECB. Со временем наибольшее распространение из этих режимов получил CBC. В частности, шифрование CBC по умолчанию используется в протоколах SSL/TLS, которые широко применяются для защиты каналов Интернет (см. § 5.9). Популярность CBC до конца непонятна. Возможно, она объясняется тем, что в FIPS PUB 81 режимы CFB и OFB были представлены как поточные методы шифрования, а CBC оставлен блочным. При разработке криптонаборов на основе блочной криптосистемы F отдавать предпочтение поточным методам кажется нелогичным. Кроме этого, в FIPS PUB 81 режимы CFB и OFB перегружены и представляют собой целые параметрические семейства. Необходимость учитывать в криптонаборах дополнительные параметры этих семейств могла являться дополнительным аргументом в пользу CBC.

В СССР при введении ГОСТ 28147 были определены три режима шифрования: простой замены, гаммирования с обратной связью и гаммирования. Наибольшее распространение получил режим гаммирования с обратной связью — аналог CFB. В ГОСТ 28147-89, в отличие от FIPS PUB 81, параметризация CFB отсутствует, режим всегда является полноблоковым.

Отличия между режимами выглядят несущественными. Но это только на первый взгляд. Изменение правил зашифрования приводит к изменению двух важных показателей (сравните с табл. 3.4):

Показатель	CBC	CFB
Можно обрабатывать открытые тексты с последним неполным блоком?	нет	да
Синхропосылка должна быть непредсказуемой?	да	нет

По обоим показателям CFB предпочтительнее CBC. Во-первых, в режиме CFB можно обрабатывать открытые тексты любой длины. Напротив, в CBC длина открытого текста должна быть кратна длине блока. Казалось бы, этот недостаток легко преодолевается выравниванием данных на границу блока перед зашифрованием и снятием выравнивания после расшифрования (см. § 3.1). При этом если полученный после расшифрования текст не удовлетворяет формату выравнивания, то логично его отбросить, а отправителю выслать сообщение об ошибке. Такая схема обработки открытых текстов произвольной длины была применена в старых версиях SSL/TLS и оказалась уязвимой. Выяснилось, что противник может расшифровать любой шифртекст, используя другие специально подобранные шифртексты и анализируя сообщения о нарушениях формата их выравнивания после расшифрования. В последних версиях TLS выравнивание сохранено, но сообщение о нарушении формата выравнивания не отсылается. Во-вторых, в режиме CFB достаточно обеспечить уникальность синхропосылки, обеспечивать ее непредсказуемость не требуется. В CBC ситуация другая. В старых версиях SSL/TLS синхропосылка при CBC-зашифровании очередного пакета данных определялась как последний блок шифртекста из предыдущего пакета и, таким образом, была известна противнику. Оказалось, что противник может использовать знание синхропосылки для проверки того, что перехваченному блоку шифртекста соответствует открытый текст с определенным значением (см. упр. 3.41). Правда для этого противник должен иметь возможность навязывать открытый текст, который будет передан в очередном пакете. Тем не менее в последних версиях TLS синхропосылка выбирается как случайное и, следовательно, непредсказуемое слово.

Описанные в § 3.15 схемы построения систем имитозащиты предложены в работах [29, 44].

Глава 4
ФУНКЦИИ ХЭШИРОВАНИЯ

4.1. Определение и использование

Пусть Бобу требуется подписать сообщение $X \in \{0,1\}^*$. Боб сталкивается со следующей проблемой: алгоритм выработки ЭЦП принимает на вход слова фиксированной длины n, хотя длина X может быть произвольной. Выходом в данной ситуации является использование функции h, которая ставит в соответствие сообщению X слово $Y \in \{0,1\}^n$. Боб использует h и подписывает не X, а Y. Важно при этом, чтобы алгоритм вычисления значений $h(X)$ имел высокое быстродействие, по крайней мере был полиномиальным (от $|X|$). Важно также, чтобы этот алгоритм был общедоступным: и Алиса, и другие абоненты информационной системы будут проверять подпись Боба и при этом также вычислять $h(X)$.

Первоначально выходное слово $h(X)$ называли *отпечатком* (Digest) входного сообщения (Message), а h, соответственно, MD-функцией. Затем терминология изменилась.

Определение 4.1. *Функция хэширования (хэш-функция) — это отображение* $h\colon \{0,1\}^* \to \{0,1\}^n$*, действие которого задается общедоступным полиномиальным алгоритмом.*

Со временем функции хэширования стали использоваться не только для построения систем ЭЦП, они превратились в один из основных криптографических примитивов. Укажем наиболее важные примеры использования хэш-функций.

Контрольные суммы. Алиса вычисляет и сохраняет (с мерами защиты от модификаций) хэш-значение файла $X \in \{0,1\}^*$. Последующее совпадение сохраненного хэш-значения с $h(X)$ служит подтверждением того, что файл X не был изменен.

Например, в дистрибутивы многих операционных систем входит программа `md5sum`. Программа реализует весьма распространенный в недавнем времени алгоритм хэширования `MD5`. Программа принимает на вход файл и возвращает его хэш-значение (16 октетов).

Построение ключей. По паролю $X \in \{0,1\}^*$ Алиса строит секретный ключ $\theta = h(X)$. Этот ключ Алиса использует для шифрования или имитозащиты своих критических данных (например, контрольных сумм).

Число возможных паролей может быть сравнительно небольшим. Поэтому в процедуру построения ключей вводятся дополнительные механизмы, направленные на защиту от атаки «грубой силой», направленной на определение X. Во-первых, Алиса применяет h не один, а несколько раз:

$$\theta = h^c(X) = \underbrace{h(h(\ldots h}_{c \text{ раз}}(X)\ldots)).$$

Регулируя число итераций c, можно сделать неприемлемо большим время, которое требуется Виктору для перебора паролей, оставляя допустимым время, затрачиваемое Алисой на генерацию ключа. Во-вторых, при построении θ Алиса кроме пароля использует синхропосылку S (ее еще называют «соль»). Ключ θ зависит от выбранной синхропосылки, и Виктор лишается возможности предварительно рассчитывать ключи для определенных классов паролей, т. е. проводить так называемые *словарные атаки*. Число итераций c и синхропосылка S являются несекретными элементами и могут сохраняться вместе с защищенными на θ данными.

Аутентификация. Алиса регистрируется на сервере Боба и отсылает ему свой пароль X по секретному каналу связи. Перед тем как предоставить Алисе доступ к ресурсам сервера, Боб проводит ее аутентификацию. Аутентификация основана на проверке знания X. Боб пересылает Алисе случайное слово R, Алиса возвращает $Y = h(X \,\|\, R)$. Аутентификация завершена успешно, если полученное Бобом слово действительно совпадает с $h(X \,\|\, R)$.

Похожим образом выполняется, например, протокол `NTLM`. Протокол имеет недостаток: Виктор может перехватить (R, Y) и определить X «грубой силой», проверяя совпадение $Y \stackrel{?}{=} h(\hat{X} \,\|\, R)$ для паролей-кандидатов $\hat{X}$. Известны другие протоколы аутентификации, в которых Виктор по данным перехвата не в состоянии определить пароль X, даже если он короткий или низкоэнтропийный.

Имитозащита. Алиса преобразует функцию h в систему имитозащиты $H = \{h_\theta \colon \theta \in \Theta\}$, где h_θ действует из $\{0,1\}^*$ в $\{0,1\}^n$ (подробнее см. § 3.15). Алиса строит H так, что значение $h_\theta(X)$ является результатом применения h (возможно многократного) к X и θ.

Например, в известной системе имитозащиты `HMAC` при использовании определенных h и для $\theta \in \{0,1\}^n$ выполняется 2-кратное хэширование:

$$h_\theta(X) = h((\theta \oplus \alpha) \,\|\, h((\theta \oplus \beta) \,\|\, X)).$$

Здесь $\alpha, \beta \in \{0,1\}^n$ — различные фиксированные слова.

Генерация псевдослучайных чисел. Алиса комбинирует свой секретный ключ θ с неповторяющейся синхропосылкой S и вычисляет хэш-значение $Y = h(\theta \,\|\, S)$, которое интерпретируется как псевдослучайное число. Это число Алиса использует для построения других секретных или личных ключей.

Детали генерации могут отличаться. Например, в СТБ 34.101.47-2012 «Информационные технологии и безопасность. Криптографические алгоритмы генерации псевдослучайных чисел» определен следующий алгоритм.

Алгоритм Генерация псевдослучайных чисел (СТБ 34.101.47)

Вход: $T \in \mathbb{N}$, $\theta \in \{0,1\}^n$ — ключ, $S \in \{0,1\}^n$ — синхропосылка, $X_1, \ldots, X_T \in \{0,1\}^n$ — произвольные входные данные, которые могут повысить неопределенность выходных (текущее время, сетевая активность, данные от физических источников случайности и др.), или нулевые блоки.

Выход: $Y_1, \ldots, Y_T \in \{0,1\}^n$ — псевдослучайные числа.

Шаги:

1. $s \leftarrow S$.
2. $r \leftarrow S \oplus 1^n$.
3. Для $t = 1, 2, \ldots, T$:
   (1) $Y_i \leftarrow h(\theta \,\|\, s \,\|\, X_t \,\|\, r)$;
   (2) $s \leftarrow s \boxplus \langle 1 \rangle_n$;
   (3) $r \leftarrow r \oplus Y_t$.
4. $Y \leftarrow Y_1 \,\|\, Y_2 \,\|\, \ldots \,\|\, Y_T$.
5. Возвратить Y.

Здесь и далее используются обозначения, введенные в гл. 3. Как и в криптосистеме `Belt`, числа представляются двоичными словами по правилам «от младших к старшим».

4.2. Задачи криптоанализа

Виктору требуется решить одну из следующих задач:

H1: по заданному $Y = h(X)$ определить X;

H2: для заданного X найти $X' \neq X$ такое, что $h(X) = h(X')$;

H3: найти различные X и X' такие, что $h(X) = h(X')$.

В таблице 4.1 приводятся общепринятые названия этих задач, объясняется мотивация атак по решению задач, характеризуются функции хэширования, атаки на которые провести вычислительно трудно.

Обратим внимание на фразу «вычислительно трудно». Дело в том, что фиксированная длина хэш-значений не позволяет дать такие же строгие определения криптографической стойкости, как, к примеру, определение односторонности в главе 2. Мы говорим «вычислительно трудно», имея в виду «невозможно решить за приемлемое время при современном уровне развития вычислительной техники».

Таблица 4.1. Задачи криптоанализа функций хэширования

Задача	Название	Мотивация атак	Стойкая хэш-функция
H1	Обращение	Виктор может определить пароль X по аутентификационным данным $h(X \parallel R)$	Односторонняя
H2	Определение 2-го прообраза	Виктор может осуществить подмену файла X' на файл X с такой же контрольной суммой	Свободная от коллизий
H3	Построение коллизии	Виктор может подобрать два различных документа — подлинный X и поддельный X' — с одинаковыми хэш-значениями. Виктор передает Алисе подлинный документ для ознакомления и выработки ЭЦП, а затем прилагает полученную подпись Алисы к поддельному документу	Строго свободная от коллизий

Всякий алгоритм, который находит 2-й прообраз, является также алгоритмом построения коллизии. Поэтому если h строго свободна от коллизий, то h свободна от коллизий. Следующая теорема показывает, что если h строго свободна от коллизий, то h является односторонней.

Теорема 4.1. *Пусть A — алгоритм, который решает задачу* **H1** *для h. Тогда существует вероятностный алгоритм, который с вероятностью не менее $1/2$ решает задачу* **H3**, *используя одно обращение к A и еще фиксированное количество операций.*

Доказательство. Заявленный алгоритм решения **H1** выглядит следующим образом:

1. $X \xleftarrow{R} \{0,1\}^{n+1}$.
2. $Y \leftarrow h(X)$.
3. $X' \leftarrow A(Y)$.
4. Если $X = X'$, то возвратить $\perp$.
5. Возвратить (X, X').

На шаге 3 алгоритм A возвращает слово X' такое, что $h(X') = Y = h(X)$. Поэтому пара, возвращаемая на шаге 5, действительно является решением **H3**.

Введем на $\{0,1\}^{n+1}$ отношение эквивалентности, считая эквивалентными слова с одинаковыми хэш-значениями. Пусть C — множество всех классов эквивалентности. Для $c \in C$ вероятность

$$\mathsf{P}\{X \neq A(h(X)) \mid X \in c\} \geqslant [A(h(X)) \notin \{0,1\}^{n+1} \text{ или } A(h(X)) \in c] \geqslant \frac{|c|-1}{|c|}.$$

Пусть p — вероятность успеха алгоритма (вероятность того, что будет возвращена коллизионная пара, а не символ $\perp$). Имеем

$$p = \mathsf{P}\{X \neq A(h(X))\} = \sum_{c \in C} \mathsf{P}\{X \neq M(h(X)) \mid X \in c\}\, \mathsf{P}\{X \in c\} \geqslant$$
$$\geqslant \sum_{c \in C} \frac{|c|}{2^{n+1}} \cdot \frac{|c|-1}{|c|} = 1 - \frac{|C|}{2^{n+1}} \geqslant \frac{1}{2}. \qquad \square$$

В обозначениях главы 2 сказанное выше означает, что $\mathbf{H3} \leqslant_P \mathbf{H2}$ и $\mathbf{H3} \leqslant_R \mathbf{H1}$. Задача **H3** является самой простой из трех рассмотренных.

4.3. Блочно-итерационные функции хэширования

Блочно-итерационные функции хэширования, как и блочные криптосистемы, обрабатывают данные блоками некоторой фиксированной длины n_b. Обработка выполняется с помощью *шаговой функции хэширования* $\sigma\colon \{0,1\}^{n_b+n} \to \{0,1\}^n$. Типовой алгоритм блочно-итерационного хэширования имеет следующий вид.

Алгоритм Блочно-итерационное хэширование

Вход: $X \in \{0,1\}^*$ — сообщение.
Выход: $Y \in \{0,1\}^n$ — хэш-значение.
Шаги:
1. Определить m — минимальное неотрицательное целое такое, что n_b делит $|X| + m$.
2. Разбить $X \parallel 0^m$ на блоки: $X_1 \parallel \ldots \parallel X_T \leftarrow X \parallel 0^m$, $X_t \in \{0,1\}^{n_b}$.
3. Сформировать дополнительный блок $X_{T+1} \in \{0,1\}^{n_b}$, представляющий число $|X|$.
4. $Y \leftarrow Y_0$, где $Y_0 \in \{0,1\}^n$ — фиксированное *начальное хэш-значение*.
5. Для $t = 1, 2, \ldots, T+1$: $Y \leftarrow \sigma(X_t \parallel Y)$.
6. Возвратить Y.

Обратим внимание на дополнительный блок X_{T+1}, который формируется на шаге 3. Обработку этого блока принято называть *усилением Меркля — Дамгарда*. Без усиления Виктор легко решает задачу **H3**, выбирая сообщения X и X', после дописывания нулей к которым на шаге 2 получаются одинаковые слова: $X \parallel 0^m = X' \parallel 0^{m'}$.

Детали алгоритма могут отличаться. Например, в схеме хэширования, предложенной И. Дамгардом, отличия следующие:
- последний блок X_{T+1} представляет не число $|X|$, а число m дописанных к X нулевых символов;
- шаговая функция хэширования σ действует на $\{0,1\}^{n_b+n+1}$, а не на $\{0,1\}^{n_b+n}$;
- итерации хэширования на шаге 5 немного меняются: сначала $Y \leftarrow (X_1 \parallel 0 \parallel Y)$, а затем $Y \leftarrow (X_t \parallel 1 \parallel Y)$, $t = 2, \ldots, T+1$.

Введенные для хэш-функции h задачи **H1**, **H2**, **H3** можно поставить также для шаговой функции хэширования σ. При обосновании стойкости h стараются показать, что решение некоторой задачи для h приводит к решению некоторой задачи для σ. Другими словами, пытаются свести одну задачу к другой. Если задача для σ трудна, то сведéние означает, что соответствующая задача для h также трудна. Примером подобного обоснования является следующая теорема.

Теорема 4.2. *Пусть в схеме Дамгарда шаговая функция хэширования σ строго свободна от коллизий. Тогда построенная на ее основе функция h также строго свободна от коллизий.*

Доказательство. Предположим от противного, что σ строго свободна от коллизий, а h — нет, и найдены различные $X, X' \in \{0,1\}^*$ такие, что $h(X) = h(X')$. Далее мы покажем, что по X и X' можно построить различные слова $x, x' \in \{0,1\}^{n_b+n+1}$, которые дают коллизию для шаговой функции хэширования: $\sigma(x) = \sigma(x')$. Тем самым мы получим противоречие и докажем нужный результат.

Через Y_t обозначим значение переменной Y после завершения t-й итерации хэширования X. Все выражения, касающиеся обработки X', снабдим штрихами. Пусть $\alpha_t = \mathbf{l}\{t > 1\}$. Рассмотрим три случая.

1. $|X| \not\equiv |X'| \pmod{n_b}$. Тогда $X_{T+1} \neq X'_{T'+1}$ и найдена коллизия:

$$\sigma(X_{T+1} \parallel \alpha_{T+1} \parallel Y_T) = h(X) = h(X') = \sigma(X'_{T'+1} \parallel \alpha_{T'} \parallel Y'_{T'}).$$

2. $|X| = |X'|$. Имеем

$$\sigma(X_{T+1} \parallel \alpha_{T+1} \parallel Y_T) = h(X) = h(X') = \sigma(X_{T+1} \parallel \alpha_{T+1} \parallel Y'_T).$$

Если $Y_T \neq Y'_T$, то мы получаем коллизию для σ. Если же $Y_T = Y'_T$, то перейдем к соотношению

$$\sigma(X_T \parallel \alpha_T \parallel Y_{T-1}) = Y_T = Y'_T = \sigma(X'_T \parallel \alpha_T \parallel Y'_{T-1}).$$

Если $X_T \neq X'_T$ или $Y_{T-1} \neq Y'_{T-1}$, то мы получаем коллизию. В случае двух равенств рассматриваем новое соотношение и т. д. Ясно, что в некоторый момент мы либо найдем коллизию для σ, либо придем к равенству $X = X'$, которое противоречит первоначальному предположению.

3. $|X| \equiv |X'| \pmod{n_b}$, $|X| < |X'|$. Будем рассуждать как в предыдущем случае. Рассматривая обработку блоков X_{T+1} и $X'_{T'+1}$, X_T и $X'_{T'}$ и далее, мы либо найдем коллизию, либо достигнем пары блоков X_1 и $X'_{T'-T+1}$ и получим соотношение вида

$$\sigma(X_1 \parallel 0 \parallel Y_0) = \sigma(X'_{T'-T+1} \parallel 1 \parallel Y'_{T'-T}).$$

Данное соотношение дает коллизию, поскольку (n_b+1)-е символы прообразов обязательно отличаются. □

4.4. Шаговые функции хэширования

Шаговые функции хэширования строят по тем же принципам, что и подстановки шифрования блочных криптосистем: многократно комбинируя преобразования усложнения и перемешивания. При построении шаговой функции $\sigma\colon \{0,1\}^{n_b+n} \to \{0,1\}^n$ можно, в отличие от подстановок шифрования, не заботиться о биективности и, наоборот, следует продумать организацию сжатия данных. Конечно, следует также обеспечить криптографическую стойкость и учесть требования односторонности и (строгой) свободы от коллизий.

Пусть $F = \{F_\theta\colon \theta \in \Theta\}$ — блочная криптосистема, которая действует на $\{0,1\}^m$ и имеет множество ключей $\Theta = \{0,1\}^l$. Эту криптосистему можно интерпретировать как сжимающее отображение $X \parallel \theta \mapsto F_\theta(X)$ и использовать для построения шаговой функции хэширования.

Пусть длина блока хэшируемых данных, длина хэш-значения, длина блока и длина ключа криптосистемы F совпадают: $n_b = n = m = l$. В этом случае σ можно строить по F по следующей схеме:

$$\sigma(X \parallel Y) = F_{\alpha_1 X \oplus \alpha_2 Y}(\beta_1 X \oplus \beta_2 Y) \oplus \gamma_1 X \oplus \gamma_2 Y, \tag{4.1}$$

где $X, Y \in \{0,1\}^n \sim \mathbb{F}_2^n$, $\alpha_i, \beta_i, \gamma_i \in \mathbb{F}_2$ — фиксированные константы.

Не всякий выбор констант дает криптографически стойкую функцию σ. Например, функция $\sigma(X \,\|\, Y) = F_Y(X \oplus Y) \oplus Y$ не является односторонней. Действительно, для заданного $Z \in \{0,1\}^n$ можно выбрать произвольное Y и определить $X = F_Y^{-1}(Z \oplus Y) \oplus Y$. При этом $\sigma(X \,\|\, Y) = Z$ и задача обращения решена.

Криптографы проанализировали всевозможные варианты выбора $\alpha_i, \beta_i, \gamma_i$ и определили 12 надежных конструкций шаговых функций хэширования вида (4.1). Эти конструкции приведены в таблице 4.2.

Таблица 4.2. Шаговые функции хэширования на основе блочной криптосистемы F

№	$\sigma(X \,\|\, Y)$	Название
1	$F_Y(X) \oplus X$	Матиаса — Мейера — Озеаса
2	$F_Y(X \oplus Y) \oplus X \oplus Y$	
3	$F_Y(X) \oplus X \oplus Y$	Миягучи — Приниля
4	$F_Y(X \oplus Y) \oplus X$	
5	$F_X(Y) \oplus Y$	Дэвиса — Мейера
6	$F_X(X \oplus Y) \oplus X \oplus Y$	
7	$F_X(Y) \oplus X \oplus Y$	
8	$F_X(X \oplus Y) \oplus Y$	
9	$F_{X \oplus Y}(X) \oplus X$	
10	$F_{X \oplus Y}(Y) \oplus Y$	LOKI
11	$F_{X \oplus Y}(X) \oplus Y$	
12	$F_{X \oplus Y}(Y) \oplus X$	

Длина блока m криптосистемы F может быть меньше n, и тогда описанные конструкции применить нельзя. Но все равно функцию σ можно строить на основе F, выполняя не одно, а c зашифрований и комбинируя результаты зашифрований с помощью перестановок и сложений.

При построении σ, кроме обеспечения криптографической стойкости стараются достичь максимальной скорости хэширования. Ее часто характеризуют величиной $n_b/(cm)$, которая примерно показывает, во сколько раз хэширование медленнее шифрования. Скорость хэширования всех конструкций, рассмотренных в таблице 4.2, равняется 1. Но для $n_b > m$ такой скорости добиться не удается без ущерба для криптографической стойкости.

В СТБ 34.101.31 определен алгоритм хэширования на основе блочной криптосистемы `Belt` ($m = 128$, $l = 256$). Длина блока хэшируемых данных $n_b = 256$, длина хэш-значения $n = 256$. Используется шаговая функция хэширования $\sigma\colon \{0,1\}^{512} \to \{0,1\}^{256}$, которая дает скорость хэширования 2/3. Функция σ определяется следующим алгоритмом (см. также рис. 4.1).

Алгоритм Шаговая функция хэширования (СТБ 34.101.31)

Вход: $u_1 \,\|\, u_2 \,\|\, u_3 \,\|\, u_4$, $u_i \in \{0,1\}^{128}$.
Выход: $v_1 \,\|\, v_2$, $v_i \in \{0,1\}^{128}$.
Шаги:

1. $\kappa \leftarrow \mathtt{Belt}_{u_1 \| u_2}(u_3 \oplus u_4) \oplus u_3 \oplus u_4$.
2. $\theta_1 \leftarrow \kappa \,\|\, u_4$.
3. $\theta_2 \leftarrow (\kappa \oplus 1^{128}) \,\|\, u_3$.
4. $v_1 \leftarrow \mathtt{Belt}_{\theta_1}(u_1) \oplus u_1$.
5. $v_2 \leftarrow \mathtt{Belt}_{\theta_2}(u_2) \oplus u_2$.

6. Возвратить $v_1 \| v_2$.

Слово κ, которое вычисляется на шаге 1 алгоритма, используется не только в σ, но еще для обновления служебного слова s, которое участвует в формировании дополнительного блока X_{T+1} при хэшировании X.

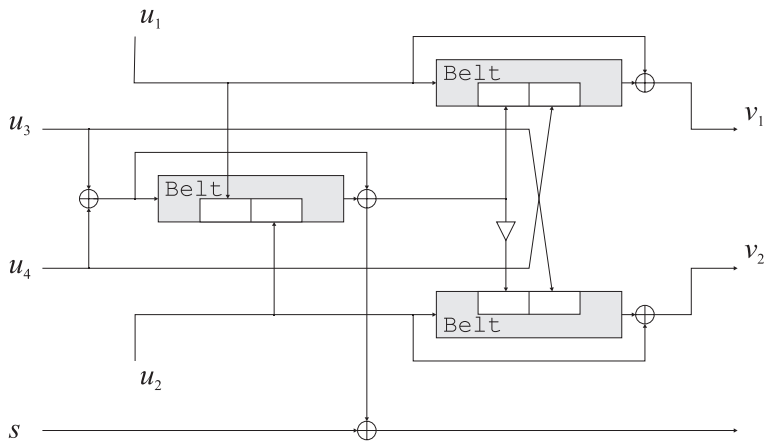


Рис. 4.1. Шаговая функция хэширования СТБ 34.101.31

4.5. Атака «дней рождения»

Атака «дней рождения» направлена на решение задачи **H3**, т. е. на нахождение коллизии. Атака является универсальной и может быть применена к произвольной функции хэширования h. Пусть, как обычно, хэш-значения — это слова длины n и пусть $N = 2^n$.

Алгоритм Атака «дней рождения»

Вход: h (задается описанием алгоритма хэширования).
Выход: $X, X' \in \{0,1\}^*$ — коллизионная пара ($X \neq X'$, $h(X) = h(X')$).
Шаги:

1. Выбрать конечное множество $\mathfrak{X} \subset \{0,1\}^*$ мощностью $|\mathfrak{X}| \gg N$.
2. Зарезервировать массив H из N ячеек памяти. В ячейках размещаются элементы $\mathfrak{X}$, ячейки индексируются словами из $\{0,1\}^n$: $H[Y]$ — ячейка по индексу Y. Первоначально все ячейки заполняются символом $\perp$ (пусто).
3. $X \xleftarrow{R} \mathfrak{X}$.
4. $Y \leftarrow h(X)$.
5. Если $H[Y] \neq \perp$ и $X \neq H[Y]$, то перейти к шагу 7.
6. $H[Y] \leftarrow X$ и перейти к шагу 3.
7. Возвратить $(X, H[Y])$.

Проанализируем среднее время атаки. Будем идеализировать функцию хэширования и считать, что вычисляемые в ходе атаки хэш-значения Y являются реализациями независимых случайных величин с равномерным распределением на $\{0,1\}^n$.

Хэш-значения Y можно интерпретировать как частицы, которые случайно независимо друг от друга размещаются в N ячеек. Пусть ν — номер первой частицы, которая попадает в уже занятую ячейку. Попадание означает, что $h(X) = h(X')$ для некоторых случайных $X, X' \in \mathfrak{X}$. Поскольку $|\mathfrak{X}| \gg N$, вероятность совпадения $X = X'$ пренебрежимо мала и попадание дает коллизионную пару (X, X'). При этом ν — *время ожидания*

коллизии, или время атаки «дней рождения», выраженное в количестве обращений к алгоритму h.

В следующей теореме мы получим точные и асимптотические выражения для $\mathsf{E}\nu$. Предварительно напомним, что через $N^{[t]} = N(N-1)\dots(N-t+1)$ обозначается t-я *факториальная степень* N (считается, что $N^{[0]} = 1$) и приведем без доказательства следующий результат.

Лемма 4.1 (метод Лапласа). *Пусть выполнены следующие условия:*
1) $[a, b]$ — *конечный отрезок,*
2) *функция* $S(x)$ *бесконечное число раз дифференцируема на* $[a, b]$,
3) $\max_{x\in[a,b]} S(x)$ *достигается при* $x = a$,
4) $S''(a) \neq 0$.

Тогда при $N \to \infty$

$$\int_a^b \exp(NS(x))dx = \sqrt{-\frac{\pi}{2NS''(a)}}e^{NS(a)}(1 + o(1)).$$

Теорема 4.3. *Среднее время ожидания коллизии*

$$\mathsf{E}\nu = \sum_{t=0}^{N} \frac{N^{[t]}}{N^t} = N\int_0^\infty e^{-Nx}(1+x)^N dx = \sqrt{\frac{\pi N}{2}}(1 + o(1))$$

(последнее равенство в асимптотике $N \to \infty$*).*

Доказательство. Наша цель — последовательно доказать три равенства из формулировки теоремы.

1. Имеется N^t способов размещения t частиц по N ячейкам. При этом $N^{[t]}$ способов размещения не приведут к появлению коллизии. Поэтому $\mathsf{P}\{\nu > t\} = N^{[t]}/N^t$ и

$$\mathsf{E}\nu = \sum_{t=1}^{\infty} t \cdot \mathsf{P}\{\nu = t\} = \sum_{t=0}^{\infty} \mathsf{P}\{\nu > t\} = \sum_{t=0}^{N} \frac{N^{[t]}}{N^t}.$$

2. Преобразуем интеграл (Γ — гамма-функция Эйлера):

$$\begin{aligned} N\int_0^\infty e^{-Nx}(1+x)^N dx &= \int_0^\infty e^{-x}\left(1+\frac{x}{N}\right)^N dx = \\ &= \int_0^\infty e^{-x}\left(\sum_{t=0}^{N}\binom{N}{t}\left(\frac{x}{N}\right)^t\right) = \\ &= \sum_{t=0}^{N}\binom{N}{t}\frac{1}{N^t}\int_0^\infty e^{-x}x^t dx = \\ &= \sum_{t=0}^{N}\binom{N}{t}\frac{1}{N^t}\Gamma(t+1) = \\ &= \sum_{t=0}^{N}\binom{N}{t}\frac{t!}{N^t} = \mathsf{E}\nu. \end{aligned}$$

3. Разобъем искомый интеграл на два: $I_1 = N\int_0^1$ и $I_2 = N\int_1^\infty$. Для второго интеграла

справедлива оценка (с учетом неравенства $\ln(1+y/2) < y/2$):

$$\begin{aligned} I_2 &= N\int_1^\infty e^{-Nx}(1+x)^N dx = \\ &= N(2/e)^N \int_0^\infty e^{-Ny}(1+y/2)^N dy < \\ &< N(2/e)^N \int_0^\infty e^{-Ny} e^{Ny/2} dy = N(2/e)^N \cdot 2/N = 2(2/e)^N = o(1). \end{aligned}$$

Для оценки первого интеграла применим метод Лапласа с функцией

$$S(x) = -x + \ln(1+x).$$

Условия леммы выполнены, и мы получаем нужный результат. □

Название атаки объясняется известным *парадоксом дней рождения*: при случайном равновероятном «размещении» дней рождения 23 учеников класса по 365 дням года у некоторых двух учеников дни рождения совпадут с вероятностью более $1/2$. Парадоксом является то, что совпадения начинают происходить при достаточно небольшом числе учеников. Теорема говорит о классах из $O(\sqrt{N})$ учеников. Отметим, что оценки типа «корень квадратный из N» часто имеют место в криптографии.

4.6. Модернизированная атака «дней рождения»

Атака «дней рождения» обладает двумя серьезными недостатками. Во-первых, она имеет низкую криптографическую мотивацию. Действительно, Виктора интересуют коллизионные пары (X, X'), в которых слово X семантически близко к истинному сообщению, а слово X' — к поддельному (см. таблицу 4.1), в то время как атака позволяет найти среди элементов $\mathcal{X}$ случайную коллизионную пару. Во-вторых, для атаки требуются большие ресурсы памяти (N ячеек).

Известны модернизации атаки «дней рождения», лишенные указанных недостатков. Приведем описание одной из них.

Подготовительный этап. При подготовке к атаке выполняются следующие построения:

1. Строится разбиение $\mathcal{B}_1 \cup \mathcal{B}_2$ множества $\{0,1\}^n$. Части разбиения равновелики: $|\mathcal{B}_1| = |\mathcal{B}_2|$.
2. Строится функция модификаций $\delta\colon \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^*$. Значение $\delta(Z,Y)$ есть результат модификации слова Z, определяемой словом Y, причем $\delta(Z,Y) \neq \delta(Z,Y')$, если $Y \neq Y'$. Модификации не изменяют сути Z и могут, например, состоять в добавлении или удалении незначащих символов.
3. Строится функция $\varphi\colon \{0,1\}^n \to \{0,1\}^n$,

$$\varphi(Y) = \begin{cases} h(\delta(X,Y)), & \text{если } Y \in \mathcal{B}_1, \\ h(\delta(X',Y)), & \text{если } Y \in \mathcal{B}_2. \end{cases}$$

Выполнив построения, можно выбрать $Y_0 \in \{0,1\}^n$ и определить последовательность $Y_t = \varphi(Y_{t-1}) = \varphi^t(Y_0)$, $t = 1, 2, \ldots$. Данная последовательность (*траектория*) является периодической, пусть r — ее минимальный период и t_0 — предпериод (рис. 4.2).

Если $t_0 \neq 0$, то (Y_{t_0-1}, Y_{t_0+r-1}) — коллизионная пара для φ:

$$\varphi(Y_{t_0-1}) = \varphi(Y_{t_0+r-1}).$$

В частности, если Y_{t_0-1} и Y_{t_0+r-1} лежат в разных множествах $\mathcal{B}_1$ и $\mathcal{B}_2$, например, $Y_{t_0-1} \in \mathcal{B}_1$ и $Y_{t_0+r-1} \in \mathcal{B}_2$, то модификациям $\delta(X, Y_{t_0-1})$ и $\delta(X', Y_{t_0+r-1})$ истинного и поддельного сообщений соответствует одинаковое хэш-значение Y_{t_0}.

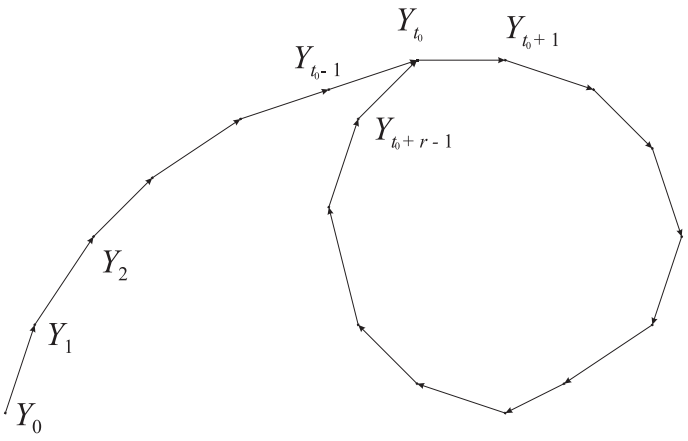


Рис. 4.2. Периодическая траектория

Известно, что если φ выбрано случайно равновероятно из множества всех преобразований на $\{0,1\}^n$, то

$$\mathsf{E}t_0 = \frac{1}{2}\sqrt{\frac{\pi N}{2}} + O(1), \quad \mathsf{E}r = \frac{1}{2}\sqrt{\frac{\pi N}{2}} + O(1),$$

т. е. для обнаружения коллизии снова потребуется выполнить порядка $\sqrt{N}$ хэшированій в среднем.

Оперативный этап. Оперативный этап проводится следующим образом (реализацию шага 2 рассмотрим немного позже):

1. $Y_0 \xleftarrow{R} \{0,1\}^n$.
2. Определить минимальный период r последовательности $Y_t = \varphi^t(Y_0)$, $t = 1, 2, \ldots$.
3. $A \leftarrow Y_0$, $B \leftarrow \varphi^r(Y_0)$.
4. Если $A = B$, то возвратить $\perp$ ($t_0 = 0$).
5. Пока $\varphi(A) \neq \varphi(B)$: $A \leftarrow \varphi(A)$, $B \leftarrow \varphi(B)$.
6. Если $A, B \in \mathcal{B}_1$ или $A, B \in \mathcal{B}_2$, то возвратить $\perp$ (коллизия для φ найдена, но оказалась бесполезной).
7. Если $A \in \mathcal{B}_1$ и $B \in \mathcal{B}_2$, то возвратить $(\delta(X, A), \delta(X', B))$.
8. Если $A \in \mathcal{B}_2$ и $B \in \mathcal{B}_1$, то возвратить $(\delta(X, B), \delta(X', A))$.

При вероятностной идеализации φ (φ выбирается наудачу из множества всех преобразований $\{0,1\}^n$) и достаточно большом n вероятность события $\{t_0 = 0\}$ незначительна, а переменные A и B попадут в различные множества $\mathcal{B}_1$ и $\mathcal{B}_2$ с вероятностью, близкой к $1/2$. Таким образом, атака завершится успехом примерно в половине случаев. При необходимости атаку можно повторить, повышая вероятность успеха.

Остается определиться с реализацией шага 2. Для определения минимального периода последовательности (Y_t) можно использовать следующий алгоритм, который работает на совсем небольшой памяти.

Алгоритм Брента

Вход: $Y_0 \in \{0,1\}^n$, φ — преобразование $\{0,1\}^n$ (задается алгоритмически).

Выход: r — минимальный период последовательности Y_0, $Y_1 = \varphi(Y_0)$, $Y_2 = \varphi(Y_1)$,....

Шаги:

1. $A \leftarrow Y_0$.
2. Для $i = 0, 1, \ldots$ выполнить:
   (1) $B \leftarrow \varphi(A)$;

(2) для $r = 1, 2, \ldots, 2^i$:
если $A = B$, то возвратить r, иначе $B \leftarrow \varphi(B)$;
(3) $A \leftarrow B$.

На i-й итерации алгоритма проверяется совпадение сохраненного значения Y_t, $t = 2^i - 1$, с вычисляемыми значениями Y_τ, $\tau = 2^i, 2^i + 1, \ldots, 2^{i+1} - 1$:

Y_0	Y_1	Y_3	$\ldots$	Y_{2^i-1}	$\ldots$
Y_1	Y_2, Y_3	Y_4, Y_5, Y_6, Y_7	$\ldots$	$Y_{2^i}, \ldots, Y_{2^{i+1}-1}$	$\ldots$

Первое найденное совпадение $Y_t = Y_\tau$ означает, что минимальный период $r = t - \tau$. Действительно, если минимальный период $r' < r$, то $Y_t = Y_{t+r'}$ и совпадение $Y_t = Y_\tau$ не является первым.

4.7. Задания

Задание 4.1. Пусть p — простое, g — примитивный элемент $\mathbb{F}_p^*$, $h\colon \{0,1\}^* \to \mathbb{F}_p^*$, $x \mapsto g^x \bmod p$ (слово x отождествляется с числом). Сформулировать для h криптоаналитические задачи. Какие задачи допускают простое решение? (Считать, что задача дискретного логарифмирования в $\mathbb{F}_p^*$ является труднорешаемой).

Задание 4.2. Пусть $n = pq$ — модуль RSA, $h\colon \{0,1\}^* \to \mathbb{Z}_n$, $x \mapsto x^2 \bmod n$ — функция хэширования (слово x отождествляется с числом). Сформулировать для h криптоаналитические задачи. Какие задачи допускают простое решение? (Считать, что задача факторизации n является труднорешаемой).

Задание 4.3. Пусть функция $h_1\colon \{0,1\}^{2n} \to \{0,1\}^n$ является строго свободной от коллизий. Доказать, что строго свободной от коллизий будет также всякая функция

$$h_i\colon \{0,1\}^{2^i n} \to \{0,1\}^n, \quad X_1 \parallel X_2 \mapsto h_1(h_{i-1}(X_1) \parallel h_{i-1}(X_2)).$$

Здесь $X_1, X_2 \in \{0,1\}^{2^{i-1}n}$, $i = 2, 3, \ldots$.

Задание 4.4. Количество образов блочно-итерационной хэш-функции h не больше количества образов ее шаговой функции хэширования σ. Доказать, что имеется

$$\sum_{i=0}^{N} (-1)^i \binom{N}{i} (N-i)^R$$

различных сюръективных отображений $\{0,1\}^{n_b+n} \to \{0,1\}^n$. Здесь $N = 2^n$, $R = 2^{n_b+n}$.

Задание 4.5. Выбрать произвольную шаговую функцию хэширования из таблицы 4.2 и показать, что атаки на нее сводятся к атакам на криптосистему F.

Задание 4.6. Предложить способ обращения и построения коллизии для шаговой функции хэширования $\sigma(X \parallel Y) = F_Y(X \oplus Y) \oplus Y$.

Задание 4.7. Предложить способ обращения и построения коллизии для шаговой функции хэширования $\sigma(X \parallel Y) = F_{X \oplus Y}(X) \oplus X \oplus Y$.

Задание 4.8. Предложить способ обращения и построения коллизии для шаговой функции хэширования $\sigma(X \parallel Y) = F_{X \oplus Y}(Y) \oplus X \oplus Y$.

Задание 4.9 (алгоритм Флойда). Алгоритм Флойда поиска совпадения элементов периодической (с минимальным периодом r и предпериодом t_0) последовательности $Y_0, Y_1, \ldots$ состоит в проверке равенств $Y_t = Y_{2t}$ для $t = 1, 2, \ldots$. Доказать, что номер первого совпадения $t = r(1 + \lfloor t_0/r \rfloor)$.

Задание 4.10. Пусть $h\colon \{0,1\}^* \to \{0,1\}^n$ — блочно-итерационная функция и ее шаговая функция хэширования σ такова, что по заданным X_t и $\sigma(X_t \parallel Y)$ можно легко найти Y. Разработать атаку на h, направленную на решение задачи **H2**. Время атаки должно иметь порядок $2^{n/2}$. Считать, что слово X, которое входит в постановку задачи, состоит из не менее чем двух блоков.

Задание 4.11. Пусть в системе имитозащиты I (см. 3.15) используется блочно-итерационная хэш-функция $h\colon \{0,1\}^* \to \{0,1\}^n$, $\Theta = \{0,1\}^n$, ключом является начальное хэш-значение Y_0. Разработать атаку на I, направленную на решение задачи **M3**. В ходе атаки разрешается выбирать сообщения, подлежащие имитозащите.

4.8. Комментарии

Функции хэширования описываются в книгах [2, 37, 41, 43]. Самые современные методы построения и оценки стойкости функций хэширования представлены в материалах конкурса SHA-3, проведенного в США в 2007 — 2012 гг. (см. `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`).

Среди алгоритмов построения ключа по паролю наибольшее распространение получил алгоритм `PBKDF2`, определенный в PKCS#5. Этот алгоритм включен в СТБ 34.101.45-2013 «Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых».

В некоторых алгоритмах построения ключа по паролю используются большие массивы вспомогательных данных. Делается это для того, чтобы алгоритмы трудно было перенести на спецустройства с высоким быстродействием, но малой оперативной памятью.

Кроме задач, перечисленных в табл. 4.1, существуют и другие. Например, задача построения r-коллизии

H4: найти r различных сообщений $X_1, \dots, X_r$ таких, что $h(X_1) = \dots = h(X_r)$.

Если h — идеальная функция хэширования, то для построения r-коллизии требуется порядка $N^{(r-1)/r}$ операций ($N = 2^n$). А. Жу в [31] показал, что если h является блочно-итерационной, то атаку по решению **H4** можно провести за существенно меньшее время: порядка $\log r \cdot \sqrt{N}$. Задача **H4** является достаточно искусственной. Тем не менее атака Жу показывает, что итерационная структура функций хэширования является потенциальным источником слабостей и должна каким-то образом корректироваться.

В 2005 г. группа китайских исследователей под руководством С. Ванг разработала метод построения коллизий для функции хэширования `MD5`. Метод является чрезвычайно эффективным и позволяет строить коллизии почти в реальном времени. Коллизии `MD5` были использованы в компьютерном вирусе Flame (2012 год) для обхода механизмов защиты операционной системы Windows. Предложенный группой Ванг метод, как оказалось, можно применить к схожим хэш-функциям, например, к распространенной функции `SHA-1`. На сегодняшний день самая эффективная атака по построению коллизии для `SHA-1` требует около 2^{60} хэшированій. По некоторым оценкам, такой объем вычислений станет подъемным для обычного университетского вычислительного кластера в 2021 году.

Сумма для $\mathsf{E}\nu$, которая фигурирует в формулировке теоремы 4.3, без первого члена $N^{[0]}/N^0$ введена Д. Кнутом в [3] и названа им Q-функцией Раманужана. Более точная асимптотика для Q-функции:

$$Q(N) \sim \sqrt{\frac{\pi N}{2}} - \frac{1}{3} + \frac{1}{12}\sqrt{\frac{\pi}{2N}} - \frac{4}{135N} + \dots.$$

Пусть поиск коллизии ведется для блочно-итерационной хэш-функции с $n_b = n$. В работе [13] показано, что если искать коллизию на сообщениях одинаковой длины, то при

идеализации шаговой функции хэширования время ожидания коллизии уменьшается до

$$\mathsf{E}\nu = \frac{\sqrt{\pi N}}{2} + \frac{5}{6} + o(1),$$

т. е. примерно в $\sqrt{2}$ раза. В рассуждениях используется двойная Q-функция:

$$Q(M, N) = \sum_{k=0}^{\min(M,N)} \frac{M^{[k]} N^{[k]}}{M^k N^k}.$$

Глава 5
ПРОТОКОЛЫ ФОРМИРОВАНИЯ ОБЩЕГО КЛЮЧА

5.1. Головоломки Меркля

Пусть Алиса и Боб — абоненты некоторой информационной системы. Данные в системе передаются по открытым каналам связи, которые могут прослушиваться противником Виктором. Для защиты от Виктора Алиса и Боб используют блочную криптосистему и выполняют шифрование сообщений на секретных ключах, которые им поставляет Трент. Если в системе имеется n абонентов, то для организации взаимодействия всех возможных пар потребуется $n(n-1)/2$ ключей. Число ключей быстро растет с ростом n. Кроме этого, ключи следует распространять по секретным каналам связи, организация и поддержка которых может быть очень трудоемкой. Возникает вопрос: можно ли обойтись без Трента и без секретных каналов?

Положительный ответ на этот вопрос дал Р. Меркль в 1972 г. Меркль предложил протокол, с помощью которого стороны могут согласовать общий секретный ключ по *аутентифицируемому каналу связи* (АКС), занимающему промежуточное положение между открытым и секретным. Виктор может перехватить сообщение, передаваемое по АКС, но не может изменить его так, чтобы это не было обнаружено Алисой или Бобом. Виктор не может также передавать по АКС свои сообщения от чужого имени. Другими словами, АКС обеспечивает контроль целостности и подлинности данных, но не обязательно обеспечивает их конфиденциальность.

Протокол Меркля

Предназначен для формирования сторонами общего секретного ключа K_i

---

Стороны: Алиса, Боб.

Каналы: АКС.

Параметры: N и M.

Шаги:

1. Алиса составляет и отправляет Бобу список из N головоломок. Любой абонент (и Боб, и Виктор) может решить головоломку за время $O(M)$. Решением i-й головоломки является ключ K_i, выбранный Алисой.
   В качестве головоломки можно использовать результат зашифрования пары $(\texttt{‘головоломка’}, K_i)$ на ключе $\theta_i \in \Theta$, $|\Theta| = M$. Решение состоит в проведении атаки «грубой силой» по определению θ_i при известном открытом тексте `‘головоломка’`.
2. Боб выбирает головоломку со случайным номером i, решает ее, определяет ключ K_i и отправляет Алисе сообщение `‘Привет’`, зашифрованное на K_i.
3. Алиса просматривает $K_1, \ldots, K_N$ и находит среди них ключ K_i, на котором было зашифровано сообщение `‘Привет’`.

Обсудим надежность протокола. Боб решает всего одну головоломку и тратит на это время $O(M)$. Алисе требуется проверить не более N ключей, что можно сделать за время $O(N)$. А вот Виктору для проверки тех же ключей потребуется время $O(NM)$, поскольку для определения каждого ключа K_i ему надо решить новую головоломку.

Сторона	Время
Алиса	$O(N)$
Боб	$O(M)$
Виктор	$O(NM)$

Управляя N и M, можно добиться того, что время $O(NM)$ будет неприемлемо большим, хотя вычисления за время $O(N)$ и $O(M)$ будут устраивать Алису и Боба.

5.2. Протокол Диффи — Хеллмана

Головоломки Меркля не нашли практического применения. Они так и остались концептом, демонстрирующим возможность организации приципиально новой системы криптографической связи. Тем не менее протокол Меркля натолкнул У. Диффи и М. Хеллмана на разработку похожего протокола, который впоследствии был назван в их честь. *Протокол Диффи — Хеллмана* был описан в 1976 году в знаменитой работе «Новые направления в криптографии».

В протоколе нам потребуется циклическая группа G порядка q. Будем записывать эту группу аддитивно, подразумевая соглашения для групп точек эллиптических кривых. Пусть $\mathrm{descr}(G)$ — описание группы G, P — ее образующий, O — ее единица: $G = \{O, P, 2P, \ldots, (q-1)P\}$.

Протокол Диффи — Хеллмана

Предназначен для формирования общего секретного ключа K

Стороны: A (Алиса), B (Боб).

Каналы: АКС.

Параметры: $\mathrm{descr}(G)$, P.

Шаги:

1. A: $d_A \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $Q_A \leftarrow d_A P$.
2. B: $d_B \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $Q_B \leftarrow d_B P$.
3. $A \xrightarrow{\text{АКС}} B$: Q_A.
4. $A \xleftarrow{\text{АКС}} B$: Q_B.
5. A: $K \leftarrow d_A Q_B$.
6. B: $K \leftarrow d_B Q_A$.

Корректность. Формируемые сторонами ключи совпадают: $d_A Q_B = d_A d_B P = d_B Q_A$.

Далее в главе мы будем последовательно уточнять данный протокол. Попутно будем рассматривать основные атаки на протоколы формирования общего ключа и соответствующие требования к этим протоколам.

Числа d_A, d_B называются *личными* ключами. Как и секретные ключи блочных или поточных криптосистем, личные ключи выбираются случайным образом, хранятся в секрете. Элементы Q_A, Q_B группы G называются *открытыми* ключами. Личный ключ d_A однозначно определяет открытый ключ Q_A и, наоборот, Q_A однозначно определяет d_A.

Открытые ключи обращаются в информационной системе в открытом виде и доступны Виктору. Однако при надлежащем выборе G нахождение кратного $(\mathrm{descr}(G), P, d_A) \mapsto d_A P$ является простой вычислительной задачей, которую Алиса решит за приемлемое время, а дискретное логарифмирование $\mathtt{DL}$: $(\mathrm{descr}(G), P, Q_A) \mapsto d_A$ — вычислительно трудной, с которой Виктор не справится. Оказывается, что для определенных семейств групп G растущего порядка q времена работы сторон имеют следующий вид:

Сторона	Время
Алиса	$(\log q)^{O(1)}$
Боб	$(\log q)^{O(1)}$
Виктор	$O(q^{1/2})$ (экспоненциальное) или $2^{(c+o(1))(\log q)^{\alpha}(\log\log q)^{1-\alpha}}$, $0<\alpha<1$ (субэкспоненциальное)

Для атаки на протокол Виктору не обязательно находить личные ключи сторон. Достаточно по $(\mathrm{descr}(G), P, d_AP, d_BP)$ определить ключ $K = d_Ad_BP$. Эта задача, известная как вычислительная задача Диффи — Хеллмана (CDH, см. § 2.2), также признается вычислительно трудной для определенных семейств групп G.

Имеется полинимиальная сводимость задачи CDH к задаче DL в той же группе. Действительно, используя алгоритм решения DL, можно найти d_A по $(\mathrm{descr}(G), P, d_AP)$, а затем определить $K = d_A(d_BP)$. Обратное сведéние DL к CDH на сегодняшний день не доказано. Доказательство этого сведéния является одной из известных нерешенных проблем теоретической криптографии.

Протокол Диффи — Хеллмана может быть преобразован в протокол выработки общего ключа тремя абонентами. Пусть C (Клара) — третий абонент. Клара генерирует личный ключ $d_C \xleftarrow{R} \{1, 2, \ldots, q-1\}$ и определяет открытый ключ $Q_C \leftarrow d_CP$. Стороны обмениваются между собой открытыми ключами, а затем выполняют следующие пересылки:

$$A \overset{\text{АКС}}{\to} B\colon d_AQ_C;$$
$$B \overset{\text{АКС}}{\to} C\colon d_BQ_A;$$
$$C \overset{\text{АКС}}{\to} A\colon d_CQ_B.$$

По окончании пересылок каждая из сторон находит один и тот же общий ключ:

$$A\colon K \leftarrow d_A(d_CQ_B);$$
$$B\colon K \leftarrow d_B(d_AQ_C);$$
$$C\colon K \leftarrow d_C(d_BQ_A).$$

5.3. Атака «противник посередине»

Аутентифицируемый канал связи занимает промежуточное положение между секретным и открытым. Реализовать АКС проще, чем СКС. Представим, что Боб уехал в далекую страну, все каналы связи с ним проходят по специальному кабелю, проложенному по дну океана, и потенциально прослушиваются Виктором. Алиса вполне резонно сомневается в конфиденциальности пересылаемых Бобу данных. Алиса могла бы зашифровать данные, но для этого нужен ключ, который снова требуется передать конфиденциально. С организацией СКС ничего не получается. С другой стороны, Алиса и Боб вполне могут организовать АКС. Например, Алиса и Боб могут обменяться данными по электронной почте (ОКС), а затем проверить целостность и подлинность данных по телефону (АКС). Контроль подлинности данных поддерживается голосовой (характерный тембр голоса) и вербальной (характерные жаргонизмы) аутентификацией абонентов, целостность — проверкой контрольных характеристик переданных данных. Отметим, что телефонный АКС рекомендован в системе защищенной электронной почты PGP при обмене открытыми ключами.

При выполнении протокола Диффи — Хеллмана стороны вырабатывают общий секрет K, с помощью которого могут организовать СКС. Таким образом, протокол Диффи — Хеллмана фактически позволяет преобразовать АКС в СКС.

Можно ли отказаться от АКС и передавать открытые ключи по ОКС? Оказывается, что нет. Если CDH — вычислительно трудная задача, то протокол Диффи — Хеллмана является стойким относительно атак пассивного противника Виктора. Однако ситуация кардинальным образом меняется, если у Виктора есть возможность не только перехватывать, но и менять сообщения, пересылаемые по каналу связи.

Виктор может провести следующую *атаку «противник посередине»*:

1. Виктор выбирает случайные d'_A, d'_B и меняет в канале связи Q_A на $Q'_A = d'_A P$, Q_B на $Q'_B = d'_B P$.
2. По окончании протокола Алиса сформирует ключ $K_1 = d_A Q'_B$, а Боб — ключ $K_2 = d_B Q'_A$.
3. Оба ключа известны Виктору: $K_1 = d'_B Q_A$, $K_2 = d'_A Q_B$.

Пусть по завершении протокола Алиса использует K_1 для шифрования сообщений, передаваемых Бобу по ОКС. Виктор перехватывает зашифрованные сообщения, расшифровывает их на K_1 и зашифровывает на K_2. Виктор прочитывает все сообщения Алисы, но ни Алиса, ни Боб от этом даже не догадываются.

Атаку «противник посередине» в англоязычной литературе принято обозначать MITM (от Man-In-The-Middle). Атака является универсальной в том смысле, что может применяться к широкому классу протоколов, стороны которых не связаны АКС. Как показывает следующий пример, протоколы могут быть даже не криптографическими.

Пример 5.1 (CAPTCHA). С помощью тестов CAPTCHA (от англ. Completely Automated Public Turing test to tell Computers and Humans Apart) машина (компьютер) проверяет, что взаимодействует с человеком. Машина преобразует текст в графическую картинку, искажая начертания символов, и предлагает восстановить по картинке текст. Человек это делает легко, а вот машина (другой компьютер) — нет. Для автоматического прохождения тестов с сайта A Виктор разработал сайт B, на котором дублирует тесты. Тестирование проходят люди, их ответы пересылаются на сайт A и засчитываются как правильные. Проделанные Виктором манипуляции по сути являются атакой «противник посередине». □

5.4. Сертификаты открытых ключей

Для защиты от атаки «противник посередине» Алиса и Боб должны распределять свои открытые ключи так, чтобы можно было контролировать их подлинность и целостность. Первоначально вопросам распространения открытых ключей не уделялось большого внимания. Считалось, что открытые ключи будут размещаться в некотором общедоступном справочнике Трента и абоненты будут получать доступ к этому справочнику по АКС.

Со временем стало понятно, что управление открытыми ключами является, по словам В. Столлингса, «ахиллесовой пятой криптографии с открытыми ключами». Возникли вопросы: как защищать справочник? как организовать доступ к нему? как организовать защиту каналов доступа? Вместо глобального общедоступного справочника были разработаны более гибкие решения. Самое распространенное из них основано на *сертификатах открытых ключей* и регламентируется международным стандартом X.509, введенным в 1988 году. Фактически X.509 предлагает способ организации *распределенного* справочника. Стандарт X.509 принят в нашей стране в виде СТБ 34.101.19.

Пусть Трент также вырабатывает личный ключ d_T и определяет по нему открытый ключ Q_T. Трент использует пару (d_T, Q_T) не в протоколах формирования общего ключа, а в алгоритмах электронной цифровой подписи. Будем для простоты считать, что в алгоритмах ЭЦП используются параметры $\mathrm{descr}(G)$ и P, и что ключи d_T и Q_T устроены также, как и ключи протокола Диффи – Хеллмана (например, алгоритмы ЭЦП построены по схеме Шнорра). С помощью d_T Трент вырабатывает ЭЦП $S_T(X)$

сообщения $X \in \{0,1\}^*$, а с помощью Q_T любой другой абонент проверяет соответствие подписи сообщению. Положительный результат проверки означает, что ЭЦП создана Трентом (контроль подлинности) и с момента создания подписи сообщение X не изменялось (контроль целостности).

Сертификат открытого ключа абонента — это данные об абоненте, его открытом ключе, атрибутах открытого ключа, заверенные ЭЦП Трента. В стандарте X.509 Трента называют *удостоверяющим центром*. Сертификат Алисы обозначается $\mathrm{Cert}_T(Id_A, Q_A)$ и состоит из следующих полей:

$\mathrm{Cert}_T(Id_A, Q_A) =$

Версия сертификата
Номер сертификата
Id_T (полное имя Трента)
Срок действия сертификата
Id_A (полное имя Алисы)
$\mathrm{descr}(G)$, P
Q_A
Расширения сертификата (дополнительные атрибуты)
S_T(объединение предыдущих полей)

Фактически, сертификат связывает имя Алисы Id_A и ее открытый ключ Q_A.

Боб, располагая сертификатом $Cert_T(Id_A, Q_A)$ и открытым ключом Q_T, может проверить подлинность и целостность Q_A. Открытый ключ Трента также может распространяться в виде сертификата $\mathrm{Cert}_{T_1}(Id_T, Q_T)$, заверенного еще одним удостоверяющим центром T_1. Для проверки открытого ключа T_1 может использоваться еще один сертификат $\mathrm{Cert}_{T_2}(Id_{T_1}, Q_{T_1})$ и так далее. В конце концов образуется *цепочка сертификатов*

$$\mathrm{Cert}_{T_1}(Id_T, Q_T), \mathrm{Cert}_{T_2}(Id_{T_1}, Q_{T_1}), \ldots, \mathrm{Cert}_{T_{n-1}}(Id_{T_n}, Q_{T_n}), \mathrm{Cert}_{T_n}(Id_{T_n}, Q_{T_n}).$$

Последнее звено цепочки — это самоподписанный сертификат *корневого* удостоверяющего центра T_n. Такие сертификаты, как правило, выпускаются известными компаниями, оказывающими услуги в области криптографии с открытым ключом. Сертификаты корневых удостоверяющих центров включаются в дистрибутивы операционных систем, «прошиваются» в браузерах и т. д.

5.5. Протокол с сертификатами

Модернизируем протокол Диффи — Хеллмана, использовав в нем сертификаты. Здесь и далее при описании протоколов будем опускать параметры $\mathrm{descr}(G)$ и P.

Протокол Диффи – Хеллмана с сертификатами

Стороны: A (Алиса), B (Боб), T (Трент).

Каналы: ОКС, АКС.

Генерация и распределение ключей Трента:

1. T: $d_T \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $Q_T \leftarrow d_T P$.
2. $T \xrightarrow{\text{АКС}} A$: Q_T.
3. $T \xrightarrow{\text{АКС}} B$: Q_T.

Выдача сертификата Алисе (*Бобу – аналогично*):

1. A: $d_A \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $Q_A \leftarrow d_A P$.
2. $A \xrightarrow{\text{АКС}} T$: Q_A.
3. T: формирует $\mathrm{Cert}_T(Id_A, Q_A)$, используя d_T.

4. $A \overset{\text{АКС}}{\leftarrow} T$: $\text{Cert}_T(Id_A, Q_A)$.

Формирование общего ключа K:

1. $A \overset{\text{ОКС}}{\rightarrow} B$: $\text{Cert}_T(Id_A, Q_A)$.
2. $A \overset{\text{ОКС}}{\leftarrow} B$: $\text{Cert}_T(Id_B, Q_B)$.
3. A: проверяет $\text{Cert}_T(Id_B, Q_B)$ на Q_T, определяет $K \leftarrow d_A Q_B$.
4. B: проверяет $\text{Cert}_T(Id_A, Q_A)$ на Q_T, определяет $K \leftarrow d_B Q_A$.

Здесь и далее при описании протоколов считаем, что при неверной проверке на любом шаге стороны немедленно прерывают выполнение протокола с ошибочным результатом.

В новом протоколе открытые ключи Алисы передаются в виде сертификатов по ОКС. Пересылку сертификатов предваряет распределение открытого ключа Трента и выдача им сертификатов. Подготовительные мероприятия фактически преобразуют канал передачи сертификатов в АКС.

Трент может распространять Q_T в виде сертификата или в виде цепочки сертификатов, корневой удостоверяющий центр которой признается и Алисой, и Бобом. Передача Q_A, Q_B реализуется тем, что сторона A или B обращается в удостоверяющий центр T, там проверяется подлинность стороны (например, паспортная аутентификация), а затем принимается ее открытый ключ.

Далее для простоты будем опускать предварительные этапы распределения ключей Трента и выдачи сертификатов. По умолчанию, пересылки в последующих протоколах выполняются по открытым каналам связи.

5.6. Протоколы MTI

Ключ K, который формируется в результате выполнения последнего протокола, впоследствии используется для шифрования, имитозащиты и для решения других криптографических задач в сеансе связи между Алисой и Бобом. Поэтому ключ K называется *сеансовым*.

Существует угроза того, что ключ K будет использоваться неправильно и станет известен Виктору. Например, Алиса может просто забыть удалить K по завершении сеанса. Важно, чтобы компрометация ключа одного сеанса не сказалась на стойкости ключей других сеансов. Соответствующее требование к протоколам называется *защитой при компрометации сеансовых ключей* (KKS, Known Key Security).

В последнем протоколе ключи d_A, Q_A, d_B, Q_B целесообразно сделать долговременными или, как еще говорят, *статическими*. Действительно, процедура выдачи сертификата открытого ключа являются достаточно трудоемкой, и поэтому использование сертификата только в одном сеансе — весьма неэффективное решение, лишающее применение открытых ключей практического смысла. С другой стороны, при многократном использовании одних и тех же личных и открытых ключей протокол не будет удовлетворять требованию KKS, поскольку вырабатываемые сторонами ключи K будут все время одинаковыми.

Т. Мацумото, Ю. Такашима и Х. Имаи разработали в 1986 г. протоколы, которые впоследствии получил название MTI по первым буквам фамилий авторов. В протоколах MTI кроме статических используются еще одноразовые или, как их еще называют, *эфемерные* ключи. Эфемерные ключи выбираются сторонами случайным образом. Их применение делает сеансовые ключи K также случайными даже при фиксированных статических ключах, что обеспечивает выполнение свойства KKS.

Протокол MTI/A0

Шаги:

1. A: $u_A \overset{R}{\leftarrow} \{1, 2, \ldots, q-1\}$, $V_A \leftarrow u_A P$.
2. $A \to B$: $\mathrm{Cert}_T(Id_A, Q_A)$, V_A.
3. B: $u_B \overset{R}{\leftarrow} \{1, 2, \ldots, q-1\}$, $V_B \leftarrow u_B P$.
4. $A \leftarrow B$: $\mathrm{Cert}_T(Id_B, Q_B)$, V_B.
5. A: проверяет $\mathrm{Cert}_T(Id_B, Q_B)$, $K \leftarrow d_A V_B + u_A Q_B$.
6. B: проверяет $\mathrm{Cert}_T(Id_A, Q_A)$, $K \leftarrow d_B V_A + u_B Q_A$.

Корректность: $d_A V_B + u_A Q_B = (d_A u_B + d_B u_A)P = d_B V_A + u_B Q_A$.

Протокол MTI/C0

Ограничения: q — простое.

Шаги:

1. $A \to B$: $\mathrm{Cert}_T(Id_A, Q_A)$.
2. $A \leftarrow B$: $\mathrm{Cert}_T(Id_B, Q_B)$.
3. A: проверяет $\mathrm{Cert}_T(Id_B, Q_B)$, $u_A \overset{R}{\leftarrow} \{1, 2, \ldots, q-1\}$, $V_A \leftarrow u_A Q_B$.
4. B: проверяет $\mathrm{Cert}_T(Id_A, Q_A)$, $u_B \overset{R}{\leftarrow} \{1, 2, \ldots, q-1\}$, $V_B \leftarrow u_B Q_A$.
5. $A \to B$: V_A.
6. $A \leftarrow B$: V_B.
7. A: $K \leftarrow (d_A^{-1} \bmod q) u_A V_B$.
8. B: $K \leftarrow (d_B^{-1} \bmod q) u_B V_A$.

Корректность: $d_A^{-1} u_A V_B = u_A u_B P = d_B^{-1} u_B V_A$.

В этих протоколах u_A, u_B — эфемерные личные ключи, V_A, V_B — эфемерные открытые ключи.

Протоколы MTI имеют слабости, которые мы сейчас рассмотрим, параллельно обсудив дополнительные требования к протоколам формирования общего ключа.

Защита от «чтения назад». Долговременные личные ключи d_A, d_B могут быть скомпрометированы, например, если потеряны ключевые носители, на которых они хранятся. Даже при раскрытии долговременных ключей противник не должен получать возможность определять предыдущие сеансовые ключи. Соответствующее требование к протоколам называется *защита от «чтения назад»* (PFS, Perfect Forward Secrecy).

Рассмотрим выполнение свойства PFS для протоколов MTI.

Пусть Виктор получил оба ключа — d_A, d_B — протокола MTI/C0. Тогда он может вычислить $u_A P = (d_B^{-1} \bmod q) V_A$ и $u_B P = (d_A^{-1} \bmod q) V_B$. Для определения K Виктору надо найти $u_A u_B P$ по тройке $(P, u_A P, u_B P)$. Но это трудная задача `CDH` и, таким образом, протокол MTI/C0 обладает свойством PFS.

Пусть Виктор получил один из ключей, например d_A, протокола MTI/C0. Определение $K = d_A V_B + u_A Q_B$ для Виктора равносильно определению $u_A Q_B = u_A d_B P$ по известным $u_A P = V_A$ и $d_B P = Q_B$. Но это снова трудная задача `CDH`. Ситуация меняется, если Виктор получает два ключа. В этом случае он может найти K по данным перехвата:

$$K = d_A V_B + d_B V_A.$$

Таким образом, MTI/A0 обеспечивает защиту от «чтения назад» при компрометации одного долговременного личного ключа и не обеспечивает при компрометации двух ключей.

Атака «малая подгруппа». В протоколах MTI стороны обмениваются элементами V_A, V_B группы G. Эти элементы представляются двоичными словами. Стороны не проверяют формат присылаемых данных. Виктор может воспользоваться этим, навязывая сторонам использование вместо G другой группы H, которая имеет малый порядок и,

следовательно, элементы которой могут быть легко угаданы.

Пусть в протоколе MTI/C0 группа G представляет собой группу точек эллиптической кривой над полем $\mathbb{F}_p$: $G = E(\mathbb{F}_p)$. Эллиптическая кривая задается уравнением $E\colon y^2 = x^3+ax+b$, $a, b \in \mathbb{F}_p$. Коэффициент b определяет строение группы, базовую точку P, но не используется непосредственно в формулах сложения точек. Виктор учитывает данный факт и проводит следующую *атаку «малая подгруппа»*.

1. Виктор находит коэффициент $b' \in \mathbb{F}_p$, который определяет новое уравнение $E'\colon y^2 = x^3 + ax + b'$ и новую группу $H = E'(\mathbb{F}_p)$. Виктор подбирает b' так, чтобы группа H содержала точку W малого порядка m. Важно, что вычисления с элементами $E'(\mathbb{F}_p)$ ведутся по тем же формулам, что и вычисления с элементами $E(\mathbb{F}_p)$.
2. Виктор меняет открытые ключи V_A, V_B на произвольные точки $W_A, W_B \in \langle W\rangle$.
3. По окончании протокола Алиса и Боб определят случайные ключи $K_A, K_B \in \langle W\rangle$, причем $K_A = K_B$ с вероятностью $1/m$. Если m невелико, то ключи совпадут и стороны не обнаружат подмены, а противник сможет перебрать все элементы $\langle W\rangle$ и определить совпавшие ключи.

Как правило, существует простой способ защиты от атаки «малая подгруппа». Сторонам протокола следует всего лишь дополнительно проверять, что присланные сообщения действительно представляют собой элементы G.

5.7. Аутентификация

После завершения протокола Диффи — Хеллмана стороны располагают ключом K. В некоторых случаях Алиса и Боб должны убедиться, что ключ действительно общий. Для этого стороны могут выполнить дополнительные шаги, которые называются *подтверждением ключа*. Подтверждение может быть организовано с помощью системы имитозащиты (см. § 3.15), как это сделано в следующем протоколе. Будем использовать систему $I = \{I_K\colon K \in G\}$, функции которой ставят в соответствие сообщениям $X \in \{0,1\}^*$ имитовставки $Z = I_K(X)$.

Протокол Подтверждение ключа

Стороны A и B проверяют, что ключ K, выработанный по протоколу Диффи — Хеллмана, действительно общий

Шаги (*после шагов по формированию* K):

1. A: $Z_A \leftarrow I_K(\texttt{`1'})$.
2. B: $Z_B \leftarrow I_K(\texttt{`2'})$.
3. $A \to B$: Z_A.
4. $A \leftarrow B$: Z_B.
5. A: $Z_B \stackrel{?}{=} I_K(\texttt{`2'})$.
6. B: $Z_A \stackrel{?}{=} I_K(\texttt{`1'})$.

Стороны протокола демонстрируют друг другу, что могут определять имитовставки заданных сообщений на ключе K и, таким образом, показывают знание K. Если предшествующий протокол является надежным, то знать K могут только легитимные стороны. Поэтому протокол формирования общего ключа с дополнительным подтверждением может использоваться для аутентификации сторон. Более того, в некоторых случаях формирование общего ключа является вспомогательным этапом аутентификации.

Различные сообщения `'1'` и `'2'` защищают от *атаки «зеркалирования»*: если сообщения будут одинаковыми, то Виктор сможет выдать себя за Боба, ответив Алисе ее же имитовставкой.

Вместо имитозащиты можно использовать шифрование. Но так делать не рекомендуют, поскольку шифрование служит для обеспечения конфиденциальности сообщений, а не для их аутентификации (т. е. проверки подлинности).

Ошибочное разделение ключа. Если протокол MTI/A0 используется только для аутентификации, то Виктор может провести следующую атаку:

1. Виктор выбирает $e \in \{1, 2, \ldots, q-1\}$ и получает у Трента сертификат $\mathrm{Cert}(Id_{\tilde{A}}, Q_{\tilde{A}})$ на имя $I_{\tilde{A}}$ и на открытый ключ $Q_{\tilde{A}} = eQ_A$.
2. Во время выполнения протокола Виктор перехватывает эфемерный открытый ключ V_A и отсылает его Бобу вместе с сертификатом $\mathrm{Cert}(Id_{\tilde{A}}, Q_{\tilde{A}})$. Ответный ключ V_B Виктор меняет на eV_B и отсылает его Алисе вместе с сертификатом Боба.
3. Алиса и Боб вычислят общий секретный ключ

$$K = d_A e V_B + u_A Q_B = u_B Q_{\tilde{A}} + d_B V_A,$$

при этом Боб будет ошибочно считать, что разделяет его с Виктором (хотя Виктор даже не знает K).

Описанная атака называется «*ошибочное разделение ключа*» (UKS, Unknown Key Share). Атака является достаточно общей, известны случаи ее успешного применения к нескольким протоколам аутентификации.

Приведем пример использования атаки.

Пример 5.2. Пусть B — банк, A — его клиент. Банк и клиент формируют общий ключ K и выполняют на нем подтверждение ключа. Если подтверждение ключа завершено успешно, то банк признает клиента подлинным и переводит на его счет деньги. Виктор, выполняя описанные выше манипуляции, может вынудить банк перевести деньги на свой счет. □

Атака UKS не состоится, если при выдаче сертификата Трент предложит Виктору доказать владение личным ключом $d_{\tilde{A}}$, который соответствует открытому ключу $Q_{\tilde{A}}$ (например, Трент может предложить Виктору подписать на $d_{\tilde{A}}$ запрос на выдачу сертификата). Виктор не знает $d_{\tilde{A}} = ed_A \bmod q$ и поэтому проверку Трента не пройдет.

Еще одним способом защиты от атаки является усиленное подтверждение ключа, при котором стороны протокола формируют имитовставки Z_A, Z_B по новым правилам:

$$Z_A \leftarrow I_K(\text{`1'} \parallel Id_A \parallel Id_B \parallel V_A \parallel V_B);$$
$$Z_B \leftarrow I_K(\text{`2'} \parallel Id_B \parallel Id_A \parallel V_B \parallel V_A).$$

Имитозащита идентификаторов Id_A, Id_B указывает на пару сторон протокола. Имитозащита эфемерных ключей указывает на их сеанс.

5.8. Протокол MQV

Одним из наиболее обоснованных протоколов формирования общего ключа типа Диффи — Хеллмана является протокол MQV. Этот протокол был разработан А. Менезесом (M), М. Кью (Q) и С. Ванстоуном (V) в 1995 году, а затем несколько раз дорабатывался.

Приведем протокол в редакции, которая отличается от первоначальной. Будем считать, что порядок группы G близок к 2^{2l}. Обозначим $G^* = G \setminus \{O\}$. Будем использовать в протоколе функцию хэширования $\varphi\colon G^* \times \{0,1\}^* \to \{2^l, \ldots, 2^{l+1} - 1\}$.

Протокол MQV

Шаги:

1. A: $u_A \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $V_A \leftarrow u_A P$.

2. $A \to B$: $\mathrm{Cert}_T(Id_A, Q_A)$, V_A.
3. B: $u_B \xleftarrow{R} \{1, 2, \ldots, q-1\}$, $V_B \leftarrow u_B P$.
4. $A \leftarrow B$: $\mathrm{Cert}_T(Id_B, Q_B)$, V_B.
5. A:
   (1) проверяет $\mathrm{Cert}_T(Id_B, Q_B)$;
   (2) проверяет, что $V_B \in G^*$;
   (3) $X_A \leftarrow Id_A \parallel Id_B \parallel V_B$, $X_B \leftarrow Id_B \parallel Id_A \parallel V_A$;
   (4) $s_A \leftarrow (u_A - \varphi(V_A, X_A) d_A) \bmod q$;
   (5) $K \leftarrow s_A(V_B - \varphi(V_B, X_B) Q_B)$.
6. B:
   (1) проверяет $\mathrm{Cert}_T(Id_A, Q_A)$;
   (2) проверяет, что $V_A \in G^*$;
   (3) $X_A \leftarrow Id_A \parallel Id_B \parallel V_B$, $X_B \leftarrow Id_B \parallel Id_A \parallel V_A$;
   (4) $s_B \leftarrow (u_B - \varphi(V_B, X_B) d_B) \bmod q$;
   (5) $K \leftarrow s_B(V_A - \varphi(V_A, X_A) Q_A)$.

Корректность: $s_A(V_B - \varphi(V_B, X_B) Q_B) = s_A s_B P = s_B(V_A - \varphi(V_A, X_A) Q_A)$.

Используемые в протоколе числа s_A, s_B называются *одноразовыми подписями*. Название объясняется тем, что эти числа фактически являются вторыми частями подписи Шнорра:

ЭЦП Шнорра	Протокол MQV
$k \xleftarrow{R} \{1, 2, \ldots, q-1\}$	$u_A \xleftarrow{R} \{1, 2, \ldots, q-1\}$
$R \leftarrow kP$	$V_A \leftarrow u_A P$
$s_0 \leftarrow \varphi(R, X)$	$\varphi(V_A, X_A)$
$s_1 \leftarrow (k - s_0 d) \bmod q$	$s_A \leftarrow (u_A - \varphi(V_A, X_A) d_A) \bmod q$

Определение s_A без d_A (или s_B без d_B) соответствует задаче `Schnorr`, состоящей в построении поддельной подписи. Для криптографически надежных G и φ задача `Schnorr` признается трудной.

Обсудим стойкость протокола MQV.

Стойкость (*пассивный противник*). Пусть Виктор не вмешивается в выполнение протокола, а только перехватывает сообщения, которыми обмениваются стороны. Виктору требуется найти $K = s_A s_B P$. Используя данные перехвата, Виктор может вычислить $s_A P = V_A - \varphi(V_A, X_A) Q_A$ и $s_B P = V_B - \varphi(V_B, X_B) Q_B$. Но даже при этом для определения K ему требуется решить трудную задачу `CDH`.

Стойкость (*активный противник*). Пусть Виктор может вмешиваться в выполнение протокола. Пусть Виктор хочет выдать себя за Боба. Для этого он по V_A, Q_A, V_B и Q_B должен найти $V \in G^*$ и $K \in G$ такие, что

$$K = s s_A P, \quad sP = V - \varphi(V, X_B) Q_B, \quad s \in \{1, 2, \ldots, q-1\}.$$

Виктор может:

1. Выбрать V, определить sP, а затем найти s с последующим определением $K = s(s_A P)$. Но нахождение s есть трудная задача `DL`.
2. Выбрать V, определить sP, а затем найти K по sP и $s_A P$. Но это задача `CDH`.
3. Найти (V, s) такие, что $sP = V - \varphi(V, X_B) Q_B$, а затем определить K. Но нахождение пары (V, s) есть задача `Schnorr`.

Как видим, во всех рассмотренных случаях противник сталкивается с необходимостью решения некоторой трудной задачи.

5.9. Протокол TLS

Протокол TLS (Transport Layer Security) — это наиболее распространенный на сегодняшний день протокол защиты соединений между клиентом и сервером в сети Интернет. Протокол был разработан в 1996 году компанией Netscape и первоначально назывался SSL. Протокол прошел несколько существенных модернизаций. На данный момент наиболее совершенной считается редакция 1.2, принятая в качестве RFC 5246 в 2008 году.

TLS обеспечивает взаимную аутентификацию сторон протокола, конфиденциальность и контроль целостности передаваемых между сторонами данных. TLS встраивается в стек коммуникационных протоколов поверх транспортного уровня и обеспечивает защиту данных этого уровня.

TLS представляет собой набор субпротоколов различного назначения. В частности, в TLS имеется субпротокол Handshake (рукопожатия), с помощью которого стороны формируют общий секретный ключ и параллельно проводят аутентификацию друг друга.

В начале выполнения Handshake стороны обмениваются случайными двоичными словами `client_random`, `server_random`. Затем вырабатывают предварительный общий ключ `pre_master_secret` и определяют окончательный общий ключ

$$\texttt{master\_secret} = h(\texttt{pre\_master\_secret}, \texttt{client\_random}, \texttt{server\_random}),$$

где h — специальная хэш-функция.

RFC 5246 определяет следующие стандартные алгоритмы формирования общего ключа (A — клиент, B — сервер).

`DH_anon` (*протокол Диффи — Хеллмана без сертификатов*). Сервер пересылает клиенту описание группы G и свой открытый ключ $V_B = u_B P$. Клиент пересылает серверу свой открытый ключ $V_A = u_A P$. Стороны вычисляют общий ключ $u_A u_B P$, по которому строится `pre_master_key`. В протоколе не используются сертификаты и, таким образом, подлинность сторон не проверяется. Поэтому протокол не противостоит атаке «противник посередине» и его рекомендуется использовать только в специальных случаях.

`DH_fixed` (*протокол Диффи – Хеллмана с сертификатами*). Сервер пересылает клиенту сертификат, который содержит описание G и открытый ключ $Q_B = d_B P$. Клиент по запросу сервера передает статический открытый ключ $Q_A = d_A P$. Если запроса от сервера нет, то клиент передает эфемерный открытый ключ $V_A = u_A P$. Стороны вычисляют общий ключ $d_A d_B P$ или $u_A d_B P$, по которому определяется `pre_master_secret`.

`DHE` (*протокол Диффи – Хеллмана с эфемерными ключами*). Сервер пересылает клиенту сертификат, открытый ключ $Q_B = d_B P$ которого можно использовать для проверки ЭЦП. Затем сервер передает описание G, свой эфемерный открытый ключ $V_B = u_B P$ и подписывает эти данные вместе с `client_random`, `server_random` на своем личном ключе d_B. Клиент проверяет ЭЦП и передает свой открытый ключ $V_A = u_A P$. Стороны вычисляют общий ключ $u_A u_B P$, по которому строится `pre_master_secret`. Сертификат клиента может не передаваться.

Транспорт ключа. Сервер передает клиенту свой сертификат, открытый ключ которого можно использовать для шифрования. Клиент зашифровывает на этом ключе `pre_master_secret` и передает шифртекст серверу. Сервер выполняет расшифрование `pre_master_secret` на своем личном ключе.

5.10. Задания

Задание 5.1. Разработать протокол типа Диффи — Хеллмана для формирования общего ключа между n абонентами.

Задание 5.2. Пусть $g = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$ — матрица над полем $\mathbb{F}_3$ и $G = \langle g \rangle$ — циклическая группа, порожденная g. Доказать, что порядок G равняется 6. Стороны используют G для выработки общего ключа по протоколу Диффи — Хеллмана. Первая сторона выбирает $a = 4$, вторая сторона выбирает $b = 5$. Чему будет равняться общий ключ?

Задание 5.3. Пусть $g = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$ — матрица над полем $\mathbb{F}_5$ и $G = \langle g \rangle$ — циклическая группа, порожденная g. Доказать, что порядок G равняется 4. Стороны используют G для выработки общего ключа по протоколу Диффи — Хеллмана. Первая сторона выбирает $a = 3$, вторая сторона выбирает $b = 2$. Чему будет равняться общий ключ?

Задание 5.4. Пусть $g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 3 & 6 & 2 & 1 & 0 & 4 \end{pmatrix}$ — подстановка из $S(\mathbb{F}_7)$ и $G = \langle g \rangle$ — циклическая группа, порожденная g. Доказать, что порядок G равняется 10. Стороны используют G для выработки общего ключа по протоколу Диффи — Хеллмана. Первая сторона выбирает $a = 3$, вторая сторона выбирает $b = 4$. Чему будет равняться общий ключ?

Задание 5.5. Пусть $g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 6 & 1 & 2 & 0 & 4 & 3 \end{pmatrix}$ — подстановка из $S(\mathbb{F}_7)$ и $G = \langle g \rangle$ — циклическая группа, порожденная g. Доказать, что порядок G равняется 12. Стороны используют G для выработки общего ключа по протоколу Диффи — Хеллмана. Первая сторона выбирает $a = 3$, вторая сторона выбирает $b = 5$. Чему будет равняться общий ключ?

Задание 5.6. Пусть $G = \langle g \rangle$ — группа простого нечетного порядка q. Алгоритм решения задачи `CDH`: $(g, g^a, g^b) \mapsto g^{ab}$ может быть преобразован в алгоритм решения задачи `SqDH`: $(g, g^a) \mapsto g^{a^2}$. Доказать, что верно и обратное: алгоритм решения `SqDH` может быть преобразован в алгоритм решения `CDH`.

Задание 5.7. Пусть d и m — взаимно простые натуральные числа, $g\colon x \mapsto x + d \bmod m$ — подстановка на $\mathbb{Z}_m$. Найти порядок g. Почему использование группы $G = \langle g \rangle$ в протоколе Диффи — Хеллмана не является безопасным?

Задание 5.8. Проанализировать алгоритмы формирования общего ключа TLS и установить, какие из них обеспечивают защиту от «чтения назад».

5.11. Комментарии

Протоколы формирования общего ключа описываются в книгах [2, 10, 37, 41]. Имеется русскоязычный обзор [12], посвященный криптографическим протоколам в целом.

Протокол Диффи – Хеллмана введен в [21]. Протоколы MTI определены в [36], а протокол MQV — в [34]. Схема ЭЦП Шнорра введена в [42].

Протокол TLS версии 1.2 определен в СТБ 34.101.65 «Информационные технологии и безопасность. Протокол защиты транспортного уровня (TLS)».

СТБ 34.101.66 «Информационные технологии и безопасность. Протоколы формирования общего ключа» определяет протоколы BMQV, BSTS и BPACE. Первый протокол уточняет MQV, второй — известную схему STS [22], третий — схему PACE [16]. Протокол BSTS обеспечивает анонимность: противник, который не вступает во взаимодействие со сторонами, а только перехватывает их сообщения, не получает информации о том, какие стороны участвуют в протоколе. Протокол BPACE позволяет сторонам сформировать общий ключ, используя общий пароль. Противнику, который перехватывает все сообщения протокола (вступая или не вступая во взаимодействие со сторонами), вычислительно трудно определить пароль, даже если он короткий или низкоэнтропийный. Выполняя сеанс протокола с легальной стороной, противник может проверить только один вариант пароля.

ЛИТЕРАТУРА

1. Агиевич С. В., Афоненко А. А. О свойствах экспоненциальных подстановок. Вести НАН Беларуси, 2005, № 1. 106–112.

2. Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В. Основы криптографии. М.: Гелиос АРВ, 2005.

3. Кнут Д. Искусство программирования, том 1. Основные алгоритмы. М.: Вильямс, 2000.

4. Кнут Д. Искусство программирования, том 2. Получисленные алгоритмы. М.: Вильямс, 2000.

5. Левин Л. А. Универсальные задачи перебора. Проблемы передачи информации, IX, вып. 3, 115-116, 1973.

6. Логачев О. А., Сальников А. А., Смышляев С.В., Ященко В. В. Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2012.

7. Китаев А., Шень А., Вялый М. Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999.

8. Кузюрин Н. Н., Фомин С. А. Эффективные алгоритмы и сложность вычислений. М: Издательство МФТИ, 2007.

9. Лидл Р., Нидеррайтер Г. Конечные поля, в 2-х т. М.: Мир, 1988.

10. Столлингс В. Криптография и защита сетей. Принципы и практика. М.: Вильямс, 2002.

11. Токарева Н. Н. Симметричная криптография. Краткий курс: учебное пособие. Новосиб. гос. ун-т. Новосибирск, 2012.

12. Черемушкин А. В. Криптографические протоколы: основные свойства и уязвимости. Прикладная дискретная математика, 2009, приложение № 2, 115–150.

13. Agievich S. Two-stage allocations and the double Q-function. Electronic Journal of Combinatorics, 10, 2003, R21.

14. Arora S., Barak B. Computational Complexity. A Modern Approach. Cambridge University Press, 2007.

15. Bach E., Shallit J. Algorithmic Number Theory. Volume I: Efficient Algorithms. The MIT Press, 1997.

16. Bender J., Fischlin M., Kuegler D. Security Analysis of the PACE Key-Agreement Protocol. Cryptology ePrint Archive, Report 2009/624, 2009.

17. Beth T., Ding C. On Almost Perfect Nonlinear Permutations. Advances in Cryptology — EUROCRYPT'93 Proceedings, Springer-Verlag, 65–76, 1994.

18. Biham E., Shamir A. Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology. V. 4, 1991. P. 3–72.

19. Cohen H. A Course in Computational Algebraic Number Theory. Springer-Verlag, 1993.

20. Cook S. The complexity of theorem-proving procedures. Proc. 3rd STOC, 151–158, 1971.

21. Diffie W., Hellman M. New Directions in Cryptography. IEEE Transactions on Information Theory, IT-22, 644–654, 1976.

22. Diffie W., Oorschot P., Wiener M. Authentication and Authenticated Key Exchanges. Designs, Codes and Cryptography, 2(2): 107–125, 1992.

23. Elias P. Universal Codeword Sets and Representations of the Integers. IEEE Trans. Information Theory 21(2): 194-203, 1975.

24. Goldreich O. Computational Complexity. A Conseptual Perspective. Cambridge University Press, 2008.

25. Goldreich O. Foundations of Cryptography. Basic Tools. Cambridge University Press, 2004.

26. Goldwasser S., Bellare M. Lecture Notes on Cryptography. Avail. at: `http://cseweb.ucsd.edu/~mihir/papers/gb.pdf`, 2008.

27. Hankerson D., Menezes A., Vanstone S. Guide to Elliptic Curve Cryptography. N. Y.: Springer, 2004.

28. Hellman M. A Cryptanalytic Time-Memory Tradeoff. IEEE Trans. on Inf. Theory, vol. IT-26 (4), 1980, pp. 401–406.

29. Iwata T., Kurosawa K. OMAC: One-Key CBC MAC. Fast Software Encryption, FSE 2003, LNCS 2887, pp. 129–153. Springer-Verlag, 2003.

30. Johnson D. A Catalog of Complexity Classes. In: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A). MIT Press, 67–161, 1990.

31. Joux A. Multicollisions in iterated hash functions. application to cascaded constructions. In Matthew K. Franklin, editor, CRYPTO, volume 3152 of LNCS, pages 306-–316. Springer, 2004.

32. Karp R. Reducibility among combinatorial problems. Complexity of Computer Computations, R. E. Miller and J. W. Thatcher eds., 85–103, 1972.

33. Lachaud G., Wolfmann J. The Weights of the Orthogonals of the Extended Quadratic Binary Goppa Codes. IEEE Trans. Inform. Theory, Vol. 36, 686–692, 1990.

34. Menezes A., Qu M., Vanstone S. Some New Key Agreement Protocols Providing Mutual Implicit Authentication. Workshop on Selected Areas in Cryptography (SAC '95), 22–32, 1995.

35. Matsui M. Linear Cryptanalysis Method for DES Cipher. Advances in Cryptology — EUROCRYPT'93 Proceedings. LNCS 765, 1994. P. 386–397.

36. Matsumoto T., Takashima Y., Imai H. On Seeking Smart Public-key Distribution Systems. Transactions of the IECE of Japan, E69:99–106, 1986.

37. Menezes A., Van Oorshot P., Vanstone S. Handbook of Applied Cryptography. CRC Press, 1997.

38. Nyberg K. Differentially Uniform Mapping for Cryptography. Advances in Cryptology — EUROCRYPT'93 Proceedings, Springer-Verlag, 55–64, 1994.

39. Pratt V. Every Prime Has a Succint Certificate. SIAM J. Comput., 4(3): 214–220, 1974.

40. Rabin M. Digitalized signatures and public-key functions as intractable as factorization, Technical Report 212, MIT Laboratory for Computer Science, 1979.

41. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. N. Y.: John Wiley and Sons, 1996.

42. Schnorr C. P. Efficient Signature Generation by Smart Cards. J. Cryptology, 4(3): 161–174, 1991.

43. Stinson D. R. Cryptography. Theory and Practice. N. Y.: CRC Press, 1995.

44. Wegman M., Carter J. New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences, 22, 1981, 265–279.