1 Погружение в мир криптографии

Добро пожаловать в мир криптографии. Данная книга раскроет секреты этой потрясающей науки, которые могут стать полезными как для вашей карьеры, так и для общего развития. Вы познакомитесь с ключевыми алгоритмами, которыми славится криптография, а также откроете для себя некоторые новые алгоритмы, которые я разработал и применил на практике. Я надеюсь, что чтение не только доставит вам удовольствие, но и поможет в достижении ваших академических и профессиональных целей.

В главе 1 вы узнаете, что такое криптография, для чего она нужна и почему имеет такое большое значение в ИТ. Здесь будет представлен краткий обзор основных алгоритмов, разработанных за все время существования криптографии: от шифра Цезаря до шифра Вернама и других менее известных алгоритмов, например криптограммы Бейла. Далее будут подробно описаны алгоритм Ривеста, Шамира и Адлемана (RSA), алгоритм Диффи — Хеллмана, расширенный стандарт шифрования (AES), доказательство с нулевым разглашением, эллиптические кривые, гомоморфное шифрование, квантовая криптография и другие известные алгоритмы.

Эта глава позволит вам понять суть криптографии и основы обеспечения безопасности.

Введение в криптографию

Одним из наиболее важных аспектов изучения криптографии является понимание определений и обозначений. Мне никогда не нравились определения и обозначения, прежде всего потому, что я один использую обозначения, которые сам и ввел. Однако я осознаю, что это очень важное условие для того, чтобы понимать друг друга, особенно когда мы говорим о математике. Поэтому сейчас я представлю основные сведения и понятия, связанные с криптографией.

Начнем с определения алгоритма.

В математике и информатике *алгоритмом* называется конечная последовательность точно заданных компьютеризируемых инструкций.

Здесь возникает важный вопрос: а что такое шифр?

Шифр — это любая система, позволяющая преобразовать открытый текст (сообщение) в недоступный для понимания (шифротекст или криптограмму) (рис. 1.1).

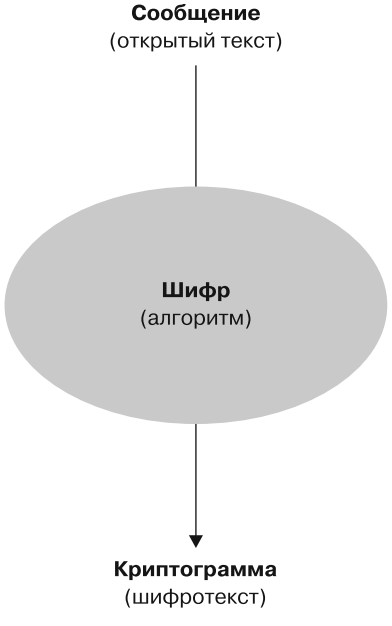

Рис. 1.1. Процесс шифрования

Для того чтобы шифр работал, нам нужно определить две операции: шифрование и расшифрование. Проще говоря, необходимо обеспечить секретность и безопасность сообщения в течение некоторого времени.

Определим M как совокупность всех сообщений (messages) и C как совокупность всех криптограмм (cryptograms).

Шифрование (encryption) — это операция, которая преобразует общее сообщение m в криптограмму с помощью функции E:

$$m \to f(E) \to c.$$

Расшифрование (decryption), или расшифровка[1], — это операция, которая возвращает криптограмме c ее первоначальный вид открытого текста m с помощью функции D:

$$c \rightarrow f(D) \rightarrow m.$$

Математически расшифровку можно выразить так:

$$D(E(m)) = m.$$

Следовательно, E и D являются обратными друг другу функциями, при этом функция E должна быть инъективной. Под *инъективностью* подразумевается то, что разные значения M должны соответствовать разным значениям C.

Обратите внимание: неважно, использую я прописные или строчные буквы, например (M) или (m), так как это не имеет никакого значения. Сейчас я добавил круглые скобки без какого-то скрытого смысла, однако дальше секретные параметры функции будут обозначаться квадратными скобками, чтобы можно было отличать их от известных параметров. Таким образом, секретное сообщение M будет обозначено $[M]$, как и любой другой секретный параметр. В данном случае наша цель заключается в том, чтобы просто показать, как работают алгоритмы, а их реализацию оставим инженерам.

Шифрование и расшифрование имеют еще один важный, можно сказать, ключевой компонент. Для того чтобы зашифровать или расшифровать сообщение, необходим ключ. В криптографии ключом называется параметр, который определяет функциональный выход криптографического алгоритма или шифра. Без него алгоритм бесполезен.

Мы определяем K как набор всех ключей (key), используемых для шифрования и расшифрования сообщений M, а k — как единственный ключ шифрования или расшифрования, который называется также сеансовым ключом. Эти два способа определения ключа (набор ключей — K, один ключ — k) будут использоваться всегда, независимо от типа ключа — закрытый или открытый.

Теперь, когда мы разобрались с основными криптографическими обозначениями, пришло время объяснить разницу между закрытыми и открытыми ключами.

- В криптографии *закрытый*, или *секретный*, *ключ*, обозначаемый $[K]$ или $[k]$, — это параметр шифрования или расшифрования, известный только одной или нескольким сторонам, участвующим в обмене секретными сообщениями.
- В криптографии *открытый ключ*, или (K), — это ключ шифрования, известный всем, кто хочет отправить секретное сообщение или аутентифицировать пользователя.

---

[1] В русском языке понятия расшифровки и дешифровки отличаются. Расшифрование — это восстановление открытого текста из шифротекста при наличии ключа. Дешифрование — это попытка восстановить текст без ключа, то есть взлом шифра.

В чем же основная разница между закрытыми и открытыми ключами?

Закрытый ключ используется как для шифрования, так и для расшифровки сообщения, а открытый — только для шифрования сообщения и проверки личности (цифровых подписей) людей и компьютеров. Это существенная и очень важная деталь, поскольку она определяет разницу между симметричным и асимметричным шифрованием.

Дам общее определение этих двух методов шифрования.

- В *симметричном шифровании* используется только один общий ключ как для шифрования, так и для расшифровки сообщения.
- В *асимметричном шифровании* для генерирования открытого ключа (для шифрования сообщения) задействуются несколько параметров, в то время как для расшифровки сообщения требуется только один закрытый ключ.

Как вы увидите далее, в симметричном шифровании один и тот же закрытый ключ применяется для шифрования и расшифрования сообщения, а в асимметричном шифровании закрытый ключ используется только для расшифрования. Открытые ключи применяются только в асимметричном шифровании для шифрования сообщения и обработки цифровой подписи. Функции этих двух типов ключей будут раскрыты позже, а пока необходимо запомнить, что закрытый ключ используется как в симметричном, так и в асимметричном шифровании, а открытый ключ — только в асимметричном. В мои планы не входит обсуждение академических определений и обозначений, поэтому, пожалуйста, постарайтесь просто понять, какова область применения каждого элемента и как они используются.

Одна из основных проблем в криптографии заключается в передаче ключа или обмене ключами. В свое время она вызвала бурную полемику в сообществе математиков и криптографов, поскольку было очень сложно определить, как передать ключ без физического контакта.

Например, если Алиса и Боб хотели обменяться ключом (до появления асимметричного шифрования), единственным надежным способом сделать это была физическая встреча. С массовым внедрением телекоммуникационных систем и Интернета это условие создало множество проблем. Первая проблема состояла в том, что связь через Сеть осуществлялась посредством обмена данными по небезопасным каналам. Как можно легко понять, если Алиса общается с Бобом по небезопасному публичному каналу связи, существует большая вероятность компрометации[1] закрытого ключа, что крайне опасно для безопасности и конфиденциальности коммуникаций.

---

[1] Компрометация — это ситуация, при которой конфиденциальная информация, ключи, пароли или системы становятся доступными посторонним лицам или теряют свою безопасность.

По этой причине возникает вопрос: *если мы используем симметричный шифр для защиты секретной информации, как обеспечить безопасность при обмене секретным ключом?*

Простой ответ таков: мы должны обеспечить *безопасность канала* связи для обмена ключом.

Кто-то может спросить: *а как обеспечить безопасность канала?*

Ответ, а точнее, несколько ответов вы найдете далее в этой книге. Даже в сложных военных ситуациях, например во время холодной войны, в легендарной красной линии между лидерами США и СССР использовались симметричные ключи связи. В наши дни принято применять асимметричное шифрование для обмена ключом. Затем, в следующем сеансе связи, этот ключ объединяется с симметричным шифрованием для шифрования передаваемых сообщений.

По многим причинам асимметричное шифрование является хорошим способом обмена ключами и подходит для аутентификации и цифровых подписей. С вычислительной точки зрения симметричное шифрование лучше, поскольку оно работает с ключами небольшой длины, что позволяет экономить производительные ресурсы и время. Так, алгоритмы симметричного шифрования эффективно обеспечивают безопасность, применяя ключи длиной всего 256–512 бит. Сравните это с более чем 4000 бит асимметричного шифрования RSA, например[1]. Подробнее о том, почему и как это возможно, я расскажу во время анализа алгоритмов асимметричного и симметричного шифрования.

В этой книге я проанализирую множество криптографических техник, однако, по большому счету, все алгоритмы можно разделить на два больших семейства: симметричное и асимметричное шифрование.

Для полного понимания криптографии будут полезны еще несколько определений.

- *Открытый текст.* В криптографии это незашифрованный текст или все, что можно представить в открытом доступе. Например, *(встретимся завтра в 10 утра)* — это открытый текст.
- *Шифротекст.* В криптографии это результат шифрования текста. Например, *«встретимся завтра в 10 утра»* может быть представлено в виде шифротекста *[x549559*ehebibcm3494]*.
- Как уже говорилось, для обозначения открытого текста и шифротекста я использую *разные виды скобок.* В частности, круглые скобки (...) обозначают

---

[1] В асимметричном шифровании можно использовать ключи длиной 1024 и менее бит. Но это уже небезопасно, есть риск дешифровки. В 2025 году рекомендуется использовать минимум 2048-битные ключи. В качестве примера можно привести рекомендации NIST (США): https://doi.org/10.6028/NIST.SP.800-57pt1r5.

открытый текст, а квадратные скобки [...] — шифротекст. Таким образом, сообщение, о котором говорилось ранее, — *[x549559*ehebibcm3494]* — считается секретным.

Двоичные числа, код ASCII и условные обозначения

Когда мы работаем с данными на компьютерах, мы обычно используем их в виде строк, состоящих из 0 и 1 (бит). Таким образом, числа из обычной системы счисления с основанием 10 можно перевести в биты (система счисления с основанием 2). Посмотрим, как работает механизм такого преобразования. Например, число 123 можно записать с основанием 10 как[1]:

$$1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0.$$

Аналогичным образом можем перевести число с основанием 10 в число с основанием 2. Возьмем в качестве примера число 29 (рис. 1.2)[2].

Перевод числа 29 в число с основанием 2

Шаг	Операция	Результат	Остаток	Перевод (основание 2)
Шаг 1	29 / 2	14	1	**(11101)2**
Шаг 2	14 / 2	7	0	
Шаг 3	7 / 2	3	1	
Шаг 4	3 / 2	1	1	
Шаг 5	1 / 2	0	1	

Рис. 1.2. Перевод числа 29 в число с основанием 2 (биты)

[1] Такой формат записи числа называется позиционным разложением, или разложением числа по основанию системы счисления.

[2] При переводе числа в другую систему счисления его последовательно делят на основание новой системы до тех пор, пока частное не станет нулем. Запись числа в новой системе формируется из остатков, выписанных в обратном порядке.

Деление с остатком — очень популярная операция в криптографии, потому что *модульная арифметика* основана на концепции остатков. Мы углубимся в эту тему в следующем разделе, когда будем говорить о простых числах и модульной арифметике.

Чтобы компьютеры могли кодировать буквы, Американская ассоциация стандартов в 1960 году изобрела код ASCII, переводящий символы в двоичную систему.

На сайте ASCII имеется следующее определение:

> *«ASCII расшифровывается как American Standard Code for Information Interchange (Американская стандартная кодировка для обмена информацией). Это семибитный символьный код, в котором каждый бит представляет собой уникальный символ».*

На рис. 1.3 приведен пример таблицы кодов ASCII с первыми десятью знаками и управляющими символами[1].

DEC	OCT	HEX	BIN	Символ	HTML-код	Описание
0	000	00	00000000	NUL	&#000;	Нулевой байт
1	001	01	00000001	SOH	&#001;	Начало заголовка
2	002	02	00000010	SIX	&#002;	Начало текста
3	003	03	00000011	ETX	&#003;	Конец текста
4	004	04	00000100	EOT	&#004;	Конец передачи
5	005	05	00000101	ENQ	&#005;	«Прошу подтверждения!»
6	006	06	00000110	ACK	&#006;	«Подтверждаю!»
7	007	07	00000111	BEL	&#007;	Звуковой сигнал — звонок
8	010	08	00001000	BS	&#008;	Возврат на один символ (BACKSPACE)
9	011	09	00001001	HT		Табуляция
10	012	0A	00001010	LF
Перевод строки

Рис. 1.3. Первые десять знаков и символов, выраженных в коде ASCII

В главе 4 вы познакомитесь с шестнадцатеричной (hex) и восьмеричной (octal) системами счисления, а пока ограничимся двоичной системой.

Обратите внимание на то, что в своих реализациях, выполненных с помощью исследовательского программного обеспечения *Wolfram Mathematica*, я буду часто использовать символ 88 в качестве X для обозначения номера шифруемого

[1] Управляющие символы — это специальные символы в кодировках, которые не отображаются как обычный знак, а используются для управления устройствами ввода-вывода или форматированием данных.

сообщения. Как можно увидеть в следующем примере, в коде ASCII число 88 соответствует символу *X*:

88 130 58 01011000 *X* X прописная *X*.

Последняя теорема Ферма, простые числа и модульная арифметика

Говоря о криптографии, мы всегда должны держать в голове, что эта дисциплина, по сути, связана с математикой и логикой[1]. Прежде чем приступить к объяснению *последней теоремы Ферма*, я бы хотел ввести еще несколько основных обозначений, которые будут использоваться в книге, что поможет избежать путаницы и лучше понимать текст. Важно знать, что символы =, ≡ (эквивалент) и := (его можно найти в системе Mathematica для операции =) нужны только для того, чтобы сказать, что два элемента соответствуют друг другу в равной степени вне зависимости от того, в каком контакте они представлены: в конечном поле (не волнуйтесь, я объясню этот термин), информатике или обычной алгебре. Математики могут прийти в ужас от такого определения, но я надеюсь, что вы будете искать суть, а не единообразие.

Другой символ, ≈ (приблизительно), может использоваться для обозначения подобных приближенных элементов.

Вы также будете встречать символ ^ (возведение в степень) в качестве классического способа выражения степени: $a\hat{}x$ обозначает «a в степени x», или a^x.

Символ ≠, как вы должны помнить из курса средней школы, означает «не равно», что в модульной арифметике представлено в виде ≢, то есть «не эквивалентно».

Тем не менее вы всегда получите объяснение уравнений. Даже если вы не очень хорошо ладите с математическими и логическими обозначениями, то можете положиться на описания уравнения. В любом случае я буду объяснять каждое новое определение или символ.

Как вы, скорее всего, знаете, простое число — это целое число, которое может делиться только на себя и на 1, например 2, 3, 5, 7... 23... 67... *p*.

Простые числа являются краеугольным камнем математики, поскольку все остальные числа происходят от них. Числа, которые имеют делители, отличные от 1 и самого себя, называются составными. Это, например, 4, 6, 8, 9 и т. д. Вы увидите, что использование в криптографии составного числа вместо простого может стать причиной серьезного снижения безопасности или возникновения атаки (см. главу 5).

---

[1] Под логикой здесь и далее в книге подразумевается математическая логика. Это раздел математики, который формально описывает правильные рассуждения и правила вывода новых утверждений из уже известных.

Теперь посмотрим, что такое последняя теорема Ферма, где она применяется и чем полезна для нас.

Последняя, или великая, теорема Ферма — одна из лучших и самых красивых теорем классической математики, строго связанная с простыми числами, которая имеет основополагающее значение для криптографии. Согласно Википедии,

> *«в теории чисел последняя теорема Ферма (иногда называемая догадкой Ферма, особенно в старых текстах) утверждает, что для любого натурального числа $n > 2$ уравнение $a^n + b^n = c^n$ не имеет решений в целых ненулевых числах a, b, c. С древности было известно, что случаи $n = 1$ и $n = 2$ имеют бесконечно много решений».*

Другими словами, для любой степени $n \geqslant 3$ не существует целого числа a, b или c, которое удовлетворяет уравнению[1]:

$$a^n + b^n = c^n.$$

Почему эта теорема так важна для нас? С каждым новым изученным алгоритмом вы будете все лучше понимать ее значение. По сути, последняя теорема Ферма имеет непосредственное отношение к простым числам. Учитывая свойства простых чисел, продемонстрировать последнюю теорему Ферма можно с помощью уравнения:

$$a^p + b^p \neq c^p,$$

где p — любое простое число больше 2.

Сам Ферма в одной из своих работ отметил, что у него было прекрасное подтверждение теоремы, *которое было слишком обширным, чтобы поместиться на полях его заметок*, однако оно так и не было найдено.

Доказательство Уайлса заняло более 200 страниц. В последней версии оно было сокращено до 130 страниц и оказалось чрезвычайно сложным для понимания. Основано доказательство на эллиптических кривых, которые характеризуются тем, что имеют особую модулярную форму. Уайлс пришел к этому выводу спустя семь лет работы и в 1994 году представил свое доказательство на одном из математических конгрессов. Вы частично познакомитесь с логикой, использованной в доказательстве Уайлса, в главе 7. В данный момент мы просто примем тот факт, что Уайлс подтвердил последнюю теорему Ферма с помощью гипотезы Таниямы — Шимуры, утверждающей, что *эллиптические кривые над полем рациональных чисел связаны с модулярными формами*. Опять же не волнуйтесь, если эта гипотеза покажется вам слишком сложной: по мере нашего продвижения вперед она начнет обретать смысл.

---

[1] При $n = 2$ уравнение принимает вид теоремы Пифагора и имеет бесконечно много целочисленных решений (например, $3^2 + 4^2 = 5^2$). Ключевое отличие последней теоремы Ферма состоит в том, что она касается степеней строго больше 2.

Мы тщательно проанализируем последнюю теорему Ферма в главе 6, где я представлю основанный на ней алгоритм MB09, а также другие инновационные алгоритмы открытых и закрытых ключей. Кроме того, в главе 7 мы разберем криптографическое применение эллиптических кривых.

Ферма, как и многие другие математики, был одержим простыми числами. Он изучал простые числа и их свойства всю жизнь. Он пытался найти общую формулу для обозначения всех простых чисел во Вселенной, но, к сожалению, как и многие другие математики, сумел создать формулу только для некоторых из них. Далее приводится формула, по которой находят простые числа Ферма, где n — некоторое целое положительное число:

$$p = 2^{2^n} + 1.$$

Если вместо n подставить целые числа, то можно получить простые числа:

- $n = 1, p = 5$;
- $n = 2, p = 17$;
- $n = 3, p = 65$ (не простое число);
- $n = 4, p = 257$.

Возможно, наиболее известной, но очень похожей на формулу Ферма является формула поиска простых чисел Мерсенна, где n — некоторое целое положительное число:

$$p = 2^n - 1.$$

В результате получается:

- $n = 1, p = 1$;
- $n = 2, p = 3$;
- $n = 3, p = 7$;
- $n = 4, p = 15$ (не простое число);
- $n = 5, p = 31$.

Несмотря на бесчисленные попытки найти формулу, в которой были бы представлены все простые числа, до сих пор никому не удалось сделать это.

Great Internet Mersenne Prime Search (GIMPS) — это исследовательский проект, цель которого состоит в нахождении новых и самых больших простых чисел с помощью формулы Мерсенна.

На сайте GIMPS можно узнать следующее.

- Все экспоненты ниже 53 423 543 были проверены и подтверждены.
- Все экспоненты ниже 92 111 363 были проверены хотя бы раз.
- Найдено 51-е простое число Мерсенна!

- Сообщение от 21 декабря 2018 года: «В рамках проекта Great Internet Mersenne Prime Search (GIMPS) обнаружено самое большое из известных простых чисел — $2^{82\,589\,933} - 1$, состоящее из 24 862 048 цифр. Компьютер, добровольно предоставленный Патриком Ларошем из Окалы, штат Флорида, сделал эту находку 7 декабря 2018 года. Новое простое число, также известное как M82589933, вычисляется путем умножения 82 589 933 двоек и вычитания единицы из полученного произведения. Оно более чем на полтора миллиона цифр больше предыдущего рекордного простого числа».

Кроме того, GIMPS — это, вероятно, первый децентрализованный пример того, как можно разделить мощности процессора и компьютера для достижения общей цели. Почему же поиск больших простых чисел вызывает у людей такой интерес?

На этот вопрос есть как минимум три ответа: страсть к чистому исследованию, денежное вознаграждение, причитающееся тем, кто находит бóльшие простые числа, и, наконец, потому, что простые числа так же важны для криптографии, как кислород для человека. Именно по этой причине за открытие больших простых чисел полагается вознаграждение.

Вскоре вы поймете, что большинство алгоритмов нового поколения работают с простыми числами. Но как определить, является ли число простым?[1]

В математике существует значительная разница в вычислительной сложности операций умножения и деления. Деление требует гораздо больше вычислительных ресурсов, чем умножение. Другими словами, довольно просто возвести $2x$ (где x — огромное число) в степень и в то же время крайне сложно найти делители этого числа.

Из-за этого математики, такие как Ферма, пытались найти алгоритмы, которые облегчили бы эти вычисления.

В области простых чисел Ферма вывел еще одну очень интересную теорему, известную как *малая теорема Ферма*. Прежде чем мы перейдем к ней, необходимо прояснить, что такое модульная арифметика и как с ее помощью выполнять вычисления.

Самый простой способ понять суть модульной арифметики — вспомнить часы. Когда мы говорим: «Эй, мы можем встретиться в час», мы имеем в виду, что 1 — это первый час после 12, когда часы заканчивают круговой оборот.

Таким образом, можно сказать, что мы бессознательно вычисляем по модулю 12, что записывается как mod 12, где отсчет целых чисел *циклически повторяется*

---

[1] Если вам нужно проверить, является ли число простым, можете воспользоваться базой известных простых чисел на сайте factordb.com. Обозначения можно посмотреть здесь: factordb.com/status.html.

после достижения определенного значения (в данном случае 12), называемого модулем.

Технически результат вычисления с модулем состоит из остатка от деления числа на модуль.

Так, в нашем примере с часами:

$$13 \equiv 1 \pmod{12}.$$

Это означает, что 13 *конгруэнтно* с 1 по модулю 12. Здесь *конгруэнтность* означает равенство. Другими словами, мы можем сказать, что остаток от деления 13 / 12 равен 1 (рис. 1.4).

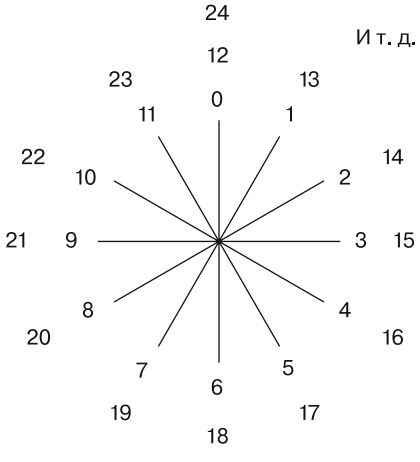

Рис. 1.4. Пример модульной арифметики с часами

Малая теорема Ферма гласит:

> *«Если p — простое число, то любое целое число a, возведенное в степень простого числа p, будет результатом уравнения* $a^p \equiv a \pmod{p}$*».*

Например, если $a = 2$ и $p = 3$, то $2^3 = 2 \pmod 3$. Другими словами, мы определяем остаток от деления 8 / 3 = 2 с остатком 2.

Малая теорема Ферма лежит в основе теста простоты Ферма и является одним из фундаментальных разделов элементарной теории чисел.

Эта теорема утверждает, что число *p*, *вероятно, является простым* в следующем случае: $a^p \equiv a \pmod{p}$.

Теперь, когда вы освежили свои знания о переводе числа в биты, увидели, как выглядит код ASCII, и изучили основные понятия математики и логики, можем начать путешествие в криптографию.