СОДЕРЖАНИЕ

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге, – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте *www.dmkpress.com* на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу *dmkpress@gmail.com*. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и No Starch Press очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

ВСТУПИТЕЛЬНОЕ СЛОВО

Если вы читали какие-нибудь книги по компьютерной безопасности, то, наверное, встречались с распространенным взглядом на криптографию. «Криптография, – говорят авторы, – самое прочное звено в цепочке». Оценка похвальная, но не вполне точная. Если криптография – действительно самая прочная часть системы, то зачем тратить время на ее улучшение, когда есть много других частей, нуждающихся во внимании?

И если попытаться сформулировать главный урок, который я хочу преподать этой книгой, то он состоит в том, что такой взгляд на криптографию, мягко говоря, идеализирован, а по сути является мифом. Криптография прочна *в теории*, а на практике столь же подвержена ошибкам, как и любой другой аспект системы безопасности. Особенно это относится к случаям, когда криптографическую систему реализуют непрофессионалы, не имеющие достаточного опыта и пренебрегающие деталями, – а таковы многие развернутые в настоящее время системы. Хуже того – если реализация криптографической системы содержит ошибки, то проявления их зачастую оказываются весьма скандальными.

Но какое вам до этого дело, и зачем нужна эта книга?

Когда почти двадцать лет назад я только начинал заниматься прикладной криптографией, разработчикам ПО была доступна лишь отрывочная и не самая актуальная информация. Криптографы придумывали алгоритмы и протоколы, а программисты, реализовывая их, создавали запутанные, плохо документированные библиотеки, пред-

назначенные в первую очередь для других специалистов. Существовало – и до сих пор существует – глубокая пропасть между теми, кто знает и понимает криптографические алгоритмы, и теми, кто этими алгоритмами пользуется (или на свой страх и риск пренебрегает ими). На рынке имеется очень немного достойных книг, и еще меньше полезных программисту-практику.

Результаты такого положения удручают. Я имею в виду факты компрометации, о которых говорят такие ярлыки, как «CVE» или «Серьезность: высокая», а в некоторых особо тревожных ситуациях – атаки, которые на слайдах снабжены грифом «СОВЕРШЕННО СЕКРЕТНО». Если вам и известны некоторые особо знаменитые примеры, то только потому, что они затрагивают системы, от которых зависит ваша работа. Многие проблемы подобного рода возникают, потому что криптография – сложная и математически элегантная теория, а специалисты по ней не потрудились поделиться знаниями с инженерами, которые пишут программный код.

По счастью, ситуация начинает меняться, и пример тому – эта книга.

Книга «О криптографии всерьез» написана одним из самых известных экспертов по прикладной криптографии, однако ориентирована не на других специалистов того же профиля. Но и поверхностным обзором этой дисциплины она тоже не является. Напротив, она содержит скрупулезное и современное обсуждение криптографической техники, призванное помочь стать лучше практикам, собирающимся подвизаться в этой области. Вы узнаете не только о том, как работают алгоритмы, но и как использовать их в реальных системах.

Изложение начинается с рассмотрения основных криптографических примитивов, в т. ч. блочных шифров, схем шифрования с открытым ключом, функций хеширования и генераторов случайных чисел. В каждой главе приводятся примеры работы алгоритмов и объясняется, что следует и чего *не* следует делать. В последних главах рассматриваются темы повышенной трудности, например TLS, а также будущее криптографии – что делать после того, как придут квантовые компьютеры и усложнят нам жизнь.

Конечно, одна книга не в состоянии решить все наши проблемы, но знания накапливаются по крупицам. В этой книге таких крупиц много. Быть может, даже достаточно для того, чтобы развернутые на практике криптографические системы стали, наконец, соответствовать тем высоким требованиям, которые к ним предъявляются.

Полезного вам чтения.

Мэттью Д. Грин,
профессор Института информационной безопасности
университета Джонса Хопкинса

ПРЕДИСЛОВИЕ

Я написал книгу, которую хотел бы иметь, когда только начал изучать криптографию. В 2005 году под Парижем я учился в магистратуре и с предвкушением записался на спецкурс по криптографии. Увы, спецкурс был отменен, потому что не набралось достаточного количества желающих. «Криптография – это слишком сложно», – говорили студенты и массово записывались на курсы по компьютерной графике и базам данных.

С тех пор слова «криптография – это сложно» я слышал неоднократно. Но так ли сложна криптография в действительности? Чтобы играть на музыкальном инструменте, овладеть языком программирования или воплотить на практике достижения какой-нибудь увлекательной дисциплины, необходимо изучить определенные концепции и символы, но для этого необязательно иметь докторскую степень. Я думаю, что это относится и к желанию стать компетентным криптографом. А также полагаю, что криптография считается такой трудной наукой, потому что криптографы не приложили достаточных усилий к ее преподаванию.

И есть еще одна причина, по которой я считаю эту книгу необходимой: криптография превратилась в довольно разветвленную область знаний. Чтобы сделать что-то полезное и важное в криптографии, нужно разбираться в *смежных* областях: как работают компьютеры и сети, что нужно пользователям и системам и как противник может злонамеренно воспользоваться алгоритмами и их реализациями. Иными словами, необходима связь с реальностью.

Подход, принятый в этой книге

Первоначально книга называлась «Crypto for Real», чтобы подчеркнуть практически ориентированный, деловой подход, которому я намеревался следовать. Я хотел не столько опростить криптографию, сколько связать с ее с реальными приложениями. Я привожу примеры исходного кода и описываю реальные ошибки и кошмарные истории.

Помимо четкой связи с реальностью, в основу книги положены еще два краеугольных камня: простота и современность. Упрощаю я изложение только по форме, не жертвуя содержанием: представляю много нетривиальных идей, но без скучного математического формализма. Я хочу, чтобы читатель понял основополагающие идеи криптографии, это кажется мне важнее, чем запоминание бесконечных формул. Что касается современности, то я рассматриваю последние теоретические результаты и приложения, например протокол TLS 1.3 и криптографию в постквантовую эпоху. Я не обсуждаю детали устаревших или небезопасных алгоритмов, например DES или MD5. Исключением является алгоритм RC4, но и он включен только для того, чтобы продемонстрировать, в чем его слабость и как работают потоковые шифры такого рода.

Книга «Криптография всерьез» не является ни путеводителем по криптографическому программному обеспечению, ни сводом технических спецификаций – такие вещи легко найти в сети. Ее главное назначение – пробудить у вас интерес к криптографии и попутно рассказать о ее фундаментальных концепциях.

Для кого предназначена эта книга

Во время работы над книгой я часто представлял себе читателя как разработчика, который оказался вынужден иметь дело с криптографией, но так и остался в растерянности после чтения заумных учебников и научных статей. Разработчики часто должны – и хотят – лучше понимать криптографию, чтобы избежать неудачных проектных решений, и я надеюсь, что в этом моя книга поможет.

Но если вы не являетесь разработчиком, тоже ничего страшного! Чтение книги не потребует от вас владения навыками кодирования, она доступна любому, кто знаком с основами информатики и математикой в объеме технического вуза (начала теории вероятностей, арифметики по модулю и т. д.).

Но, несмотря на сравнительную доступность книги, для получения от нее максимальной пользы все же требуется приложить некоторые усилия. Мне приходит на ум аналогия с альпинизмом: автор прокладывает путь, снабжает вас веревками и ледорубом, но покорить гору вам придется самостоятельно. Изучение изложенных в книге идей потребует труда, но в конце преодолевшего все препятствия ждет награда.

Структура книги

Книга состоит из четырнадцати глав, разбитых на четыре части. По большей части главы независимы, за исключением главы 9, в которой заложен фундамент для трех последующих глав. Но я все же рекомендую сначала прочитать первые три главы.

Основы

- **Глава 1 «Шифрование».** Здесь вводится понятие безопасного шифрования, начиная со слабых шифров с использованием карандаша и бумаги и заканчивая стойкими рандомизированными шифрами.
- **Глава 2 «Случайность».** Описывается, как работает генератор псевдослучайных чисел, когда такой генератор считается безопасным и как его безопасно использовать.
- **Глава 3 «Криптографическая безопасность».** Обсуждается теоретическая и практическая безопасность, сравнивается предположительная и доказуемая безопасность.

Симметричные криптографические системы

- **Глава 4 «Блочные шифры».** Рассматриваются шифры, обрабатывающие сообщения поблочно. Основное внимание уделяется самому известному из них, Advanced Encryption Standard (AES).
- **Глава 5 «Потоковые шифры».** Описываются шифры, порождающие поток случайных на первый взгляд битов, которые объединяются с сообщением операцией XOR.
- **Глава 6 «Функции хеширования».** Функции хеширования – чуть ли не единственный алгоритм, не нуждающийся в секретном ключе, и при этом один из самых распространенных строительных блоков в криптографии.
- **Глава 7 «Хеширование с секретным ключом».** Объясняется, что будет, если соединить функцию хеширования с секретным ключом, и как этим можно воспользоваться для аутентификации сообщений.
- **Глава 8 «Шифрование с аутентификацией».** На примерах описываются алгоритмы, которые могут одновременно зашифровать и аутентифицировать сообщение, в частности стандарт AES-GCM.

Асимметричные криптографические системы

- **Глава 9 «Трудные задачи».** Здесь заложен теоретический фундамент шифрования с открытым ключом; используется нотация из теории вычислительной сложности.
- **Глава 10 «RSA».** В алгоритме RSA задача разложения на множители применяется для построения схем безопасного шифрования и цифровой подписи с помощью простых арифметических операций.

- **Глава 11 «Протокол Диффи–Хеллмана».** Идея асимметричной криптографии обобщается на понятие совместной выработки ключей, когда две стороны вырабатывают секретное значение, используя только несекретные данные.
- **Глава 12 «Эллиптические кривые».** Простое введение в эллиптическую криптографию – самый быстрый вид асимметричных криптографических систем.

Приложения

- **Глава 13 «Протокол TLS».** Рассматривается протокол Transport Layer Security (TLS), пожалуй, самый важный для безопасности сетей.
- **Глава 14 «Квантовая и постквантовая криптография».** В этой заключительной главе с оттенком научной фантастики обсуждаются квантовые вычисления и новый вид криптографии.

Благодарности

Хочу поблагодарить Яна, Энни и других сотрудников издательства No Starch, принявших участие в подготовке этой книги, а особенно Билла, который поверил в проект с самого начала, терпеливо усваивал трудные темы и превращал мои беспорядочные черновики в читаемый текст. Я также благодарен Лорел, которая вносила мои многочисленные поправки и благодаря которой книга выглядит так симпатично.

Что касается технической стороны, то книга содержала бы куда больше ошибок и неточностей, если бы не помощь следующих лиц: Джон Каллас, Билл Кокс, Нильс Фергюсон, Филипп Йованович, Сэмюэл Нивс, Дэвид Рейд, Филлип Рогуэей, Эрик Тьюз, – а также читателей предварительной версии, сообщавших о найденных ошибках. Наконец, я благодарю Мэтта Грина, написавшего вступительное слово.

Я также выражаю благодарность своему работодателю, компании Kudelski Security, выделившей мне время для работы над книгой. Наконец, моя глубочайшая благодарность Александре и Мелине за поддержку и терпение.

Лозанна, 17.05.2017 (три простых числа)

СПИСОК АББРЕВИАТУР

AE	authenticated encryption (шифрование с аутентификацией)
AEAD	authentication encryption with associated data (шифрование с аутентификацией и ассоциированными данными)
AES	Advanced Encryption Standard (улучшенный стандарт шифрования)
AES-NI	AES native instructions (AES с платформенными командами)
AKA	authenticated key agreement (совместная выработка ключей с аутентификацией)
API	application program interface (интерфейс прикладной программы)
ARX	add-rotate-XOR
ASIC	application-specific integrated circuit (специализированная заказная интегральная схема)
CA	certificate authority (удостоверяющий центр, УЦ)
CAESAR	Конкурс шифрования с аутентификацией: безопасность, применимость и надежность
CBC	cipher block chaining (режим сцепления блоков шифртекста)
CCA	chosen-ciphertext attack (атака с подобранным шифртекстом)
CDH	computational Diffie–Hellman (предположение о вычислительной трудности задачи Диффи–Хеллмана)
CMAC	cipher-based MAC (имитовставка на основе блочного шифра)
COA	ciphertext-only attack (атака на основе шифртекста)
CPA	chosen-plaintext attack (атака с подобранным открытым текстом)
CRT	Chinese remainder theorem (китайская теорема об остатках)
CTR	режим счетчика
CVP	closest vector problem (задача о ближайшем векторе)
DDH	decisional Diffie–Hellman (предположение Диффи–Хеллмана о распознавании)

DES	Data Encryption Standard (стандарт шифрования данных)
DH	Diffie–Hellman
DLP	discrete logarithm problem (задача дискретного логарифмирования)
DRBG	deterministic random bit generator (детерминированный генератор случайных битов)
ECB	electronic codebook (режим электронной кодовой книги)
ECC	elliptic curve cryptography (эллиптическая криптография)
ECDH	elliptic curve Diffie–Hellman (эллиптический метод Диффи–Хеллмана)
ECDLP	elliptic-curve discrete logarithm problem (задача дискретного логарифмирования на эллиптической кривой)
ECDSA	elliptic-curve digital signature algorithm (алгоритм цифровой подписи на эллиптической кривой)
FDH	Full Domain Hash (полный хеш домена)
FHE	fully homomorphic encryption (полностью гомоморфное шифрование)
FIPS	Federal Information Processing Standards (Федеральный стандарт обработки информации)
FPE	format-preserving encryption (шифрование с сохранением формата)
FPGA	field-programmable gate array (программируемая пользователем вентильная матрица, ППВМ)
FSR	feedback shift register (регистр сдвига с обратной связью)
GCD	greatest common divisor (наибольший общий делитель, НОД)
GCM	Galois Counter Mode (режим счетчика с аутентификацией Галуа)
GNFS	general number field sieve (общий метод решета числового поля)
HKDF	HMAC-based key derivation function (функция формирования ключа на основе HMAC)
HMAC	hash-based message authentication code (имитовставка на основе функции хеширования)
HTTPS	HTTP Secure (безопасный HTTP)
IND	indistinguishablity (неразличимость)
IP	Internet Protocol
IV	initial value (начальное значение)
KDF	key derivation function (функция формирования ключа)
KPA	known-plaintext attack (атака с известным простым текстом)
LFSR	linear feedback shift register (линейный регистр сдвига с обратной связью)
LSB	least significant bit (младший бит)
LWE	learning with errors (обучение с ошибками)
MAC	messsage authentication code (имитовставка)
MD	message digest (дайджест сообщения)
MitM	meet-in-the-middle (метод встречи посередине)
MQ	multivariate quadratics (многомерные системы квадратичных уравнений)
MQV	Menezes–Qu–Vanstone (протокол Менезеса–Кью–Вэнстоуна)
MSB	most significant bit (старший бит)

MT	Mersenne Twister (вихрь Мерсенна)
NFSR	nonlinear feedback shift register (нелинейный регистр сдвига с обратной связью)
NIST	National Institute of Standards and Technology (Национальный институт стандартов и технологий)
NM	non-malleability (неподатливость)
OAEP	Optimal Asymmetric Encryption Padding (оптимальное асимметричное шифрование с дополнением)
OCB	offset codebook (режим кодовой книги со смещением)
P	polynomial time (полиномиальное время)
PLD	programmable logic device (программируемое логическое устройство)
PRF	pseudorandom function (псевдослучайная функция)
PRNG	pseudorandom number generator (генератор псевдослучайных чисел)
PRP	pseudorandom permutation (псевдослучайная перестановка)
PSK	pre-shared key (предварительно разделенный ключ)
PSS	Probabilistic Signature Scheme (вероятностная схема подписи)
QR	quarter-round (четверть раунда)
QRNG	quantum random number (квантовый генератор случайных чисел)
RFC	request for comments (запрос на комментарии)
RNG	random number generator (генератор случайных чисел)
RSA	Rivest–Shamir–Adleman (алгоритм Ривеста–Шамира–Адлемана)
SHA	Secure Hash Algorithm (безопасный алгоритм хеширования)
SIS	short integer solution (короткое целочисленное решение)
SIV	synthetic IV (синтетическое начальное значение)
SPN	substitution–permutation network (подстановочно-перестановочная сеть)
SSH	Secure Shell (безопасная оболочка)
SSL	Secure Socket Layer (уровень безопасных сокетов)
TE	tweakable encryption (настраиваемое шифрование)
TLS	Transport Layer Security (безопасность транспортного уровня)
TMTO	time-memory trade-off (компромисс между временем и памятью)
UDP	User Datagram Protocol (протокол пользовательских дейтаграмм)
UH	universal hash (универсальная функция хеширования)
WEP	Wired Equivalent Privacy (протокол безопасности в беспроводных сетях)
WOTS	Winternitz one-time signature (одноразовая подпись Винтерница)
XOR	exclusive OR (исключающее ИЛИ)

1

ШИФРОВАНИЕ

Шифрование – главное применение криптографии; его цель – сделать данные непонятными и тем самым обеспечить их *конфиденциальность*. Для шифрования используется алгоритм, называемый *шифром*, и секретное значение, называемое *ключом*; не зная секретного ключа, невозможно получить ни одного бита зашифрованного сообщения, не говоря уже о том, чтобы его дешифрировать.

В данной главе предметом нашего внимания будет *симметричное шифрование*, его простейшая разновидность. В этом случае для дешифрирования используется тот же ключ, что для шифрования (в отличие от *асимметричного шифрования*, или *шифрования с открытым ключом*, когда ключи шифрования и дешифрирования различны). Мы начнем с рассмотрения самых слабых форм симметричного шифрования – классических шифров, способных устоять только перед совсем необразованным противником, а затем перейдем к самым стойким шифрам, взломать которые вообще невозможно.

Основы

В контексте шифрования *открытым текстом* называется незашифрованное сообщение, а *шифртекстом* – зашифрованное сообщение. Шифр состоит из двух функций: *шифрование* преобразует открытый текст в шифртекст, а *дешифрирование* производит обратное преобразование шифртекста в открытый. Но часто говорят «шифр», имея в виду «шифрование». Например, на рис. 1.1 показан шифр **E**, представленный прямоугольником, который принимает открытый текст *P* и ключ *K* и порождает на выходе шифртекст *C*. Я буду записывать это соотношение в виде $C = \mathbf{E}(K, P)$. Аналогично, когда шифр работает в режиме дешифрирования, я буду писать $\mathbf{D}(K, C)$.

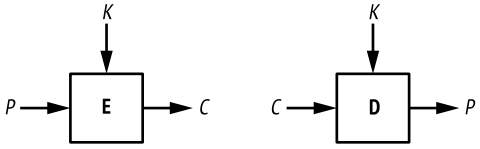

Рис. 1.1. Простейшее шифрование и дешифрирование

Примечание *Для некоторых шифров размер шифртекста равен размеру открытого текста, для других он немного больше. Но никогда шифртекст не может быть короче открытого текста.*

Классические шифры

Классическими называются шифры, появившиеся раньше компьютеров и потому применяемые к буквам, а не к битам. Они гораздо проще современных шифров типа DES – например, в Древнем Риме или во время Первой мировой войны для шифрования сообщения нельзя было воспользоваться всей мощью компьютера, так что приходилось довольствоваться бумагой и карандашом. Существует много классических шифров, но наиболее известны шифры Цезаря и Виженера.

Шифр Цезаря

Шифр Цезаря назван так, потому что, согласно древнеримскому историку Светонию, им пользовался Юлий Цезарь. Сообщение шифруется путем сдвига каждой буквы на три позиции вправо по алфавиту с оборотом по достижении Z. Например, ZOO шифруется как CRR, результатом дешифрирования FDHVDU является CAESAR и т. д., как показано на рис. 1.2. В числе 3 нет ничего особенного, просто так производить вычисления в уме проще, чем при выборе, скажем, 11 или 23.

Взломать шифр Цезаря проще простого: чтобы дешифрировать заданный шифртекст, нужно просто сдвинуть каждую букву на три позиции влево. Однако шифр Цезаря, наверное, был достаточно стойким во времена Красса и Цицерона. Поскольку никакого секретного ключа нет

(он всегда равен 3), пользователи шифра Цезаря должны были считать, что противник тупой и не знает, как его найти, – такое предположение в наши дни выглядит совсем уж нереалистично. (Правда, в 2006 году итальянская полиция арестовала босса мафии, расшифровав сообщения, написанные на клочках бумаги с использованием варианта шифра Цезаря; например, ABC было зашифровано как 456, а не DEF.)

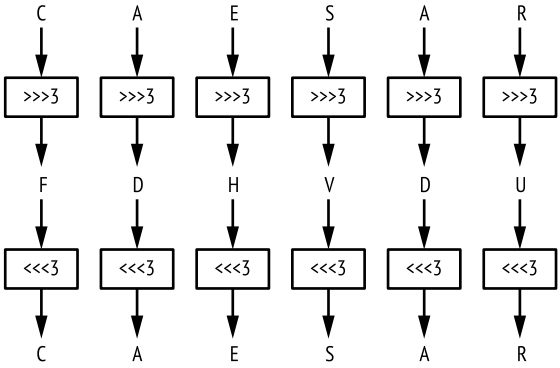

Рис. 1.2. Шифр Цезаря

Можно ли сделать шифр Цезаря более безопасным? Например, можно было сдвигать не на 3, а на какое-нибудь секретное значение, но это не спасало бы положения, потому что противнику достаточно было бы перебрать все 25 возможных значений сдвига и найти то, при котором расшифрованное сообщение становится осмысленным.

Шифр Виженера

Для существенного улучшения шифра Цезаря потребовалось примерно 1500 лет, в XVI веке итальянец Джовани Баттиста Белассо создал шифр Виженера. Имя «Виженер» принадлежит французу Блезу де Виженеру, который изобрел в XVI веке другой шифр, но из-за неправильной атрибуции вошел в историю. Так или иначе, шифр Виженера обрел популярность и использовался, в частности, конфедератами во время Гражданской войны в США и швейцарской армией во время Первой мировой войны.

Шифр Виженера похож на шифр Цезаря, только величина сдвига составляет не три позиции, а определяется *ключом*, набором букв, которым соответствуют числа, равные позиции буквы в алфавите. Например, если ключ равен DUH, то буквы открытого текста сдвигаются на 3, 20 и 7 позиций, потому что *D* отстоит от *A* на три позиции, *U* – на 20 позиций, а *H* – на семь позиций. Последовательность 3, 20, 7 повторяется, пока не будет зашифрован весь открытый текст. Например, слово CRYPTO на ключе DUH было бы зашифровано как FLFSNV: *C* сдвигается на три позиции и превращается в *F*, *R* сдвигается на 20 и превращается в *L* и т. д. На рис. 1.3 показано, как этот принцип применяется к шифрованию предложения THEY DRINK THE TEA.

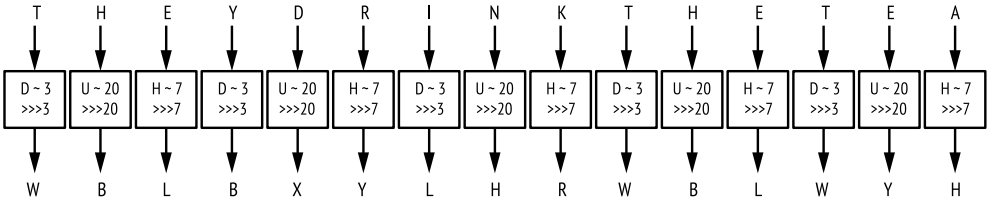

Рис. 1.3. Шифр Виженера

Очевидно, что шифр Виженера безопаснее, чем шифр Цезаря, но все равно его довольно легко взломать. Первый шаг взлома – определить длину ключа. Например, рассмотрим рис. 1.3, где результатом шифрования фразы THEY DRINK THE TEA с ключом DUH является строка WBLBXYLHRWBLWYH (пробелы обычно удаляются, чтобы не раскрывать границы слов). Заметим, что в шифртексте WBLBXYLHRWBLWYH группа из трех букв WBL встречается дважды с интервалом в девять букв. Это позволяет предположить, что было зашифровано одно и то же трехбуквенное слово с одинаковыми величинами сдвига. Поэтому криптоаналитик может сделать вывод, что длина ключа либо равна девяти, либо является делителем девяти (т. е. равна трем). Кроме того, он может догадаться, что повторяющееся трехбуквенное слово – THE, а значит, DUH – возможный ключ шифрования.

Второй шаг взлома шифра Виженера – определение самого ключа методом *частотного анализа*, в котором используется тот факт, что распределение букв в естественных языках неравномерно. Например, в английском чаще всего встречается буква *E*, поэтому, обнаружив, что в шифртексте чаще других встречается *X*, мы можем заключить, что в соответствующей позиции открытого текста, скорее всего, находится буква *E*.

Несмотря на относительную слабость, шифр Виженера, наверное, был достаточно хорош в те времена, когда применялся. Во-первых, для описанной выше атаки нужны сообщения, содержащие хотя бы несколько предложений, поэтому она не будет работать, если шифруются только короткие сообщения. Во-вторых, большая часть сообщений должна была сохранять секретность только в течение короткого промежутка времени, так что даже если в конечном итоге враг их расшифрует, это уже не будет иметь значения. (В XIX веке криптограф Огюст Керкгоффс оценил, что для большинства зашифрованных военных сообщений конфиденциальность требовалась лишь на протяжении трех-четырех часов.)

Как работают шифры

Проанализировав простые шифры, в частности Цезаря и Виженера, мы можем попытаться выделить общие принципы работы шифра и прежде всего два основных компонента: перестановка и режим работы. *Перестановка* – это функция, которая преобразует элемент (в криптографии букву или группу битов) таким образом, что для

каждого элемента однозначно определено обратное преобразование (например, в случае шифра Цезаря это сдвиг на три буквы). *Режим работы* – это алгоритм, который использует перестановку для обработки сообщений произвольного размера. Режим работы шифра Цезаря тривиален – одна и та же перестановка повторяется для каждой буквы. Но в шифре Виженера режим работы сложнее – буквы в разных позициях подвергаются разным перестановкам.

В следующих разделах я более подробно разберу эти компоненты и их связь с безопасностью шифра. Я покажу, почему классические шифры принципиально небезопасны, в отличие от современных шифров, исполняемых высокопроизводительными компьютерами.

Перестановка

В большинстве классических шифров одна буква просто заменяется другой, т. е. мы выполняем *подстановку*. В шифрах Цезаря и Виженера подстановка – это сдвиг в алфавите, хотя сам алфавит или набор символов может меняться: вместо английского языка может быть арабский, а вместо букв – слова, числа или идеограммы. Представление, или кодирование, информации – отдельная тема, слабо связанная с безопасностью. (Мы рассматриваем латинские буквы просто потому, что так использовались классические шифры.)

Подстановка в шифре не может быть произвольной. Это должна быть перестановка букв от *A* до *Z*, при которой для каждой буквы однозначно определена обратная ей. Например, подстановка, преобразующая буквы *A*, *B*, *C*, *D* соответственно в *C*, *A*, *D*, *B*, является перестановкой, поскольку на каждую букву отображается ровно одна буква. Но подстановка, преобразующая *A*, *B*, *C*, *D* в *D*, *A*, *A*, *C*, перестановкой не является, потому что обе буквы *B* и *C* отображаются в *A*, а в случае перестановки у каждой буквы должен быть ровно один прообраз. Кроме того, не каждая перестановка безопасна. Чтобы считаться безопасной, перестановка шифра должна удовлетворять трем условиям.

- **Перестановка должна определяться ключом**, это позволит держать ее в секрете, пока ключ не скомпрометирован. В шифре Виженера, не зная ключа, невозможно сказать, какая из 26 перестановок использовалась, поэтому взломать его не слишком просто.
- **Разным ключам должны соответствовать разные перестановки.** В противном случае будет проще расшифровать сообщение, не зная ключа: если разные ключи порождают одинаковые перестановки, то различных ключей меньше, чем перестановок, и, значит, для дешифрирования без ключа нужно будет перебрать меньше вариантов. В шифре Виженера каждая буква ключа определяет подстановку; всего существует 26 разных букв и столько же различных перестановок.
- **Перестановка должна выглядеть случайной (формальные детали опустим).** После применения перестановки в шифртекс-

те не должно быть никаких закономерностей, поскольку наличие таковых позволит противнику предсказать перестановку, а значит, уровень безопасности снизится. Например, в шифре Виженера подстановка в достаточной мере предсказуема: определив, что *A* преобразуется в *F*, можно было бы заключить, что величина сдвига равна 5, и тогда мы знали бы, что *B* преобразуется в *G*, *C* в *H* и т. д. Но если перестановка выбирается случайно, то, зная, что шифр преобразует *A* в *F*, мы могли бы только сказать, что он *не* преобразует *B* в *F*.

Перестановки, удовлетворяющие этим условиям, мы будем называть *безопасными*. Но далее мы увидим, что безопасность перестановки – необходимое, но еще недостаточное условие для построения безопасного шифра. Необходим еще режим работы шифра, чтобы поддержать сообщения произвольной длины.

Режим работы

Пусть имеется безопасная перестановка, преобразующая, например, *A* в *X*, *B* в *M* и *N* в *L*. Слово BANANA тогда будет преобразовано в MXLXLX, в котором каждая буква *A* заменена на *X*. Таким образом, использование одной и той же перестановки для всех букв открытого текста показывает, какие буквы повторяются. Путем анализа повторений вы, возможно, не раскроете все сообщение целиком, но *что-то* о нем узнаете. В примере слова BANANA не нужно знать ключ, чтобы догадаться, что в тех позициях, где в зашифрованном тексте мы видим *X*, в открытом тексте находятся одинаковые буквы. И то же самое относится к двум позициям, занятым буквой *L*. Так что если заранее известно, к примеру, что сообщение содержит название фрукта, то можно быть уверенным, что это не CHERRY, LYCHEE или другой фрукт из шести букв, а, скорее всего, BANANA.

Режим работы (или просто *режим*) шифра уменьшает шансы на раскрытие повторяющихся букв в открытом тексте, потому что для них используются разные перестановки. Режим шифра Виженера решает эту проблему частично: если ключ состоит из *N* букв, то для каждой из *N* последовательных букв будет использована своя перестановка. Но это все же не исключает появления закономерностей в шифртексте, потому что для каждой *N*-й буквы сообщения используется одна и та же перестановка. Именно поэтому шифр Виженера можно взломать с помощью частотного анализа.

В шифре Виженера от частотного анализа можно защититься, если шифровать только открытые тексты, длина которых совпадает с длиной ключа. Но все равно остается другая проблема: при многократном использовании одного и того же ключа не скрыть сходство открытых текстов. Например, если ключ равен KYN, то слова TIE и PIE преобразуются в DGR и ZGR соответственно. Оба шифртекста заканчиваются двумя одинаковыми буквами (*GR*), это говорит о том, что и в обоих открытых текстах две последние буквы совпадают. Безопасный шифр не

должен раскрывать таких закономерностей. Для построения безопасного шифра нужно сочетать безопасную перестановку с безопасным режимом. В идеале их комбинация не должна позволить противнику узнать о сообщении что-то, кроме его длины.

Почему классические шифры небезопасны

Классические шифры обречены оставаться небезопасными, потому что мы ограничены операциями, которые можно выполнить в уме или на бумаге. Им недостает вычислительной мощности компьютера, поэтому они легко взламываются с помощью простых программ. Обсудим фундаментальную причину, по которой в современном мире простота делает такие шифры небезопасными.

Напомним, что перестановка шифра должна выглядеть случайной, чтобы считаться безопасной. Конечно, лучший способ выглядеть случайной – это *быть* случайной, т. е. мы должны случайным образом выбирать каждую перестановку из множества всех возможных. А выбирать есть из чего. Для английского алфавита из 26 букв имеется приблизительно 2^{88} перестановок:

$$26! = 403291461126605635584000000 \approx 2^{88}.$$

Здесь восклицательный знак (!) – символ факториала, определяемого следующим образом:

$$n! = n \times (n - 1) \times (n - 2) \times \ldots \times 3 \times 2.$$

(Чтобы понять, откуда берется это число, будем подсчитывать перестановки как упорядоченные списки букв: существует 26 способов выбрать первую букву, после чего вторую букву можно выбрать 25 способами, третью – 24 способами и т. д.) Это число поистине огромно; по порядку величины оно такое же, как число атомов в теле человека. Но для классических шифров доступна лишь малая доля этого множества перестановок, а именно те, для вычисления которых достаточно простых операций (например, сдвигов) и которые имеют простое описание (например, простой алгоритм или небольшую таблицу преобразования). Проблема в том, что перестановка, удовлетворяющая обоим ограничениям, не может быть безопасной.

Безопасные перестановки можно получить с помощью простых операций, выбрав случайную перестановку из множества, представленного в виде таблицы, содержащей 25 букв (достаточно, чтобы представить перестановку 26 букв, из которых двадцать шестая отсутствует), а для ее применения нужно будет найти буквы в этой таблице. Но тогда описание не будет коротким. Например, чтобы описать 10 разных перестановок, потребуется 250 букв, а не 10, как в шифре Виженера.

Можно также получить безопасные перестановки с коротким описанием, если не ограничиваться сдвигами по алфавиту, а добавить

более сложные операции, например сложение, умножение и т. д. Так работают современные шифры: имея ключ длиной 128 или 256 бит, они выполняют сотни поразрядных операций для шифрования одной-единственной буквы. На компьютере, выполняющем миллиарды поразрядных операций в секунду, это не занимает много времени, но на вычисления вручную пришлось бы потратить несколько часов, и все равно результат был бы уязвим для частотного анализа.

Идеальное шифрование: одноразовый блокнот

В принципе, классический шифр не может быть безопасным, если только не используется очень длинный ключ, но шифровать гигантским ключом практически неудобно. Однако такой шифр существует, он называется одноразовым блокнотом и является самым безопасным. По существу, он гарантирует *идеальную секретность*: даже располагая неограниченной вычислительной мощностью, противник не сможет узнать об открытом тексте ничего, кроме длины.

В следующих разделах я покажу, как работает одноразовый блокнот, а затем приведу набросок доказательства его безопасности.

Шифрование с помощью одноразового блокнота

Одноразовый блокнот принимает открытый текст P и случайный ключ K такой же длины, как P, и порождает шифртекст C, вычисляемый по формуле

$$C = P \oplus K,$$

где C, P, K – битовые строки одинаковой длины, а $\oplus$ – операция поразрядного исключающего ИЛИ (XOR), определенная следующим образом: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$.

Примечание *Я описываю одноразовый блокнот в стандартной форме, применяемой к битам, но его можно адаптировать и для других символов. Например, в случае букв мы получим вариант шифра Цезаря, в котором величина сдвига каждой буквы выбирается случайным образом.*

Дешифрирование с одноразовым блокнотом выполняется точно так же, как шифрование; это просто операция XOR: $P = C \oplus K$. Действительно, легко проверить, что $C \oplus K = P \oplus K \oplus K = P$, поскольку $K \oplus K$ дает строку, состоящую из одних нулей 000 … 000. Вот так – даже проще, чем шифр Цезаря.

Например, если P = 01101101 и K = 10110100, то вычисление выглядит следующим образом:

$$C = P \oplus K = 01101101 \oplus 10110100 = 11011001.$$

Дешифрирование дает P в результате такого вычисления:

$$P = C \oplus K = 11011001 \oplus 10110100 = 01101101.$$

Важно отметить, что одноразовый блокнот можно использовать только *один раз*: каждый ключ K следует использовать лишь однократно. Если бы один и тот же K использовался для зашифровывания P_1 и P_2 в C_1 и C_2, то пассивный противник мог бы подслушать и вычислить выражение:

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2 \oplus K \oplus K = P_1 \oplus P_2.$$

Тогда пассивный противник узнал бы результат применения XOR к P_1 и P_2, хотя эта информация должна держаться в секрете. Более того, если хотя бы одно открытое сообщение известно, то можно восстановить и другое.

Разумеется, одноразовый блокнот крайне неудобно использовать, потому что требуется ключ такой же длины, как простой текст, да еще для каждого нового сообщения или группы данных необходимо выбирать новый случайный ключ. Чтобы зашифровать терабайтный жесткий диск, понадобился бы другой диск такого же объема для хранения ключа! И тем не менее одноразовые блокноты использовались на практике. Например, их применяло Британское управление специальных операций во время Второй мировой войны, сотрудники КГБ и АНБ, да и сейчас они используются в особых ситуациях. (Я слышал историю о швейцарских банкирах, которые не смогли договориться о шифре, которому доверяли бы обе стороны, и потому остановились на одноразовых блокнотах, но поступать так не рекомендую.)

Почему одноразовый блокнот безопасен?

Хотя одноразовый блокнот непрактичен, важно понимать, почему именно он безопасен. В 1940-х годах американский математик Клод Шеннон доказал, что для достижения идеальной секретности ключ одноразового блокнота должен быть не короче сообщения. Идея доказательства довольно проста. Предположим, что противник обладает неограниченными возможностями и, стало быть, может перепробовать все ключи. Наша цель – зашифровать сообщение так, чтобы противник, располагающий шифртекстом, не мог исключить ни одного возможного открытого текста.

Интуитивно свойство одноразового блокнота, обеспечивающее идеальную секретность, можно описать так: если ключ K случайный, то результирующее сообщение C представляется противнику таким же случайным, как K, потому что применение XOR к случайной строке и произвольной фиксированной строке дает случайную строку. Чтобы убедиться в этом, рассмотрим вероятность получения 0 в первом бите случайной строки (она равна 1/2). Какова вероятность, что результат применения XOR к случайному биту и второму биту будет равен 0?

Правильно, снова 1/2. То же рассуждение можно распространить на битовые строки произвольной длины. Таким образом, шифртекст *C* кажется случайным противнику, не знающему *K*, поэтому действительно невозможно узнать что-то о *P* по *C*, даже при наличии у противника неограниченного времени и вычислительной мощности. Иными словами, знание шифртекста не дает никакой информации об открытом тексте, кроме его длины, а это и есть определение безопасного шифра.

Например, если длина шифртекста равна 128 бит (а это значит, что и длина открытого текста составляет 128 бит), то количество возможных шифртекстов равно 2^{128}, поэтому, с точки зрения противника, количество возможных открытых текстов тоже должно быть равно 2^{128}. Но если возможных ключей меньше, чем 2^{128}, то противник сможет исключить некоторые открытые тексты. Например, если длина ключа равна всего 64 бита, то противник сможет определить 2^{64} возможных открытых текстов и исключить подавляющее большинство 128-битовых строк. Противник не узнает, чему равен открытый текст, но узнает, чему он точно не равен, а уже этого достаточно, чтобы секретность шифрования была неидеальной.

Как видим, для обеспечения идеальной безопасности ключ должен быть не короче открытого текста, но в реальной жизни это требование быстро становится непрактичным. Далее мы обсудим современные подходы к шифрованию, позволяющие обеспечить оптимальную безопасность практически осуществимым способом.

Вероятность в криптографии

Вероятность количественно выражает шансы возникновения некоторого события. Это число от 0 до 1, причем 0 означает «никогда не произойдет», а 1 – «обязательно произойдет». Чем выше вероятность, тем больше шансов. В литературе можно найти различные объяснения вероятности, обычно речь идет о ящике с красными и белыми шарами и вероятности вытащить шар определенного цвета.

В криптографии вероятность часто используется для измерения шансов противника на успех. Для этого подсчитывается 1) число успешных событий (например, «найден единственно правильный секретный ключ») и 2) общее число возможных событий (например, общее число n-битовых ключей равно 2^n). В данном случае вероятность, что случайно выбранный ключ окажется правильным, равна $1/2^n$, т. е. отношению числа успешных событий (единственный секретный ключ) к общему числу событий (2^n возможных ключей). Число $1/2^n$ пренебрежимо мало для ключей типичной длины (128 и 256).

Вероятность того, что событие не *произойдет*, равна $1 - p$, где p – вероятность события. В примере выше вероятность получить неправильный ключ равна $1 - 1/2^n$, она очень близка к 1, можно считать, что почти наверняка так и будет.

Безопасность шифрования

Мы видели, что классические шифры небезопасны, а идеально безопасный шифр типа одноразового блокнота практически неудобен. Таким образом, мы должны несколько ослабить требования к безопасности, если хотим получить безопасные *и* практически применимые шифры. Но что вообще означает «безопасный», помимо очевидного и неформального «пассивный противник не может дешифрировать безопасные сообщения»?

Интуитивно представляется очевидным, что шифр безопасен, если даже при наличии большого числа пар «открытый текст – шифртекст» о его поведении *ничего нельзя узнать* при применении к другим открытым или шифртекстам. Это ставит новые вопросы.

- Как противник может заполучить такие пары? Насколько велико «большое число»? Все это определяется *моделями атак* – предположениями о том, что может и чего не может сделать противник.
- Что можно «узнать», и что такое «поведение шифра»? Это определяется *целями безопасности* (security goal) – описаниями того, что считать успешной атакой.

Модели атаки и цели безопасности неразделимы; нельзя заявить, что система безопасна, не объяснив, от кого и от чего ее защищают. Поэтому вводится понятие *аспекта безопасности* (security notion) – комбинации цели безопасности и модели атаки. Говорят, что шифр *реализует* некоторый аспект безопасности, если противник, работающий в рамках данной модели, не может достичь поставленной цели безопасности.

Модели атак

Модель атаки – это набор предположений о том, как противник мог бы взаимодействовать с шифром и что он может, а чего не может делать. Модель атаки должна:

- формулировать требования к криптографам, проектирующим шифры, чтобы они знали, что представляет собой противник, и от каких атак следует защищаться;
- давать пользователям указания о том, безопасно ли использовать шифр в конкретной ситуации;
- давать информацию криптоаналитикам, пытающимся взломать шифры, чтобы они понимали, допустима ли данная атака. Атака считается допустимой, только если она выполнима в контексте рассматриваемой модели.

Модели атак не обязаны в точности соответствовать реальности, это всего лишь аппроксимация. Статистик Джордж Э. П. Бокс писал: «Все модели неверны, но некоторые полезны». В криптографии, чтобы считаться полезной, модель атаки должна как минимум описы-

вать, какие действия доступны противнику для атаки на шифр. Нет ничего плохого в том, чтобы переоценить возможности противника, поскольку это помогает предвидеть будущие приемы атак – успеха добиваются лишь криптографы, страдающие паранойей. Плохая модель недооценивает противника и вселяет ложную уверенность в стойкости шифра, который кажется безопасным теоретически, но практически таковым не является.

Принцип Керкгоффса

Одно из предположений, присущее всем моделям, называется принципом Керкгоффса и утверждает, что безопасность шифра должна опираться только на секретность ключа, но не на секретность алгоритма. Сегодня, когда шифры и протоколы открыто публикуются и могут использоваться всеми желающими, это положение может показаться очевидным. Но в свое время голландский лингвист Огюст Керкгоффс говорил о военных шифровальных машинах, которые специально проектировались для некоторой армии или дивизии. В книге «Военная криптография», вышедшей в 1883 году, где были сформулированы шесть требований к военной криптографической системе, он писал: «Система не должна требовать секретности и при попадании в руки врага не должна терять надежности».

Модели черного ящика

Теперь рассмотрим некоторые полезные модели атак, формулируемые в терминах того, что противник может наблюдать и какие запросы он может предъявлять к шифру. *Запросом* в этом контексте называется операция, которая передает входное значение некоторой функции и получает в ответ результат; при этом детали работы функции не раскрываются.

Например, *запрос шифрования* принимает открытый текст и возвращает соответствующий шифртекст, не раскрывая секретного ключа.

Такие модели называются *моделями черного ящика*, потому что противник видит только данные на входе и выходе шифра. Например, некоторые микросхемы на смарт-картах безопасно защищают не только ключи шифра, но и его внутреннее устройство, но никто не мешает подключиться к микросхеме и попросить ее дешифрировать любой шифртекст. В результате противник получит соответствующий открытый текст, и это может помочь ему в определении ключа. Это реальный пример *запроса дешифрирования*.

Существует несколько разных моделей черного ящика. Я перечислю их в порядке расширения возможностей противника.

- *Атаки на основе шифртекста* (ciphertext-only attack – COA). Противник может наблюдать шифртекст, но не знает соответствующего ему открытого текста. Он также не знает, как выбирались открытые тексты. В модели COA противник пассивен и не может предъявлять запросов шифрования или дешифрирования.

- *Атаки с известным открытым текстом* (known-plaintext attack – KPA). Противник наблюдает шифртекст и знает соответствующий ему открытый текст. Таким образом, в модели KPA противник может получить список пар «открытый текст – шифртекст», но предполагается, что открытые тексты выбирались случайным образом. В этой модели KPA противник также пассивен.
- *Атаки с подобранным открытым текстом* (chosen-plaintext attack – CPA). Противник может предъявлять запросы шифрования по своему выбору и наблюдать результирующие шифртексты. Эта модель отражает ситуации, в которых противник имеет возможность задавать подлежащий шифрованию открытый текст полностью или частично и наблюдать соответствующий шифртекст. В отличие от пассивных моделей COA и KPA, в модели CPA противник *активен*, т. е. может оказывать воздействие на процессы шифрования, а не только пассивно подслушивать.
- *Атаки с подобранным шифртекстом* (chosen-ciphertext attack – CCA). Противник может предъявлять запросы шифрования и дешифрирования. На первый взгляд модель CCA может показаться абсурдной – раз мы можем дешифрировать, то что еще надо? Но, как и модель CPA, она представляет ситуации, когда противник может в какой-то мере повлиять на шифртекст, а впоследствии получить доступ к открытому тексту. Кроме того, возможность дешифрировать что-то не всегда означает вскрытие системы. Например, некоторые устройства защиты видео позволяют противнику предъявлять запросы шифрования и дешифрирования к микросхеме, но в этом контексте противник стремится получить ключ, что позволит ему самостоятельно распространять видео; способности «бесплатно» дешифрировать недостаточно для взлома системы.

В описанных выше моделях наблюдение и запросы шифртекстов обходятся не бесплатно. Каждый шифртекст является результатом вычисления функции шифрования. Это означает, что для генерирования 2^N пар «открытый текст – шифртекст» путем предъявления запросов шифрования требуется примерно столько же вычислений, сколько для проверки 2^N ключей. При оценивании стоимости атаки следует принимать во внимание стоимость выполнения запросов.

Модели серого ящика

В *модели серого ящика* противник имеет доступ к *реализации* шифра. Поэтому модели серого ящика более реалистичны, чем модели черного ящика в таких приложениях, как смарт-карты, встраиваемые и виртуализированные системы, к которым противник часто имеет физический доступ и потому может манипулировать внутренними аспектами алгоритмов. По той же причине модели серого ящика труднее определить, потому что они зависят от физических, аналоговых свойств, а не только от входов и выходов алгоритма, а теоретическая

криптография зачастую не способна абстрагировать всю сложность реального мира.

Одним из видов атак в рамках моделей серого ящика являются *атаки по побочным каналам*. Побочный канал – это источник информации, зависящий от реализации шифра, он может быть как программным, так и аппаратным. Противник, пользующийся побочным каналом, наблюдает или измеряет аналоговые характеристики реализации шифра, не нарушая его целостность; это *неразрушающие* атаки. Типичными примерами программных побочных каналов являются время выполнения и поведение системы, объемлющей шифр, например сообщения об ошибках, возвращаемые значения, ветвления и т. д. А для реализаций на смарт-картах типичная атака по побочному каналу включает измерение энергопотребления, электромагнитного излучения или акустического шума.

Разрушающие атаки на реализации шифров более эффективны, чем атаки по побочному каналу, но и стоят дороже, потому что требуют специального оборудования. Если для простой атаки по побочному каналу достаточно стандартного ПК и осциллографа, то для разрушающей атаки может потребоваться микроскоп с высокой разрешающей способностью и химическая лаборатория. Разрушающая атака включает целый набор приемов и процедур, от использования азотной кислоты для растворения корпуса микросхемы до получения микроскопических снимков, частичной обратной разработки и, возможно, модификации поведения микросхемы, например путем внесения неисправностей с помощью лазера.

Цели безопасности

Я неформально определил цель безопасности как «невозможность что-то узнать о поведении шифра». Чтобы превратить эту идею в строгое математическое определение, криптографы определяют две цели безопасности, соответствующие разным представлениям о том, что понимается под знанием о поведении шифра.

Неразличимость (indistinguishability – IND). Шифртексты должны быть неотличимы от случайных строк. Обычно это иллюстрируют на примере гипотетической игры: если противник выбирает два открытых текста, а затем получает шифртекст, соответствующий одному из них (выбранному наугад), то он не должен иметь возможность определить, какой открытый текст был зашифрован, даже если выполнит запросы шифрования для обоих открытых текстов (и запросы дешифрирования, если используется модель CCA, а не CPA).

Неподатливость (non-malleability – NM). Для заданного шифртекста $C_1 = \mathbf{E}(K, P_1)$ должно быть невозможно создать другой шифртекст C_2 такой, что соответствующий ему открытый текст P_2 каким-то осмысленным образом связан с P_1 (например, создать P_2, равный $P_1 \oplus 1$ или $P_1 \oplus X$ для некоторого известного значения X). Как ни

странно, одноразовый блокнот является податливым: если задан шифртекст $C_1 = P_1 \oplus K$, то можно определить $C_2 = C_1 \oplus 1$, являющийся допустимым шифртекстом для открытого текста $P_2 = P_1 \oplus 1$ с тем же ключом K. Вот тебе и идеальный шифр.

Далее мы обсудим эти цели безопасности в контексте различных моделей атак.

Аспекты безопасности

Цели безопасности полезны только в сочетании с моделью атаки. По соглашению, аспект безопасности записывают в виде *ЦЕЛЬ–МОДЕЛЬ*. Например, IND-CPA означает неразличимость для атак с подобранным открытым текстом, NM-CCA – неподатливость для атак с подобранным шифртекстом и т. д. Начнем с целей безопасности, представляющих интерес для противника.

Семантическая безопасность и рандомизированное шифрование: IND-CPA

Самым важным аспектом безопасности является IND-CPA, у него даже имеется специальное название *семантическая безопасность*. Он отражает интуитивную идею о том, что шифртекст не должен раскрывать никакой информации об открытом тексте, при условии что ключ секретный. Для достижения безопасности в смысле IND-CPA алгоритм шифрования должен возвращать разные шифртексты при неоднократном вызове для одного и того же открытого текста, иначе противник мог бы обнаружить повторяющиеся открытые тексты по их шифртекстам, что противоречит требованию о нераскрытии информации шифртекстом.

Один из способов добиться безопасности в смысле IND-CPA – воспользоваться *рандомизированным шифрованием*. Как следует из названия, в этом случае процесс шифрования рандомизируется и возвращает различные шифртексты при повторном шифровании одного и того же открытого текста. Тогда шифрование можно описать в виде $C = \mathbf{E}(K, R, P)$, где R – случайные биты, каждый раз выбираемые заново. Но дешифрирование остается детерминированным, потому что при заданном значении $\mathbf{D}(K, R, C)$ неизменно должно возвращать P, каким бы ни было значение R.

А что, если шифрование не рандомизировано? В игре IND, описанной в разделе «Цели безопасности» выше, противник выбирает два открытых текста, P_1 и P_2, и получает шифртекст для одного из них, но не знает, какого именно. То есть противник получает $C_i = \mathbf{E}(K, P_i)$ и должен угадать, чему равно i: 1 или 2. В модели CPA противник может выполнить запросы шифрования и найти $C_1 = \mathbf{E}(K, P_1)$ и $C_2 = \mathbf{E}(K, P_2)$. Если шифрование не рандомизировано, то достаточно посмотреть, чему равно $C_i - C_1$ или C_2, чтобы понять, какой открытый текст был зашифрован, и, стало быть, выиграть. Поэтому для аспекта безопасности IND-CPA рандомизация является неотъемлемой частью.

Примечание *При рандомизированном шифровании шифртекст может быть немного длиннее открытого текста, чтобы одному и тому же открытому тексту могло соответствовать несколько шифртекстов. Например, если одному открытому тексту может соответствовать 2^{64} шифртекстов, то шифртекст должен быть длиннее открытого по крайней мере на 64 бита.*

Достижение семантически безопасного шифрования

В одном из простейших построений семантически безопасного шифра используется *детерминированный генератор псевдослучайных битов* (deterministic random bit generator – DRBG) – алгоритм, который получает секретное значение и возвращает биты, выглядящие случайными:

$$\mathbf{E}(K, R, P) = (\mathbf{DRBG}(K||R) \oplus P, R).$$

Здесь строка R случайно выбирается для каждого нового акта шифрования и передается DRBG вместе с ключом ($K||R$ обозначает строку, полученную дописыванием R в конец K). Это напоминает одноразовый блокнот, только вместо выбора случайного ключа такой же длины, как у сообщения, мы пользуемся генератором псевдослучайных битов, чтобы получить строку, которая только выглядит как случайная.

Доказать, что этот шифр безопасен в смысле IND-CPA, легко, если предположить, что DRBG порождает случайные биты. Проведем доказательство от противного. Если мы можем различить шифртексты, полученные для разных случайных строк, т. е. можем отличить $\mathbf{DRBG}(K||R) \oplus P$ от случайного значения, значит, можем отличить и $\mathbf{DRBG}(K||R)$ от случайного значения. Напомним, что модель CPA позволяет получать шифртексты, соответствующие заданным открытым текстам P, поэтому мы можем применить XOR к P и $\mathbf{DRBG}(K||R) \oplus P$ и получить $\mathbf{DRBG}(K||R)$. Но это противоречит исходному предположению о том, что $\mathbf{DRBG}(K||R)$ нельзя отличить от случайного значения, т. е. порождаемые строки случайны. Итак, мы приходим к выводу, что шифртексты невозможно отличить от случайных строк, поэтому шифр безопасен.

Примечание *В качестве упражнения попробуйте установить, каким аспектам безопасности удовлетворяет шифр* $\mathbf{E}(K, R, P) = (\mathbf{DRBG}(K||R) \oplus P, R)$. *Безопасен ли он в смысле NM-CPA? А в смысле IND-CCA? Ответ будет дан в следующем разделе.*

Сравнение аспектов безопасности

Итак, мы узнали, что модели атак, например CPA и CCA, комбинируются с целями безопасности, допустим NM или IND, для построения аспектов безопасности NM-CPA, NM-CCA, IND-CPA и IND-CCA. Как соотносятся эти аспекты между собой? Можно ли доказать, что шифр, удовлетворяющий аспекту X, удовлетворяет также аспекту Y?

Некоторые соотношения очевидны: из IND-CCA следует IND-CPA, а из NM-CCA следует NM-CPA, потому что все действия, доступные CPA-противнику, доступны также CCA-противнику. То есть если невозможно взломать шифр, предъявляя запросы с подобранным шифртекстом и подобранным открытым текстом, то его невозможно взломать и с помощью одних лишь запросов с подобранным открытым текстом.

Не столь очевиден тот факт, что из IND-CPA не следует NM-CPA. Чтобы понять, почему это так, заметим, что показанное выше построение шифра IND-CPA (**DRBG**(K, R) $\oplus$ P, R) не обладает свойством NM-CPA: зная шифртекст (X, R), мы можем создать шифртекст ($X \oplus 1$, R), соответствующий $P \oplus 1$, что противоречит цели неподатливости.

Однако противоположное соотношение имеет место: из NM-CPA следует IND-CPA. Интуиция подсказывает, что система типа IND-CPA аналогична складыванию предметов в мешок: увидеть их нельзя, но можно перемешать, встряхнув мешок. NM-CPA больше похож на сейф: после того как предмет туда помещен, с ним уже никак нельзя взаимодействовать. Однако эта аналогия не работает для IND-CCA и NM-CCA, эквивалентным в том смысле, что из наличия одного аспекта следует наличие другого. Чисто техническое доказательство я опускаю.

Два типа приложений шифрования

Существует два основных типа приложений шифрования. *Транзитное шифрование* защищает данные, передаваемые с одной машины на другую: данные шифруются перед отправкой и дешифрируются после получения, как в случае зашифрованного соединения с сайтом электронной коммерции. *Стационарное шифрование* защищает данные, хранящиеся в информационной системе. Данные шифруются перед записью в память и дешифрируются перед чтением. Примерами могут служить системы шифрования диска на ноутбуках, а также шифрование виртуальных машин в облачных виртуальных экземплярах. Рассмотренные выше аспекты безопасности применимы к приложениям обоих типов, но какой аспект подходит лучше, может зависеть от приложения.

Асимметричное шифрование

До сих пор мы рассматривали только симметричное шифрование, когда обе стороны пользуются общим ключом. В случае *асимметричного шифрования* имеется два ключа: один для шифрования, другой для дешифрирования. Ключ шифрования называется *открытым*, и обычно считается, что он доступен любому, кто хочет отправить вам зашифрованное сообщение. Но ключ дешифрирования должен храниться в секрете, поэтому называется *закрытым*.

Открытый ключ можно вычислить по закрытому, но очевидно, что обратное невозможно. Иными словами, вычисление в одном направлении простое, а в другом трудное, в этом и заключается идея *криптографии с открытым ключом* – для функции, легко вычисляемой в одном направлении, практически невозможно вычислить обратную.

Модели атак и цели безопасности для асимметричного шифрования почти такие же, как для симметричного, но поскольку ключ шифрования открыт, любой противник может предъявлять запросы шифрования, используя этот ключ. Поэтому по умолчанию для асимметричного шифрования подразумевается модель атаки с подобранным открытым текстом (CPA).

Симметричное и асимметричное шифрования – два основных типа шифрования, именно они обычно применяются для построения безопасных систем связи. Они же используются как основа для более сложных схем, с которыми мы познакомимся ниже.

Дополнительные функции шифров

Базовый алгоритм шифрования преобразует открытые тексты в шифртексты и наоборот, не предъявляя никаких требований к безопасности. Но некоторым приложениям этого недостаточно, могут потребоваться дополнительные средства безопасности или дополнительная функциональность. Поэтому криптографы создали различные варианты симметричного и асимметричного шифрований. Одни из них хорошо известны, эффективны и широко распространены, другие пока являются экспериментальными, с ними трудно работать, а производительность не слишком высока.

Шифрование с аутентификацией

Шифрование с аутентификацией (authenticated encryption – AE) – вариант симметричного шифрования, при котором, помимо шифртекста, возвращается *аутентификационный жетон* (authentication tag). На рис. 1.4 показана схема шифрования с аутентификацией $\mathbf{AE}(K, P) = (C, T)$, где аутентификационный жетон T – короткая строка, которую невозможно угадать, не зная ключа. Алгоритм дешифрирования принимает K, C и T и возвращает открытый текст P, только если T является допустимым жетоном для этой пары «открытый текст – шифртекст»; в противном случае возвращается код ошибки.

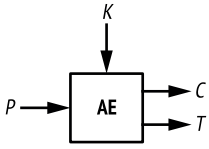

Рис. 1.4. Шифрование с аутентификацией

Жетон гарантирует *целостность* сообщения и служит доказательством того, что полученный шифртекст идентичен первоначально

отправленному правомочной стороной, знающей ключ *K*. Если *K* известен еще только одной стороне, то жетон гарантирует также, что сообщение было отправлено именно ей, т. е. он неявно *аутентифицирует* ожидаемого отправителя как фактического создателя сообщения.

Примечание *Я употребляю термин «создатель», а не «отправитель», потому что пассивный противник может подслушать и запомнить некоторые пары (C, T), отправленные стороной A стороне B, а затем снова отправить их B, притворившись A. Такую атаку повторным воспроизведением можно предотвратить, например, включив в сообщение счетчик. В момент дешифрирования сообщения его счетчик i увеличивается на 1. Таким образом, можно проверить, было ли сообщение отправлено дважды, и если да, то имеет место атака повторным воспроизведением. Одновременно это позволяет обнаружить потерянные сообщения.*

Шифрование с аутентификацией и ассоциированными данными (authenticated encryption with associated data – AEAD) – обобщение шифрования с аутентификацией, при котором некоторый открытый текст и незашифрованные данные используются для генерирования аутентификационного жетона **AEAD**(*K*, *P*, *A*) = (*C*, *T*). Типичное применение AEAD – защита дейтаграмм протоколов с открытым заголовком и зашифрованной полезной нагрузкой. В таких случаях по крайней мере какие-то данные заголовка должны оставаться открытыми; например, адреса получателей должны быть открыты, чтобы пакеты можно было маршрутизировать.

Дополнительные сведения о шифровании с аутентификацией см. в главе 8.

Шифрование с сохранением формата

Базовый шифр принимает и возвращает биты; ему безразлично, представляют ли биты текст, изображение или PDF-документ. Шифртекст тоже можно представить в виде неформатированных байтов, шестнадцатеричных символов, в кодировке base64 и в других форматах. Но что, если шифртекст должен иметь такой же формат, как открытый текст? Например, подобное требование предъявляется системами баз данных, которые могут хранить данные только в предписанном формате.

Эту задачу решает *шифрование с сохранением формата* (format-preserving encryption – FPE). Подобные алгоритмы создают шифртекст, имеющий такой же формат, как открытый текст. Например, FPE можно применить для шифрования IP-адресов (как показано на рис. 1.5), почтовых индексов, номеров кредитных карт с правильной контрольной суммой и т. д.

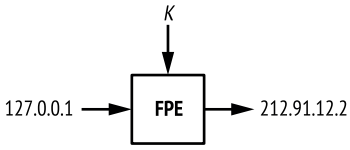

Рис. 1.5. Шифрование IP-адресов с сохранением формата

Полностью гомоморфное шифрование

Полностью гомоморфное шифрование (fully homomorphic encryption – FHE) – заветная мечта криптографа: оно позволяет пользователю заменить шифртекст $C = \mathbf{E}(K, P)$ другим шифртекстом $C' = \mathbf{E}(K, \mathbf{F}(P))$, где $\mathbf{F}(P)$ может быть произвольной функцией от P, не дешифрируя исходный шифртекст C. Например, P может быть текстовым документом, а **F** – модификацией части текста. Представьте себе облачное приложение, в котором хранятся ваши зашифрованные данные, но при этом облачный провайдер не знает ни характер данных, ни тип примененных к ним изменений. Звучит заманчиво, не правда ли?

Но у этой медали есть и обратная сторона: шифрование такого типа медленное – настолько медленное, что даже самые простые операции заняли бы недопустимо много времени. Первая схема FHE была придумана в 2009 году, с тех пор появились более эффективные варианты, но по-прежнему не ясно, сможет ли FHE когда-нибудь достичь быстроты, при которой станет полезным?

Шифрование, допускающее поиск

Такое шифрование позволяет искать по зашифрованной базе данных, не допуская утечки поисковых термов; с этой целью шифруется сам поисковый запрос. Как и полностью гомоморфное шифрование, шифрование с возможностью поиска могло бы повысить уровень конфиденциальности многих облачных приложений, скрыв поисковые запросы от облачного провайдера. Существуют коммерческие решения, рекламирующие возможность шифрования, допускающего поиск, но в большинстве случаев они основаны на стандартной криптографии, дополненной несколькими трюками, частично решающими проблему поиска. На момент написания этой книги шифрование, допускающее поиск, в исследовательском сообществе считается экспериментальной функцией.

Настраиваемое шифрование

Настраиваемое шифрование (tweakable encryption – TE) похоже на базовое, но в нем есть дополнительный настроечный параметр (tweak), призванный эмулировать различные версии шифра (см. рис. 1.6). Настроечный параметр может быть уникальным для каждого пользователя, это гарантирует, что пользовательский шифр не может быть клонирован другими сторонами, использующими тот же продукт. Основное применение TE – *шифрование диска*. Однако TE не привязано

к какому-то одному приложению, а является низкоуровневым типом шифрования и используется для построения других схем, например режимов шифрования в процессе аутентификации.

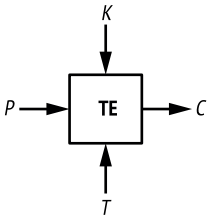

Рис. 1.6. Настраиваемое шифрование

При настраиваемом шифровании диска шифруется содержимое запоминающих устройств, например жестких или SSD-дисков. (Применять здесь рандомизированное шифрование нельзя, потому что при этом увеличился бы размер данных, что для файлов на носителе недопустимо.) Чтобы шифрование было непредсказуемым, используется настроечный параметр, зависящий от положения шифруемых данных, обычно номер сектора или индекс блока.

Какие возможны проблемы

Алгоритмы шифрования или их реализации могут не обеспечивать конфиденциальность по многим причинам. Например, могут не удовлетворяться требования к безопасности («система должна быть безопасной в смысле IND-CPA»), или сформулированные требования могут не отвечать реальности (скажем, в технических условиях прописана только безопасность в смысле IND-CPA, тогда как противник может предъявлять запросы с подобранным шифртекстом). Увы, многие инженеры даже не задумываются о требованиях к криптографической безопасности, а просто хотят, чтобы система была «безопасной», не понимая, что это означает. Это прямой путь к катастрофе. Рассмотрим два примера.

Слабый шифр

Наш первый пример касается шифров, которые можно атаковать методами криптоанализа, как случилось со стандартом мобильной связи 2G. В мобильных телефонах 2G использовался шифр A5/1, оказавшийся слабее, чем ожидалось, поэтому любой человек, обладающий соответствующими навыками и инструментами, мог перехватывать вызовы. Операторы связи вынуждены были искать обходные пути для предотвращения атак.

Примечание *В стандарте 2G был определен также шифр A5/2 для всех регионов, кроме ЕС и США. Он преднамеренно был сделан более слабым, чтобы помешать повсеместному применению стойкого шифрования.*

Заметим, впрочем, что атака на шифр A5/1 нетривиальна, исследователям потребовалось более 10 лет для открытия эффективного криптоаналитического метода. Кроме того, для атаки требуется *компромисс между временем и памятью* (time-memory trade-off – TMTO), т. е. сначала нужно потратить несколько дней или недель на вычисление очень больших справочных таблиц, которые затем применяются для проведения самой атаки. В случае A5/1 заранее вычисленные таблицы занимают более 1 ТБ. В последующих стандартах мобильного шифрования, в частности 3G и LTE, определены более стойкие шифры, но это не значит, что система в целом не может быть скомпрометирована; просто шифрование нельзя скомпрометировать, взломав симметричный шифр, являющийся частью системы.

Неправильная модель

В следующем примере рассматривается неправильная модель атаки, не учитывающая некоторых побочных каналов.

Во многих коммуникационных протоколах с шифрованием гарантируется, что используются шифры, которые считаются безопасными в модели CPA или CCA. Однако для некоторых атак не нужны запросы шифрования, как в модели CPA, или не используются запросы дешифрирования, как в модели CCA. Требуются только *запросы допустимости* – является ли данный шифртекст допустимым, и эти запросы обычно отправляются системе, отвечающей за дешифрирование шифртекстов. Примером могут служить *атаки на оракул дополнения* (padding oracle attack), когда противник узнает, правильно ли отформатирован шифртекст.

Точнее, в случае атаки на оракул дополнения шифртекст является допустимым, только если соответствующий ему открытый текст правильно *дополнен* последовательностью байтов с целью упростить шифрование. Если дополнение неправильно, то дешифрирование завершится ошибкой, а противник часто может обнаружить ошибку дешифрирования и попытаться эксплуатировать ее. Например, в программе на Java исключение `javax.crypto.BadPaddingException` означает, что открытый текст был дополнен неправильно.

В 2010 году исследователи нашли возможности для атаки на оракул дополнения в нескольких серверах веб-приложений. Запрос допустимости заключался в отправке шифртекста некоторой системе, после чего анализировалось, была ли ошибка. С помощью таких запросов удалось дешифрировать безопасные в остальных отношениях шифртексты, не зная ключа.

Криптографы часто игнорируют атаки на оракул дополнения, потому что обычно они зависят от поведения приложения и от способа взаимодействия с ним пользователей. Но если вы не предвидите таких атак и не включаете их в модель при проектировании и развертывании криптографической системы, то можете столкнуться с неприятными сюрпризами.

Для дополнительного чтения

В этой книге мы будем подробно обсуждать шифрование и его различные формы, особенно работу современных безопасных шифров. Но не сможем охватить все на свете, так что многие увлекательные темы останутся за бортом. Например, чтобы изучить теоретические основания шифрования и глубже познакомиться с понятием неразличимости (IND), рекомендую прочитать вышедшую в 1982 году статью Goldwasser and Micali «Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information», в которой впервые была сформулирована идея семантической безопасности. Если вас интересуют физические атаки и криптографическое оборудование, то обратитесь к материалам конференции CHES (Conference on Cryptographic Hardware and Embedded Systems).

Существует гораздо больше типов шифрования, чем упомянуто в этой главе, в т. ч. шифрование на основе атрибутов, широковещательное шифрование, функциональное шифрование, личностное шифрование, шифрование с фиксацией сообщения, перешифрование с посредником – и это далеко не всё. Последние работы на эту тему можно найти на сайте *https://eprint.iacr.org/*, где хранится электронный архив научных статей по криптографии.