# Введение в криптографию

* Предисловие
* Базовая терминология
* Основные алгоритмы шифрования
* Цифровые подписи
* Криптографические хэш-функции
* Криптографические генераторы случайных чисел
* Обеспечиваемая шифром степень защиты
* Криптоанализ и атаки на криптосистемы

## Предисловие

Разные люди понимают под шифрованием разные вещи. Дети играют в игрушечные шифры и секретные языки. Это, однако, не имеет ничего общего с настоящей криптографией. Настоящая криптография (**strong cryptography**) должна обеспечивать такой уровень секретности, чтобы можно было надежно защитить критическую информацию от расшифровки крупными организациями --- такими как мафия, транснациональные корпорации и крупные государства. Настоящая криптография в прошлом использовалась лишь в военных целях. Однако сейчас, с становлением информационного общества, она становится центральным инструментом для обеспечения конфиденциальности.

По мере образования информационного общества, крупным государствам становятся доступны технологические средства тотального надзора за миллионами людей. Поэтому криптография становится одним из основных инструментов обеспечивающих конфиденциальность, доверие, авторизацию, электронные платежи, корпоративную безопасность и бесчисленное множество других важных вещей.

Криптография не является более придумкой военных, с которой не стоит связываться. Настала пора снять с криптографии покровы таинственности и использовать все ее возможности на пользу современному обществу. Широкое распространение криптографии является одним из немногих способов защитить человека от ситуации, когда он вдруг обнаруживает, что живет в тоталитарном государстве, которое может контролировать каждый его шаг.

## Базовая терминология

Представьте, что вам надо отправить сообщение адресату. Вы хотите, чтобы никто кроме адресата не смог прочитать отправленную информацию. Однако всегда есть вероятность, что кто-либо вскроет конверт или перехватит электронное послание.

В криптографической терминологии исходное послание именуют **открытым текстом** (**plaintext** или **cleartext**). Изменение исходного текста так, чтобы скрыть от прочих его содержание, называют **шифрованием** (**encryption**). Зашифрованное сообщение называют **шифротекстом** (**ciphertext**). Процесс, при котором из шифротекста извлекается открытый текст называют **дешифровкой** (**decryption**). Обычно в процессе шифровки и дешифровки используется некий **ключ** (**key**) и алгоритм обеспечивает, что дешифрование можно сделать лишь зная этот ключ.

**Криптография** --- это наука о том, как обеспечить секретность сообщения. **Криптоанализ** --- это наука о том, как **вскрыть** шифрованное сообщение, то есть как извлечь открытый текст не зная ключа. Криптографией занимаются **криптографы**, а криптоанализом занимаются **криптоаналитики**.

Криптография покрывает все практические аспекты секретного обмена сообщениями, включая аутенфикацию, цифровые подписи, электронные деньги и многое другое. **Криптология** --- это раздел математики, изучающий математические основы криптографических методов.

# Основные алгоритмы шифрования

Метод шифровки/дешифровки называют **шифром** (**cipher**). Некоторые алгоритмы шифрования основаны на том, что сам метод шифрования (алгоритм) является секретным. Ныне такие методы представляют лишь исторический интерес и не имеют практического значения. Все современные алгоритмы используют **ключ** для управления шифровкой и дешифровкой; сообщение может быть успешно дешифровано только если известен ключ. Ключ, используемый для дешифровки может не совпадать с ключом, используемым для шифрования, однако в большинстве алгоритмов ключи совпадают.

Алгоритмы с использованием ключа делятся на два класса: симметричные (или алгоритмы секретным ключом) и асиметричные (или алгоритмы с открытым ключом). Разница в том, что симметричные алгоритмы используют один и тот же ключ для шифрования и для дешифрования (или же ключ для дешифровки просто вычисляется по ключу шифровки). В то время как асимметричные алгоритмы используют разные ключи, и ключ для дешифровки не может быть вычислен по ключу шифровки.

Смметричные алгоритмы подразделяют на **потоковые шифры** и **блочные шифры**. Потоковые позволяют шифровать информацию побитово, в то время как блочные работают с некоторым набором бит данных (обычно размер блока составляет 64 бита) и шифруют этот набор как единое целое. Начальное представление о них можно получить в статье об алгоритмах.

Ассиметричные шифры (также именуемые алгоритмами с открытым ключом, или --- в более общем плане --- криптографией с открытым ключом) допускают, чтобы открытый ключ был доступн всем (скажем, опубликован в газете). Это позволяет любому зашифровать сообщение. Однако расшифровать это сообщение сможет только нужный человек (тот, кто владеет ключом дешифровки). Ключ для шифрования называют **открытым ключом**, а ключ для дешифрования --- **закрытым ключом** или **секретным ключом**.

Современные алгоритмы шифровки/дешифровки достаточно сложны и их невозможно проводить вручную. Настоящие криптографические алгоритмы разработаны для использования компьютерами или специальными аппаратными устройствами. В большинстве приложений криптография производится программным обеспечением и имеется множество доступных криптографических пакетов.

Вообще говоря, симметричные алгоритмы работают быстрее, чем ассиметричные. На практке оба типа алгоритмов часто используются вместе: алгоритм с открытым ключом используется для того, чтобы передать случайным образом сгенерированный секретный ключ, который затем используется для дешифровки сообщения.

Многие качественные криптографические алгоритмы доступны широко - в книжном магазине, библиотеке, патентном бюро или в Интернет. К широко известным симметричным алгоритмам относятся DES и IDEA, Наверное самым лучшим асимметричным алгоритмом является RSA. На страничке литературы приведен список хороших учебников по криптографии и смежным вопросам.

## Цифровые подписи

Некоторые из асимметричных алгоритмов могут использоваться для генерирования **цифровой подписи**. Цифровой подписью называют блок данных, сгенерированный с использованием некоторого секретного ключа. При этом с помощью открытого ключа можно проверить, что данные были действительно сгенерированы с помощью этого секретного ключа. Алгоритм генерации цифровой подписи должен обеспечивать, чтобы было невозможно без секретного ключа создать подпись, которая при проверке окажется правильной.

Цифровые подписи используются для того, чтобы подтвердить, что сообщение пришло действительно от данного отправителя (в предположении, что лишь отправитель обладает секретным ключом, соответствующим его открытому ключу). Также подписи используются для проставления **штампа времени** (**timestamp**) на документах: сторона, которой мы доверяем, подписывает документ со штампом времени с помошью своего секретного ключа и, таким образом, подтверждает, что документ уже существовал в момент, объявленный в штампе времени.

Цифровые подписи также можно использовать для удостоверения (**сертификации** --- **to certify**) того, что документ принадлежит определенному лицу. Это делается так: открытый ключ и информация о том, кому он принадлежит подписываются стороной, которой доверяем. При этом доверять подписывающей стороне мы можем на основании того, что ее ключ был подписан третьей стороной. Таким образом возникает иерархия доверия. Очевидно, что некоторый ключ должен быть корнем иерархии (то есть ему мы доверяем не потому, что он кем-то подписан, а потому, что мы верим a-priori, что ему можно доверять). В **централизованной инфраструктуре ключей** имеется очень небольшое количество корневых ключей сети (например, облеченные полномочиями государственные агенства; их также называют **сертификационными агенствами** --- **certification authorities**). В **распределенной инфраструктуре** нет необходимости иметь универсальные для всех корневые ключи, и каждая из сторон может доверять своему набору корневых ключей (скажем своему собственному ключу и ключам, ею подписанным). Эта концепция носит название **сети доверия** (**web of trust**) и реализована, например, в PGP.

Цифровая подпись документа обычно создается так: из документа генерируется так называемый **дайджест** (**message digest**) и к нему добавляется информация о том, кто подписывает документ, штамп времени и прочее. Получившаяся строка далее зашифровывается секретным ключом подписывающего с использованием того или иного алгоритма. Получившийся зашифрованный набор бит и представляет собой подпись. К подписи обычно прикладывается открытый ключ подписывающего. Получатель сначала решает для себя доверяет ли он тому, что открытый ключ принадлежит именно тому, кому должен принадлежать (с помощью сети доверия или априорного знания), и затем дешифрует подпись с помощью открытого ключа. Если подпись нормально дешифровалась, и ее содержимое соответствует документу (дайджест и др.), то сообщение считается подтвержденным.

Свободно доступны несколько методов создания и проверки цифровых подписей. Наиболее известным является алгоритм RSA.

## Криптографические хэш-функции

**Криптографические хэш-функции** используются обычно для генерации дайджеста сообщения при создании цифровой подписи. Хэш-функции отображают сообщение в имеющее фиксированный размер **хэш-значение** (**hash value**) таким образом, что все множество возможных сообщений распределяется равномерно по множеству хэш-значений. При этом криптографическая хэш-функция делает это таким образом, что практически невозможно подогнать документ к заданному хэш-значению.

Криптографические хэш-функции обычно производят значения длиной в 128 и более бит. Это число значительно больше, чем количество собщений, которые когда-либо будут существовать в мире.

Много хороших криптографических хэш-функций доступно бесплатно. Широко известные включают MD5 и SHA.

## Криптографические генераторы случайных чисел

**Криптографические генераторы случайных чисел** производят случайные числа, которые используются в криптографических приложениях, например - для генерации ключей. Обычные генераторы случайных чисел, имеющиеся во многих языках программирования и программных средах, не подходят для нужд криптографии (они создавались с целью получить статистически случайное распределение, криптоаналитики могут предсказать поведение таких случайных генераторов).

В идеале случайные числа должны основываться на настоящем физическом источнике случайной информации, которую невозможно предсказать. Примеры таких источников включают шумящие полупроводниковые приборы, младшие биты оцифрованного звука, интервалы между прерываниями устройств или нажатиями клавиш. Полученный от физического источника шум затем "дистиллируется" криптографической хэш-функцией так, чтобы каждый бит зависел от каждого бита. Достаточно часто для хранения случайной информации используется довольно большой пул (несколько тысяч бит) и каждый бит пула делается зависимым от каждого бита шумовой информаци и каждого другого бита пула криптографически надежным (**strong**) способом.

Когда нет настоящего физического источника шума, приходится пользоваться псевдослучайными числами. Такая ситуация нежелательна, но часто возникает на компьютерах общего назначения. Всегда желательно получить некий шум окружения --- скажем от величины задержек в устройствах, цифры статистики использования ресурсов, сетевой статистики, прерываний от клавиатуры или чего-то иного. Задачей является получить данные, непредсказуемые для внешнего наблюдателя. Для достижения этого случайный пул должен содержать как минимум 128 бит настоящей энтропии.

Криптографические генераторы псевдослучайных чисел обычно используют большой пул (seed-значение), содержащий случайную информацию. Биты генерируется путем выборки из пула с возможным прогоном через криптографическую хэш-функцию, чтобы спрятать содержимое пула от внешнего наблюдателя. Когда требуется новая порция бит, пул перемешивается путем шифровки со случайным ключом (его можно взять из неиспользованной пока части пула) так, чтобы каждый бит пула зависел от каждого другого бита. Новый шум окружения должен добавляться к пулу перед перемешиваниям, дабы сделать предсказание новых значений пула еще более сложным.

Несмотря на то, что при аккуратном проектировании криптографически надежный генератор случайных чисел реализовать не так уж и трудно, этот вопрос часто упускают из вида. Таким образом, следует подчеркнуть важность криптографического генератора случайных чисел --- если он сделан плохо, он может легко стать самым уязвимым элементом системы.

Доступны несколько примеров криптографических генераторов случайных чисел.

## Обеспечиваемая шифром степень защиты

Хорошие криптографические системы создаются таким образом, чтобы сделать их вскрытие как можно более трудным делом. Можно построить системы, которые на практике невозможно вскрыть (хотя доказать сей факт обычно нельзя). При этом не требуется очень больших усилий для реализации. Единственное, что требуется --- это аккуратность и базовые знания. Нет прощения разработчику, если он оставил возможность для вскрытия системы. Все механизмы, которые могут использоваться для взлома системы надо задокументировать и довести до сведения конечных пользователей.

Теоретически, любой шифровальный алгоритм с использованием ключа может быть вскрыт методом перебора всех значений ключа. Если ключ подбирается методом **грубой силы** (**brute force**), требуемая мощность компьютера растет экспоненциально с увеличением длины ключа. Ключ длиной в 32 бита требует 2^32 (около 10^9) шагов. Такая задача под силу любому дилетанту и решается на домашнем компьютере. Системы с 40-битным ключом (например, экспортный американский вариант алгоритма RC4) требуют 2^40 шагов --- такие компьютерные мощности имеются в большинстве университетов и даже в небольших компаниях. Системы с 56-битными ключами (DES) требуют для вскрытия заметных усилий, однако могут быть легко вскрыты с помощью специальной аппаратуры. Стоимость такой аппаратуры значительна, но доступна для мафии, крупных компаний и правительств. Ключи длиной 64 бита в настоящий момент, возможно, могут быть вскрыты крупными государствами и уже в ближайшие несколько лет будут доступны для вскрытия преступными организацими, крупными компаниями и небольшими государствами. Ключи длиной 80 бит могут в будущем стать уязвимыми. Ключи длиной 128 бит вероятно останутся недоступными для вскрытия методом грубой силы в обозримом будущем. Можно использовать и более длинные ключи. В пределе нетрудно добиться того, чтобы энергия, требуемая для вскрытия (считая, что на один шаг затрачивается минимальный квантовомеханический квант энергии) превзойдет массу солнца или вселенной.

Однако, длина ключа это еще не все. Многие шифры можно вскрыть и не перебирая всех возможных комбинаций. Вообще говоря, очень трудно придумать шифр, который нельзя было бы вскрыть другим более эффективным способом. Разработка собственных шифров может стать приятным занятием, но для реальных приложений использовать самодельные шифры не рекомендуется если вы не являетесь экспертом и не уверены на 100 процентов в том, что делаете.

Вообще говоря, следует держаться в стороне от неопубликованных или секретных алгоритмов. Часто разработчик такого алгоритма не уверен в его надежности, или же надежность зависит от секретности самого алгоритма. Вообще говоря, ни один алгоритм, секретность которого зависит от секретности самого алгоритма не явяется надежным. В частности, имея шифрующую программу, можно нанять прграммиста, который дизассемблирует ее и восстановит алгоритм методом обратной инженерии. Опыт показывает, что большинство секретных алгоритмов, ставших впоследствии достоянием общественности, оказались до смешного ненадежными.

Длины ключей, используемых в криптографии с открытым ключом обычно значительно больше, чем в симметричных алгоритмах. Здесь проблема заключается не в подборе ключа, а в воссоздании секретного ключа по открытому. В случае RSA проблема эквивалентна разложению на множители большого целого числа, которое является произведением пары неизвестных простых чисел. В случае некоторых других криптосистем, проблема эквивалентна вычислению дискретного логарифма по модулю большого целого числа (такая задача считается примерно аналогичной по трудности задаче разложения на множители). Имеются криптосистемы, которые используют другие проблемы.

Чтобы дать представление о степени сложности вскрытия RSA, скажем, что модули длиной 256 бит легко факторизуются обычными программистами. Ключи в 384 бита могут быть вскрыты исследовательской группой университета или компании. 512-битные ключи находятся в пределах досягаемости крупных государств. Ключи длиной в 768 бит вероятно не будут надежны продолжительное время. Ключи длиной в 1024 бит могут считаться безопасными до тех пор, пока не будет существенного прогресса в алгоритме факторизации; ключи длиной в 2048 большинство считает надежными на десятилетия. Более подробную информацию о длинах ключей RSA можно почерпнуть из статьи Брюса Шнайера (Bruce Scheier).

Важно подчеркнуть, что **степень надежности криптографической системы определяется ее слабейшим звеном**. Нельзя упускать из вида ни одного аспекта разработки системы --- от выбора алгоритма до политики использования и распространения ключей.

## Криптоанализ и атаки на криптосистемы

Криптоанализ - это наука о дешифровке закодированных сообщений не зная ключей. Имеется много криптоаналитических подходов. Некоторые из наиболее важных для разработчиков приведены ниже.

* **Атака со знанием лишь шифрованного текста** (**ciphertext-only attack**): Это ситуация, когда атакующий не знает ничего о содержании сообщения, и ему приходтся работать лишь с самим шифрованным текстом. На практике, часто можно сделать правдоподобные предположения о структуре текста, поскольку многие сообщения имеют стандартные заголовки. Даже обычные письма и документы начинаются с легко предсказумой информации. Также часто можно предположить, что некоторый блок информации содержит заданное слово.
* **Атака со знанием содержимого шифровки** (**known-plaintext attack**): Атакующий знает или может угадать содержимое всего или части зашифрованного текста. Задача заключается в расшифровке остального сообщения. Это можно сделать либо путем вычисления ключа шифровки, либо минуя это.
* **Атака с заданным текстом** (**chosen-plaintext attack**): Атакующий имеет возможнот получить шифрованный документ для любого нужного ему текста, но не знает ключа. Задачей является нахождение ключа. Некоторые методы шифрования и, в частности, RSA, весьма уязвимы для атак этого типа. При использовании таких алгоритмов надо тщательно следить, чтобы атакующий не мог зашифровать заданный им текст.
* **Атака с подставкой** (**Man-in-the-middle attack**): Атака направлена на обмен шифрованными сообщениями и, в особенности, на протокол обмена ключами. Идея заключается в том, что когда две стороны обмениваются ключами для секретной коммуникации (например, используя шифр Диффи-Хелмана, Diffie-Hellman), противник внедряется между ними на линии обмена сообщениями. Далее противник выдает каждой стороне свои ключи. В результате, каждая из сторон будет иметь разные ключи, каждый из которых известен противнику. Теперь противник будет расшифровывать каждое сообщение своим ключом и затем зашифровывать его с помощью другого ключа перед отправкой адресату. Стороны будут иметь иллюзию секретной переписки, в то время как на самом деле противник читает все сообщения.

Одним из способов предотвратить такой тип атак заключается в том, что стороны при обмене ключами вычисляют криптографическую хэш-функцию значения протокола обмена (или по меньшей мере значения ключей), подписывают ее алгоритмом цифровой подписи и посылают подпись другой стороне. Получатель проверит подпись и то, что значение хэш-функции совпадает с вычисленным значением. Такой метод используется, в частности, в системе Фотурис (Photuris).

* **Атака с помощью таймера** (**timing attack**): Этот новый тип атак основан на последовательном измерении времен, затрачиваемых на выполнение операции возведения в стенень по модулю целого числа. Ей подвержены по крайней мере следующие шифры: RSA, Диффи-Хеллман и метод эллиптических кривых. В статье Пола Кочера подробно рассмотрен этот метод.

Имеется множество других криптографических атак и криптоаналитических подходов. Однако приведенные выше являются, по-видимому, наиболее важными для практической разработки систем. Если кто-либо собирается создавать свой алгоритм шифрования, ему необходимо понимать данные вопросы значительно глубже. Одно из мест, где можно начать систематическое изучение информации --- это замечательная книга Брюса Шнейера "Прикладная криптография" (Bruce Schneier, Applied Cryptography).

**Перевод статьи Tatu Ylonen "Introduction to Cryptography"**