# Криптографические алгоритмы: основные понятия

## Информация и криптография

Человеческая цивилизация в течение всего времени существования работает с различной информацией. Информация может представлять собой как исторические сведения или описания технологических процессов, так и частные данные человека или группы лиц. Сведения, составляющие информацию и представляющие определенную ценность, должны быть защищены от лиц, не имеющих соответствующих прав доступа. Поэтому задача защиты информации всегда была актуальной.

Защитить информацию можно тремя способами [Ященко, стр 9].

1. Создание абсолютно надежного и изолированного от доступа извне хранилища информации и инфраструктуры для него. Это является крайне трудоемкой и дорогостоящей задачей.
2. Скрытие факта существования или передачи информации. Средства и методы такого скрытия изучает стеганография.
3. Хранение и передача информации, преобразованной таким образом, чтобы обратное преобразование могли совершить только определенные лица. Средства и методы такого преобразования информации изучает криптография.

Формализуем вышесказанное.

Информация – сведения, передаваемые одними людьми другим людям устным, письменным или каким-либо другим способом, а также сам процесс передачи или получения этих сведений [БСЭ].

Защита информации – деятельность, направленная на предотвращение утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию [ГОСТ Р 50922-2006]. Утечка – неконтролируемое распространение защищаемой информации в результате ее разглашения, несанкционированного доступа к информации и получения защищаемой информации иностранными разведками [ГОСТ Р 53114-2008].

Криптография — инженерно-техническая дисциплина, изучающая математические методы защиты информации (шифры). Криптография включает в себя криптосинтез и криптоанализ. *Криптосинтез* изучает подходы к разработке шифров. *Криптоанализ* изучает подходы к вскрытию шифров [Ященко, стр 16-17].

## Шифры: основные понятия и принципы

Итак, шифр (криптографический алгоритм, шифрующая функция) – преобразование данных, обеспечивающее их защищённость (зашифрованность). Это преобразование вида , где – исходные, незашифрованные данные (представленные в виде целого числа), – полученный зашифрованный текст (также целое число), – шифрующая функция.

Дешифрующая функция – функция, обратная к шифрующей: . Результатом применения дешифрующей функции к зашифрованным данным являются исходные данные.

Для того, чтобы шифрование было эффективным, необходимо, чтобы шифрующая функция была легко вычислимой, для затруднения дешифровки дешифрующая функция должна быть достаточно сложна для вычисления [Музыкантский, стр 13]. Функции такого вида называются односторонними.

Функция называется *односторонней*, если выполняются условия:

* для любого из некоторого множества существует эффективный алгоритм вычисления ;
* не существует эффективного алгоритма обращения функции [Музыкантский, стр 14].

Однако шифр должен быть не только недоступным для взлома злоумышленником. Законный получатель должен иметь возможность эффективно получить исходные данные. Это требование формализуется как «функция с секретом» или «односторонняя функция с секретом».

Функция называется *односторонней функцией с секретом*, если выполняются условия:

* для любого из некоторого множества существует эффективный алгоритм вычисления ;
* функция обладает «секретным свойством» , таким что:
  + при использовании свойства можно построить эффективный алгоритм построения обратной функции ;
  + если свойство неизвестно, то не существует эффективного алгоритма обращения функции [Музыкантский, стр 15].

Итак, при использовании шифра на основе функции с секретом, шифровать (вычислять функцию ) могут все, а эффективно расшифровывать – только лица, которым известно секретное свойство .

Секретное свойство также называют *ключом шифра*. Он представляет собой число или набор чисел (параметров алгоритма).

Главная часть шифра, его сердце – некоторая *трудная* задана, которая используется для гарантии того, что узнать секретное свойство (т. е. обратить шифрующую функцию и получить доступ к зашифрованным данным) очень трудно. Пара примеров трудных задач:

* Задача факторизации – разложение большого числа на простые множители (используется, например, в алгоритме RSA). В этом случае секретное свойство – факторизуемое число.
* Задача дискретного логарифмирования – обращение функции вида , где – большое простое число, – параметр, подобранный для конкретного (используется, например, в схеме Эль-Гамаля).

Однако если будет представлен алгоритм, эффективно решающий трудную задачу (т. е. задача перестанет быть трудной), шифры на ее основе мгновенно устареют. Поэтому поиск трудных задач (и, наоборот, поиски быстро решающих их алгоритмов) является одним из важных направлений криптографии.

С другой стороны, в секрете хранится только ключ алгоритма, сам алгоритм должен быть открытым. То есть вся криптографическая стойкость алгоритма должна содержаться в ключе, а злоумышленник может знать о криптографической схеме все, кроме ключа. Это правило называется *принципом Керкгоффса*. Его выполнение обеспечивает надежность системы: даже в случае подбора ключа (методом перебора или, например, шантажа) злоумышленником достаточно его сменить, и начинать взлом нужно с самого начала. Похожий принцип «враг знает систему» был (вероятно, независимо) сформулирован Шенноном и называется *максимой Шеннона* [Бауэр, стр 222].

Это, однако, является только рекомендацией. В правительственных, военных и ряде других областей применяется противоположный принцип: *безопасность через неясность*. Его суть состоит в скрытии внутренней структуры системы безопасности. Для шифров общего назначения он не рекомендуется, их, напротив, публикуют и обсуждают, чтобы коллективно найти уязвимости и возможности улучшения [<https://danielmiessler.com/study/security-by-obscurity/>].

## Симметричная криптография

Изначально существовали только симметричные схемы шифрования. Они предполагают наличие одного ключа, используемого как для шифрования, так и для расшифровки сообщений. Он хранится в тайне, поэтому такие схемы также называются схемами с закрытым ключом. Классический пример – алгоритм DES.

Симметричные шифры в основном состоят из двух главных компонентов: шифры подстановки (перестановка символов/битов сообщения) и шифры замены (замена символов/битов по отдельности на другие по некоторому правилу) [Ященко, стр 14].

Симметричные шифры делятся на две категории: блочные и потоковые. Потоковые криптоалгоритмы обрабатывают данные побитово или побайтово. Блочные алгоритмы обрабатывают данные целыми группами битов (блоками), обычно размер блока кратен 64 и составляет от 64 до 256 бит.

Симметричные шифры обладает рядом недостатков:

* Ключ должен храниться в тайне обеими сторонами и передаваться только по *защищенному* каналу (либо в зашифрованном виде). Это требует дополнительных ресурсов.
* Шифровать могут только те, кто знает ключ. Это означает либо жесткие ограничения на количество шифрующих лиц, либо значительный риск компрометации ключа в случае ослабления этих ограничений (компрометация ключа – его раскрытие не криптографическим способом [Шнайер стр 19]).

## Асимметричная криптография

После появления понятия односторонней функции появился другой вид алгоритмов – асимметричные. Такой способ шифрования предполагает наличие известного всем участникам схемы шифрования *открытого* ключа и известного только законным участникам *закрытого* ключа. Открытый ключ используется для шифрования данных, а закрытый – для расшифровки. Таким образом, шифровать сообщения может кто угодно, а вот расшифровывать – только законные пользователи.

Такой способ работы снимает основные проблемы симметричного шифрования: хранить в секрете шифрующий ключ нет необходимости, также он может передаваться по открытым каналам в незашифрованном виде и транслироваться на любую аудиторию.

По причине наличия открытого ключа такие алгоритмы также называются алгоритмами с открытым ключом. Классический пример – алгоритм RSA.

В некоторых случаях, например при использовании цифровой подписи, сообщения шифруются закрытым ключом, а расшифровываются открытым.

## Криптографические хэш-функции

Другим направлением криптографии, основанным на понятии односторонней функции, являются криптографические (односторонние, однонаправленные) хэш-функции.

Хэш-функция – функция, принимающая строку произвольной (или почти произвольной) длины, и преобразующую ее в строку фиксированной, обычно меньшей, длины. Полученная строка называется отпечатком (дайджестом) входной строки или ее хэш-кодом.

Однонаправленная хэш-функция вычисляется только в одном направлении: легко вычислить значение дайджеста по входной строке, но крайне трудно создать прообраз, дайджест которого соответствует заданной строке [Шнайер, стр 37].

Криптографическая хэш-функция является открытой. Безопасность обеспечивается именно однонаправленностью функции. Одним из необходимых условий является изменение значения половины битов дайджеста при изменении в одном бите входа: невозможно путем сравнения близких входов обратить хэширующую функцию.

Хэш-функции применяются при вычислении контрольных сумм для проверки подлинности файлов и транзакций: почти невозможно подобрать поддельный файл, отличный от настоящего, но с таким же дайджестом. При таком применении от функции требуется высокое быстродействие, т. к. файлы могут иметь значительный размер, а транзакции часто должны обрабатываться в реальном времени.

Еще одно использование контрольных сумм – при использовании цифровой подписи можно подписывать не весь документ целиком, а только его контрольную сумму. Это значительно ускоряет работу с подписью и уменьшает потребление памяти при хранении подписей [Шнайер, стр 42].

Также криптографические хэш-функции используются для хранения паролей. Такие хэш-функции называются Key Derivation Functions, KDF. В этом случае на носителе пароль не сохраняется, а хранится только его хэш, и каждый раз при вводе строки вычисляется ее хэш-код и сравнивается с хэшем пароля. Если произошло совпадение, значит, считаем, пароль введен верно. Для минимизации вероятности коллизии можно хранить два дайджеста от разных хэш-функций.

Такой способ хранения паролей является в настоящее время наиболее распространенным, так как сам по себе обеспечивает дополнительных уровень защищенности: даже если злоумышленник получил доступ к хэшам, для восстановления по ним паролей ему потребуется много времени (или большие вычислительные мощности). При этом от хэширующей пароли функции требуется, чтобы она вычислялась долго: время вычисления порядка 100 мс незаметно при авторизации, однако значительно затрудняет подбор пароля методом грубой силы. Кроме того, по тем же соображениям желательно большое потребление оперативной памяти, это также усложняет перебор, особенно в многопоточном режиме. К их стойкости также предъявляются повышенные требования.

Для дополнительного повышения защищенности хэшированных паролей используется *соль*. Это случайная строка, добавляемая к шифруемым данным, она должна храниться вместе с хэшем. Она не позволит понять, что захэшированы одинаковые строки, так как они будут иметь разную соль. Благодаря этому, брутфорс одного пароля из базы не позволяет найти другие такие же пароли в базе [Шнайер, стр 52].

Итак, есть два вида криптографических хэш-функций:

1. Быстрые, используются для вычисления контрольных сумм и должны потреблять как можно меньше ресурсов. Примеры: MD5, семейство SHA2, из новых – SHA3 и BLAKE2.
2. Медленные, используются для хэширования паролей. Должны потреблять много ресурсов и быть очень стойкими. Примеры: bcrypt и scrypt, новая – Argon2.

# Интернет вещей: основные понятия

## По Сэмюэлу Грингарду

Сэмюэл Грингард (Samuel Greengard) – журналист, специализируется на новых технологиях.  Директор по маркетингу во многих технологических и бизнес-изданиях, бывший президент Американского общества журналистов. В области IoT известен главным образом книгой «Интернет вещей: будущее уже здесь (The Internet of Things)». Далее приводятся избранные положения этой работы.

Подключаемые устройства – устройства, которые обмениваются данными по обычному интернет-соединению и получают дополнительные преимущества при подключении, например, через закрытую или частную сеть. Подключаемые устройства необязательно подсоединяются именно к Интернету вещей, но это происходит все чаще. Кроме того, подключаемость распространяется все дальше и дальше, выходя за пределы компьютеров, проникая во все уголки и закоулки мира.

Радиочастотная идентификация (RFID) – это основной инструмент, который позволяет устройствам стать подключаемыми. Эта технология основана на микросхемах, которые собирают информацию c датчиков, встроенных в машины или чипы, находящиеся на устройстве или внутри него. Для радиочастотной идентификации используются как «активные» метки с источником питания, так и «пассивные» метки, которым не требуется батарейка или иной источник питания. И те, и другие позволяют расположенным неподалеку считывателям собирать данные и обмениваться ими с компьютерами. Когда радиочастотный чип находится в радиусе действия считывателя, он автоматически посылает сигнал и данные.

Пассивные радиочастотные метки особенно востребованы благодаря низкой стоимости, долговечности и отсутствию необходимости в постоянном электропитании, они получают питание от ближайшего считывателя.

Промышленный Интернет – оборудование и аппаратура, оборудованные датчиками, которые делают их «умными». Эти устройства часто служат в качестве инженерной системы или IT-основы Интернета вещей. Например, промышленное оборудование или грузовик службы доставки могут транслировать данные с помощью Интернета вещей. В сфере промышленного Интернета обмен данными обычно осуществляется тремя разными способами: машина – машина (М – М), человек – машина (Ч – М) и машина – смартфон (М – С) (или другое устройство, например, планшет).

Таким образом, возможность сделать большинство устройств подключаемыми позволяет человеку централизованно (в смысле единого канала, Интернета) управлять практически всей техникой. В результате возникает *Интернет всего* (этот термин введен компанией Cisco Systems). Интернет всего позволяет значительно увеличить автоматизацию труда за счет упрощения взаимодействия отдельных устройств, модулей и систем.

В книге упоминается «парадокс автоматизации»: по мере того развития автоматизированных систем вероятность аварии или сбоя снижается, однако степень тяжести потенциальной опасности во много раз повышается. Это создает дополнительные требования к персоналу и пользователям таких систем. Кроме того, возникает психологический эффект расслабления, когда человек целиком полагается на технику.

Автор отмечает дилемму, стоящую перед разработчиками. Создание функциональных интерфейсов и средств управления делают устройства удобнее, но также делают их мишенью для атак. Но при этом, если возможности управлять устройством у пользователя нет, он не сможет обнаружить неисправность до тех пор, пока проблема или взлом не проявятся сами, причинив немалый ущерб. Следовательно, разработчики и производители должны находить новые креативные способы обеспечения безопасности.

## «Пакистанскаая группа»

Стоит упомянуть качественную работу [Burhan], посвященную обзору элементов сети IoT, ее «слоистой» архитектуры, а также актуальных проблем безопасности.

Для идентификации устройств используются адреса IPv4 и IPv6, для именования может использоваться система ucode. Сбор информации осуществляется, например, с помощью RFID-меток, носимых устройства. Для сетевого взаимодействия – основного компонента IoT – используются такие технологии, как RFID, NFC, Bluetooth, Wi-Fi, LTE. Обработка информации включает в себя отбор информации и затем необходимые вычисления. Примеры аппаратных платформ: Arduino, Raspberry Pi, Intel Galileo. Среди ОС можно назвать TinyOS, LiteOS, Android. Для обеспечения общей семантики используются унифицированные модели представления данных RDF, OWL, EXI.

Приложения IoT предоставляют следующие 4 типа услуг пользователю:

* Распределенное распознавание объектов.
* Сбор, обработка и распределенное хранение информации.
* Распределенная решающая (управляющая) система.
* Контроль и динамическая корректировка работы различных устройств.

Архитектура сети IoT состоит из нескольких уровней. Базовый трехуровневый вариант отражает главные черты архитектуры. Он состоит из следующих уровней:

* Прикладной уровень (application layer).
* Сетевой уровень (network layer, transmission layer).
* Сенсорный уровень (perception layer, sensor layer).

Сенсорный уровень. На этом уровне осуществляется идентификация «вещей» и сбор поставляемых ими данных. Данные могут быть самыми разными, в зависимости от конкретной области применения: местоположение, температура, вибрация и другие. Этот уровень зачастую является основной целью злоумышленников. Основные атаки:

* Подслушивание (eavesdropping) – перехват информации. Для таких атак уязвимы данные, передаваемые в открытом или слабо зашифрованном виде.
* Захват узла (node capture) – захват контроля над важным узлом передачи или временного хранения данных. Может привести к утечке сразу большого количества информации, в том числе ключей шифрования.
* Введение фальшивого узла (fake and malicious node) – добавление нового узла в систему. Эта атака направлена на прекращение передачи реальной информации и передачу вместо нее фальшивой. Кроме того, узел может пытаться получить контроль над другими узлами или заставить их повысить энергопотребление для причинения максимального урона сети.
* Атака повторного воспроизведения (replay attack, playback attack) – злоумышленник собирает информацию, передаваемую от отправителя к получателю, и фиксирует реакцию получателя. После чего позднее отправляет такую же информацию с целью заставить получателя выполнить желаемые действия.
* Атака по времени (timing attack) – злоумышленники фиксирует время реакции системы на различные запросы, чтобы получить сведения о ее устройстве и обнаружить потенциальные уязвимости. Может быть также направлена на одно устройство, в этом случае наиболее удобны слабые устройства (для них различие во времени зафиксировать легче).

Сетевой уровень. Он выступает как промежуточное звено между сенсорным уровнем и прикладным уровнем. Занимается передачей информации, а также отвечает за взаимодействие различных устройств и сетей между собой. Он уязвим для следующих атак:

* DoS-атака – создание условий для прекращения или затруднения доступа пользователей к вычислительной сети. Обычно это достигается путем искусственного наводнения сети большим количеством запросов.
* Атака посредника (man in the middle, MITM) – ретрансляция и изменение связи между узлами, которые считают, что общаются друг с другом. Является серьезной угрозой, так как атака может быть осуществлена в реальном времени.
* Атака на хранилище (storage attack) – как при централизованном, так и при распределенном способе хранения информации, она может быть украдена, подделана или удалена.
* Эксплойт-атака (exploit attack) – внедрение фрагментов кода, использующих уязвимости в системе безопасности, в приложении или в аппаратном обеспечении. Целью атаки является получение над контролем системы и кража информации, также возможно нарушение функционирования системы.

На прикладном уровне находятся все приложения, использующие технологию IoT. Атаки на данный уровень часто определяются конкретным назначением сети. Примеры атак:

* Межсайтовый скриптинг (cross-site scripting) – введение инъекции на стороне клиента. Эта атака позволяет злоумышленнику полностью изменить содержимое приложения в своих целях, а также украсть информацию.
* Атака вредоносного кода (malicious code attack) – вредоносный код в любой части приложения. Данные атаки часто предотвращаются антивирусом.
* Атаки на данные – в случае большого количества пользователей, в некоторых случаях данные могут передаваться в слабо защищенном виде, что может быть использовано злоумышленником.

С развитием технологии IoT, распространение получает более сложная, пятиуровневая модель. Модель выглядит следующим образом.

* Уровень бизнес-логики (business level).
* Прикладной уровень (application layer).
* Уровень обработки (processing layer).
* Транспортный уровень (transport layer).
* Сенсорный уровень (perception layer, sensor layer).

Уровень обработки собирает, отбирает и обрабатывает информацию, полученную от транспортного уровня. Возможные атаки:

* Атака истощения ресурсов (resource exhaustion attack) – атака, в результате которой происходит не «зависание» и перегрузка устройств, как при DoS-атаке, а сбои программного или аппаратного обеспечения системы. Благодаря распределенной природе системы интернета вещей, эти атаки не слишком опасны.
* Вредоносное ПО – атака на конфиденциальную информацию. Вирусы, шпионское ПО, реклама, троянские программы и черви.

Уровень бизнес-логики занимается управлением всей системой. Устанавливает политики приложений, конфиденциальности данных, обработки данных. Уязвимость этого слоя позволит злоумышленникам «легально» использовать приложения в обход бизнес-логики. Возможные варианты атак:

* Атака на бизнес-логику (business logic attack) – использование ошибок программирования бизнес-уровня. Позволяет изменить взаимодействие между пользователем и БД приложения в сторону, выгодную злоумышленнику. Это может быть достигнуто при использовании уязвимостей кода, слабостей процедур валидации при восстановлении пароля и введении входных данных, или слабостей шифрования.
* Атака нулевого дня (zero-day attack) – использование уязвимостей системы безопасности, которые ранее были неизвестны.