# Лекция 7 Функции и задачи защиты информации

## Защитные механизмы операционных систем

### *Идентификация и аутентификация*

Наиболее распространенным способом контроля доступа является процедура регистрации. Обычно каждый пользователь в системе имеет уникальный идентификатор. Идентификация заключается в сообщении пользователем своего идентификатора. Для того чтобы установить, что пользователь именно тот, за кого себя выдает, то есть что именно ему принадлежит введенный идентификатор, в информационных системах предусмотрена процедура аутентификации (authentication, опознавание, в переводе с латинского означает "установление подлинности"), задача которой - предотвращение доступа к системе нежелательных лиц.

Распространены способы идентификации:

- с помощью ключа или магнитной карты;
- с использованием пароля;
- с применением отпечатков пальцев, подписи, голоса.

### *Пароли, уязвимость паролей*

Наиболее простой подход к аутентификации – применение пользовательского пароля.

Когда пользователь идентифицирует себя при помощи уникального идентификатора или имени, у него запрашивается пароль. Если пароль, сообщенный пользователем, совпадает с паролем, хранящимся в системе, система предполагает, что пользователь легитимен. Пароли часто используются для защиты объектов в компьютерной системе в отсутствие более сложных схем защиты.

Недостатки паролей связаны с тем, что трудно сохранить баланс между удобством пароля для пользователя и его надежностью. Пароли могут быть угаданы, случайно показаны или нелегально переданы авторизованным пользователем неавторизованному.

Есть два общих способа угадать пароль. Один связан со сбором информации о пользователе. Люди обычно используют в качестве паролей очевидную информацию (скажем, имена животных или номерные знаки автомобилей). Для иллюстрации важности разумной политики назначения идентификаторов и паролей можно привести данные исследований, проведенных в AT&T, показывающие, что из 500 попыток несанкционированного доступа около 300 составляют попытки угадывания паролей или беспарольного входа по пользовательским именам guest, demo и т. д.

Другой способ – попытаться перебрать все наиболее вероятные комбинации букв, чисел и знаков пунктуации (атака по словарю). Например, четыре десятичные цифры дают только 10 000 вариантов, более длинные пароли, введенные с учетом регистра символов и пунктуации, не столь уязвимы, но, тем не менее, таким способом удается разгадать до 25% паролей. Чтобы заставить пользователя выбрать трудноугадываемый пароль, во многих системах внедрена реактивная проверка паролей, которая при помощи собственной программы-взломщика паролей может оценить качество пароля, введенного пользователем.

Несмотря на все это, пароли распространены, поскольку они удобны и легко реализуемы.

**Шифрование пароля**

Для хранения секретного списка паролей на диске во многих ОС используется криптография. Система задействует одностороннюю функцию, которую просто вычислить, но для которой чрезвычайно трудно (разработчики надеются, что невозможно) подобрать обратную функцию.

Например, в ряде версий Unix в качестве односторонней функции используется модифицированный вариант алгоритма DES. Введенный пароль длиной до 8 знаков преобразуется в 56-битовое значение, которое служит входным параметром для процедуры crypt(), основанной на этом алгоритме. Результат шифрования зависит не только от введенного пароля, но и от случайной последовательности битов, называемой привязкой (переменная salt). Это сделано для того, чтобы решить проблему совпадающих паролей. Очевидно, что саму привязку после шифрования необходимо сохранять, иначе процесс не удастся повторить. Модифицированный алгоритм DES выполняется, имея входное значение в виде 64-битового блока нулей, с использованием пароля в качестве ключа, а на каждой следующей итерации входным параметром служит результат предыдущей итерации. Всего процедура повторяется 25 раз. Полученное 64-битовое значение преобразуется в 11 символов и хранится рядом с открытой переменной salt.

В ОС Windows NT преобразование исходного пароля также осуществляется многократным применением алгоритма DES и алгоритма MD4.

Хранятся только кодированные пароли. В процессе аутентификации представленный пользователем пароль кодируется и сравнивается с хранящимися на диске. Таким образом, файл паролей нет необходимости держать в секрете.

При удаленном доступе к ОС нежелательна передача пароля по сети в открытом виде. Одним из типовых решений является использование криптографических протоколов. В качестве примера можно рассмотреть протокол опознавания с подтверждением установления связи путем вызова - CHAP (Challenge Handshake Authentication Protocol).

Опознавание достигается за счет проверки того, что у пользователя, осуществляющего доступ к серверу, имеется секретный пароль, который уже известен серверу.

Пользователь инициирует диалог, передавая серверу свой идентификатор. В ответ сервер посылает пользователю запрос (вызов), состоящий из идентифицирующего кода, случайного числа и имени узла сервера или имени пользователя. При этом пользовательское оборудование в результате запроса пароля пользователя отвечает следующим ответом, зашифрованным с помощью алгоритма одностороннего хеширования, наиболее распространенным видом которого является MD5. После получения ответа сервер при помощи той же функции с теми же аргументами шифрует собственную версию пароля пользователя. В случае совпадения результатов вход в систему разрешается. Существенно, что незашифрованный пароль при этом по каналу связи не посылается.

В микротелефонных трубках используется аналогичный метод.

В системах, работающих с большим количеством пользователей, когда хранение всех паролей затруднительно, применяются для опознавания сертификаты, выданные доверенной стороной (см., например, [Столлингс, 2001]).

**Авторизация. Разграничение доступа к объектам ОС**

После успешной регистрации система должна осуществлять авторизацию (*authorization*) – предоставление субъекту прав на *доступ* к объекту. Средства авторизации контролируют *доступ* легальных пользователей к ресурсам системы, предоставляя каждому из них именно те *права*, которые были определены администратором, а также осуществляют *контроль* возможности выполнения пользователем различных системных функций. Система контроля базируется на общей модели, называемой *матрицей доступа*. Рассмотрим ее более подробно.

Как уже говорилось в предыдущей лекции, компьютерная система может быть смоделирована как набор субъектов (процессы, пользователи) и объектов. Под объектами мы понимаем как ресурсы оборудования (*процессор*, *сегменты* памяти, принтер, диски и ленты), так и программные ресурсы (файлы, программы, семафоры), то есть все то, *доступ* к чему контролируется. Каждый *объект* имеет уникальное имя, отличающее его от других объектов в системе, и каждый из них может быть доступен через хорошо определенные и значимые *операции*.

*Операции* зависят от объектов. Например, *процессор* может только выполнять команды, *сегменты* памяти могут быть записаны и прочитаны, считыватель магнитных карт может только читать, а файлы данных могут быть записаны, прочитаны, переименованы и т. д.

Желательно добиться того, чтобы процесс осуществлял авторизованный *доступ* только к тем ресурсам, которые ему нужны для выполнения его задачи. Это требование минимума привилегий, уже упомянутое в предыдущей лекции, полезно с точки зрения ограничения количества повреждений, которые процесс может нанести системе. Например, когда процесс P вызывает процедуру A, ей должен быть разрешен *доступ* только к переменным и формальным параметрам, переданным ей, она не должна иметь возможность влиять на другие переменные процесса. Аналогично *компилятор* не должен оказывать влияния на произвольные файлы, а только на их хорошо определенное *подмножество* (исходные файлы, листинги и др.), имеющее *отношение* к компиляции. С другой стороны, *компилятор* может иметь личные файлы, используемые для оптимизационных целей, к которым процесс P не имеет доступа.

Различают *дискреционный* (избирательный) способ управления доступом и *полномочный* (*мандатный*).

При *дискреционном доступе*, подробно рассмотренном ниже, определенные *операции* над конкретным ресурсом запрещаются или разрешаются субъектам или группам субъектов. С концептуальной точки зрения текущее состояние прав доступа при дискреционном управлении описывается матрицей, в строках которой перечислены субъекты, в столбцах - объекты, а в ячейках - *операции*, которые субъект может выполнить над объектом.

***Полномочный*** подход заключается в том, что все объекты могут иметь уровни секретности, а все субъекты делятся на группы, образующие иерархию в соответствии с уровнем допуска к информации. Иногда это называют моделью многоуровневой безопасности, которая должна обеспечивать выполнение следующих правил.

- Простое свойство секретности. Субъект может читать информацию только из объекта, уровень секретности которого не выше уровня секретности субъекта. Генерал читает документы лейтенанта, но не наоборот.

- *-свойство. Субъект может записывать информацию в объекты только своего уровня или более высоких уровней секретности. Генерал не может случайно разгласить нижним чинам секретную информацию.

Некоторые авторы утверждают [Таненбаум, 2002], что последнее требование называют *-свойством, потому что в оригинальном докладе не смогли придумать для него подходящего названия. В итоге во все последующие документы и монографии оно вошло как *-свойство.

Отметим, что данная модель разработана для хранения секретов, но не гарантирует целостности данных. Например, здесь лейтенант имеет право писать в файлы генерала. Более подробно о реализации подобных формальных моделей рассказано в [Столлингс, 2002], [Таненбаум, 2002].

Большинство операционных систем реализуют именно *дискреционное управление доступом*. Главное его достоинство - гибкость, основные недостатки - рассредоточенность управления и сложность централизованного контроля.

**Домены безопасности**

Чтобы рассмотреть схему *дискреционного доступа* более детально, введем концепцию *домена безопасности* (protection domain). Каждый домен определяет набор объектов и типов операций, которые могут производиться над каждым объектом. Возможность выполнять операции над объектом есть права доступа, каждое из которых есть упорядоченная пара <object-name, rights-set>. Домен, таким образом, есть набор прав доступа. Например, если домен D имеет права доступа <file F, {read, write}>, это означает, что процесс, выполняемый в домене D, может читать или писать в файл F, но не может выполнять других операций над этим объектом. Пример доменов можно увидеть на рис.16.1.

| Домен \ Объект | F1 | F2 | F3 | Printer |
|---|---|---|---|---|
| D1 | read | | | |
| D2 | | | | print |
| D3 | | read | execute | |
| D4 | read<br>write | | read<br>write | |

**Рис. 16.1.** Специфицирование прав доступа к ресурсам

Связь конкретных субъектов, функционирующих в операционных системах, может быть организована следующим образом.

- Каждый пользователь может быть доменом. В этом случае набор объектов, к которым может быть организован доступ, зависит от *идентификации* пользователя.
- Каждый процесс может быть доменом. В этом случае набор доступных объектов определяется *идентификацией* процесса.
- Каждая процедура может быть доменом. В этом случае набор доступных объектов соответствует локальным переменным, определенным внутри процедуры. Заметим, что когда процедура выполнена, происходит смена домена.

Рассмотрим стандартную двухрежимную модель выполнения ОС. Когда процесс выполняется в режиме системы (*kernel mode*), он может выполнять привилегированные

инструкции и иметь полный контроль над компьютерной системой. С другой стороны, если процесс выполняется в пользовательском режиме, он может вызывать только непривилегированные инструкции. Следовательно, он может выполняться только внутри предопределенного пространства памяти. Наличие этих двух режимов позволяет защитить ОС (kernel domain) от пользовательских процессов (выполняющихся в user domain). В мультипрограммных системах двух доменов недостаточно, так как появляется необходимость защиты пользователей друг от друга. Поэтому требуется более тщательно разработанная схема.

В ОС Unix домен связан с пользователем. Каждый пользователь обычно работает со своим набором объектов.

**Матрица доступа**

Модель безопасности, специфицированная в предыдущем разделе (см. рис. 16.1), имеет вид матрицы, которая называется ***матрицей доступа*** . Какова может быть эффективная реализация *матрицы доступа*? В общем случае она будет разреженной, то есть большинство ее клеток будут пустыми. Хотя существуют структуры данных для представления *разреженной матрицы*, они не слишком полезны для приложений, использующих возможности защиты. Поэтому на практике *матрица доступа* применяется редко. Эту матрицу можно разложить по столбцам, в результате чего получаются **списки прав доступа** (access control list - ACL). В результате разложения по строкам получаются *мандаты* возможностей (capability list или capability *tickets*).

**Список прав доступа. Access control list**

Каждая колонка в матрице может быть реализована как список доступа для одного объекта. Очевидно, что пустые клетки могут не учитываться. В результате для каждого объекта имеем список упорядоченных пар <domain, rights-set>, который определяет все домены с непустыми наборами прав для данного объекта.

Элементами списка могут быть процессы, пользователи или группы пользователей. При реализации широко применяется предоставление доступа по умолчанию для пользователей, права которых не указаны. Например, в Unix все субъекты-пользователи разделены на три группы (владелец, группа и остальные), и для членов каждой группы контролируются операции чтения, записи и исполнения (rwx). В итоге имеем ACL - 9-битный код, который является атрибутом разнообразных объектов Unix.

**Мандаты возможностей. Capability list**

Как отмечалось выше, если *матрицу доступа* хранить по строкам, то есть если каждый субъект хранит список объектов и для каждого объекта - список допустимых операций, то такой способ хранения называется ***"мандаты"*** или ***"перечни возможностей"*** (capability list) . Каждый пользователь обладает несколькими *мандатами* и может иметь право передавать их другим. *Мандаты* могут быть рассеяны по системе и вследствие этого представлять большую угрозу для безопасности, чем списки контроля доступа. Их хранение должно быть тщательно продумано.

Примерами систем, использующих *перечни возможностей*, являются *Hydra*, Cambridge CAP System [Denning, 1996].

**Другие способы контроля доступа**

Иногда применяется **комбинированный способ**. Например, в том же Unix на этапе открытия файла происходит анализ ACL (операция open). В случае благоприятного исхода файл заносится в список открытых процессом файлов, и при последующих

операциях чтения и записи проверки прав доступа не происходит. Список открытых файлов можно рассматривать как *перечень возможностей*.

Существует также схема **lock-key**, которая является компромиссом между списками прав доступа и *перечнями возможностей*. В этой схеме каждый объект имеет список уникальных битовых шаблонов (patterns), называемых locks. Аналогично каждый домен имеет список уникальных битовых шаблонов, называемых ключами (keys). Процесс, выполняющийся в домене, может получить доступ к объекту, только если домен имеет ключ, который соответствует одному из шаблонов объекта.

Как и в случае *мандатов*, список ключей для домена должен управляться ОС. Пользователям не разрешается проверять или модифицировать списки ключей (или шаблонов) непосредственно. Более подробно данная схема изложена в [Silberschatz, 2002].

**Смена домена**

В большинстве ОС для определения домена применяются идентификаторы пользователей. Обычно переключение между доменами происходит, когда меняется пользователь. Но почти все системы нуждаются в дополнительных механизмах смены домена, которые используются, когда некая привилегированная возможность необходима большому количеству пользователей. Например, может понадобиться разрешить пользователям иметь доступ к сети, не заставляя их писать собственные сетевые программы. В таких случаях для процессов ОС Unix предусмотрена установка бита **set-uid**. В результате установки этого бита в сетевой программе она получает привилегии ее создателя (а не пользователя), заставляя домен меняться на время ее выполнения. Таким образом, рядовой пользователь может получить нужные привилегии для доступа к сети.

**Недопустимость повторного использования объектов**

Контроль *повторного использования объекта* предназначен для предотвращения попыток незаконного получения конфиденциальной информации, остатки которой могли сохраниться в некоторых объектах, ранее использовавшихся и освобожденных другим пользователем. Безопасность повторного применения должна гарантироваться для областей оперативной памяти (в частности, для буферов с образами экрана, расшифрованными паролями и т. п.), для дисковых блоков и магнитных носителей в целом. Очистка должна производиться путем записи маскирующей информации в объект при его освобождении (перераспределении). Например, для дисков на практике применяется способ двойной перезаписи освободившихся после удаления файлов блоков случайной битовой последовательностью.

**Выявление вторжений. Аудит системы защиты**

Даже самая лучшая система защиты рано или поздно будет взломана. Обнаружение попыток вторжения является важнейшей задачей системы защиты, поскольку ее решение позволяет минимизировать *ущерб* от взлома и собирать информацию о методах вторжения. Как правило, поведение взломщика отличается от поведения легального пользователя. Иногда эти различия можно выразить количественно, например подсчитывая число некорректных вводов пароля во время регистрации.

Основным инструментом выявления вторжений является *запись* данных *аудита*. Отдельные действия пользователей протоколируются, а полученный протокол используется для выявления вторжений.

***Аудит***, таким образом, заключается в регистрации специальных данных о различных типах событий, происходящих в системе и так или иначе влияющих на

состояние безопасности компьютерной системы. К числу таких событий обычно причисляют следующие:

- вход или выход из системы;
- операции с файлами (открыть, закрыть, переименовать, удалить);
- обращение к удаленной системе;
- смена привилегий или иных атрибутов безопасности (режима доступа, уровня благонадежности пользователя и т. п.).

Если фиксировать все события, объем регистрационной информации, скорее всего, будет расти слишком быстро, а ее эффективный *анализ* станет невозможным. Следует предусматривать наличие средств *выборочного протоколирования* как в отношении пользователей, когда *слежение* осуществляется только за подозрительными личностями, так и в отношении событий. Слежка важна в первую *очередь* как профилактическое средство. Можно надеяться, что многие воздержатся от нарушений безопасности, зная, что их действия фиксируются.

Помимо протоколирования, можно периодически **сканировать** систему на наличие слабых мест в системе безопасности. Такое сканирование может проверить разнообразные аспекты системы:

- короткие или легкие пароли;
- неавторизованные set-uid программы, если система поддерживает этот механизм;
- неавторизованные программы в системных директориях;
- долго выполняющиеся программы;
- нелогичная защита как пользовательских, так и системных директорий и файлов. Примером нелогичной защиты может быть файл, который запрещено читать его автору, но в который разрешено записывать информацию постороннему пользователю;
- потенциально опасные списки поиска файлов, которые могут привести к запуску "троянского коня";
- изменения в системных программах, обнаруженные при помощи контрольных сумм.

Любая проблема, обнаруженная сканером безопасности, может быть как ликвидирована автоматически, так и передана для решения менеджеру системы.

**Анализ некоторых популярных ОС с точки зрения их защищенности**

Итак, ОС должна способствовать реализации мер безопасности или непосредственно поддерживать их. Примерами подобных решений в рамках аппаратуры и операционной системы могут быть:

- разделение команд по уровням привилегированности;
- сегментация адресного пространства процессов и организация защиты сегментов;
- защита различных процессов от взаимного влияния за счет выделения каждому своего виртуального пространства;
- особая защита ядра ОС;
- контроль *повторного использования объекта* ;
- наличие средств управления доступом;
- структурированность системы, явное выделение надежной вычислительной базы (совокупности защищенных компонентов), обеспечение компактности этой базы;
- следование принципу *минимизации привилегий* - каждому компоненту дается ровно столько привилегий, сколько необходимо для выполнения им своих функций.

Большое *значение* имеет структура файловой системы. Например, в ОС с дискреционным контролем доступа каждый *файл* должен храниться вместе с дискреционным списком прав доступа к нему, а, например, при копировании файла все атрибуты, в том числе и *ACL*, должны быть автоматически скопированы вместе с телом файла.

В принципе, меры безопасности не обязательно должны быть заранее встроены в ОС - достаточно принципиальной возможности дополнительной установки защитных продуктов. Так, сугубо ненадежная система *MS-DOS* может быть усовершенствована за счет средств проверки паролей доступа к компьютеру и/или жесткому диску, за счет борьбы с вирусами путем отслеживания попыток записи в *загрузочный сектор* CMOS-средствами и т. п. Тем не менее по-настоящему надежная система должна изначально проектироваться с акцентом на *механизмы* безопасности.

**MS-DOS**

ОС MS-DOS функционирует в реальном режиме (real-mode) процессора i80x86. В ней невозможно выполнение требования, касающегося изоляции программных модулей (отсутствует аппаратная защита памяти). Уязвимым местом для защиты является также файловая система FAT, не предполагающая у файлов наличия атрибутов, связанных с разграничением доступа к ним. Таким образом, MS-DOS находится на самом нижнем уровне в иерархии защищенных ОС.

**NetWare, IntranetWare**

Замечание об отсутствии изоляции модулей друг от друга справедливо и в отношении рабочей станции NetWare. Однако NetWare - **сетевая** ОС, поэтому к ней возможно применение и иных критериев. Это на данный момент единственная сетевая ОС, сертифицированная по классу С2 (следующей, по-видимому, будет Windows 2000). При этом важно изолировать наиболее уязвимый участок системы безопасности NetWare - консоль сервера, и тогда следование определенной практике поможет увеличить степень защищенности данной сетевой операционной системы. Возможность создания безопасных систем обусловлена тем, что число работающих приложений **фиксировано** и пользователь не имеет возможности запуска своих приложений.

**OS/2**

OS/2 работает в защищенном режиме (protected-mode) процессора i80x86. Изоляция программных модулей реализуется при помощи встроенных в этот процессор механизмов защиты памяти. Поэтому она свободна от указанного выше коренного недостатка систем типа MS-DOS. Но OS/2 была спроектирована и разработана без учета требований по защите от несанкционированного доступа. Это сказывается прежде всего на файловой системе. В файловых системах OS/2 HPFS (high performance file system) и FAT нет места ACL. Кроме того, пользовательские программы имеют возможность запрета прерываний. Следовательно, сертификация OS/2 на соответствие какому-то классу защиты не представляется возможной.

**Считается**, что такие операционные системы, как MS-DOS, Mac OS, Windows, OS/2, имеют уровень защищенности D (по оранжевой книге). Но, если быть точным, нельзя считать эти ОС даже системами уровня безопасности D, ведь они никогда не представлялись на тестирование.

**Unix**

Рост популярности Unix и все большая осведомленность о проблемах безопасности привели к осознанию необходимости достичь приемлемого уровня безопасности ОС, сохранив при этом мобильность, гибкость и открытость программных продуктов. В Unix

есть несколько уязвимых с точки зрения безопасности мест, хорошо известных опытным пользователям, вытекающих из самой природы Unix (см., например, раздел "Типичные объекты атаки хакеров" в книге [Дунаев, 1996]). Однако хорошее системное администрирование может ограничить эту уязвимость.

Относительно защищенности Unix сведения противоречивы. В Unix изначально были заложены *идентификация* пользователей и разграничение доступа. Как оказалось, средства защиты данных в Unix могут быть доработаны, и сегодня можно утверждать, что многие клоны Unix по всем параметрам соответствуют классу безопасности С2.

Обычно, говоря о защищенности Unix, рассматривают защищенность автоматизированных систем, одним из компонентов которых является Unix-сервер. Безопасность такой системы увязывается с защитой глобальных и локальных сетей, безопасностью удаленных сервисов типа telnet и rlogin/rsh и *аутентификацией* в сетевой конфигурации, безопасностью X Window-приложений. На системном уровне важно наличие средств *идентификации* и *аудита*.

В Unix существует список именованных пользователей, в соответствии с которым может быть построена система разграничения доступа.

В ОС Unix считается, что информация, нуждающаяся в защите, находится главным образом в файлах.

По отношению к конкретному файлу все пользователи делятся на три категории:

- владелец файла;
- члены группы владельца;
- прочие пользователи.

Для каждой из этих категорий режим доступа определяет права на операции с файлом, а именно:

- право на чтение;
- право на запись;
- право на выполнение (для каталогов - право на поиск).

В итоге девяти (3x3) битов защиты оказывается достаточно, чтобы специфицировать ACL каждого файла.

Аналогичным образом защищены и другие объекты ОС Unix, например семафоры, сегменты разделяемой памяти и т. п.

Указанных видов прав достаточно, чтобы определить допустимость любой операции с файлами. Например, для удаления файла необходимо иметь право на запись в соответствующий каталог. Как уже говорилось, права доступа к файлу проверяются только на этапе открытия. При последующих операциях чтения и записи проверка не выполняется. В результате, если режим доступа к файлу меняется после того, как файл был открыт, это не сказывается на процессах, уже открывших этот файл. Данное обстоятельство является уязвимым с точки зрения безопасности местом.

Наличие всего трех видов субъектов доступа: владелец, группа, все остальные - затрудняет задание прав "с точностью до пользователя", особенно в случае больших конфигураций. В популярной разновидности Unix - Solaris имеется возможность использовать списки управления доступом (ACL), позволяющие индивидуально устанавливать права доступа отдельных пользователей или групп.

Среди всех пользователей особое положение занимает пользователь root, обладающий максимальными привилегиями. Обычные правила разграничения доступа к нему не применяются - ему доступна вся информация на компьютере.

В Unix имеются инструменты системного *аудита* - хронологическая запись событий, имеющих отношение к безопасности. К таким событиям обычно относят: обращения программ к отдельным серверам; события, связанные с входом/выходом в систему и другие. Обычно регистрационные действия выполняются специализированным syslog-демоном, который проводит запись событий в регистрационный журнал в соответствии с текущей конфигурацией. Syslog-демон стартует в процессе загрузки системы.

Таким образом, безопасность ОС Unix может быть доведена до соответствия классу C2. Однако разработка на ее основе автоматизированных систем более высокого класса защищенности может быть сопряжена с большими трудозатратами.

**Windows NT/2000/XP**

С момента выхода версии 3.1 осенью 1993 года в Windows NT гарантировалось соответствие уровню безопасности C2. В настоящее время (точнее, в 1999 г.) сертифицирована версия NT 4 с Service Pack 6a с использованием файловой системы NTFS в автономной и сетевой конфигурации. Следует помнить, что этот уровень безопасности не подразумевает защиту информации, передаваемой по сети, и не гарантирует защищенности от физического доступа.

Компоненты защиты NT частично встроены в ядро, а частично реализуются подсистемой защиты. Подсистема защиты контролирует доступ и учетную информацию. Кроме того, Windows NT имеет встроенные средства, такие как поддержка резервных копий данных и управление источниками бесперебойного питания, которые не требуются "Оранжевой книгой", но в целом повышают общий уровень безопасности.

ОС Windows 2000 сертифицирована по стандарту Common Criteria. В дальнейшем линейку продуктов Windows NT/2000/XP, изготовленных по технологии NT, будем называть просто Windows NT.

Ключевая цель системы защиты Windows NT - следить за тем, кто и к каким объектам осуществляет доступ. Система защиты хранит информацию, относящуюся к безопасности для каждого пользователя, группы пользователей и объекта. Единообразие контроля доступа к различным объектам (процессам, файлам, семафорам и др.) обеспечивается тем, что с каждым процессом связан маркер доступа, а с каждым объектом - дескриптор защиты. Маркер доступа в качестве параметра имеет идентификатор пользователя, а дескриптор защиты - списки прав доступа. ОС может контролировать попытки доступа, которые производятся процессами прямо или косвенно инициированными пользователем.

Windows NT отслеживает и контролирует доступ как к объектам, которые пользователь может видеть посредством интерфейса (такие, как файлы и принтеры), так и к объектам, которые пользователь не может видеть (например, процессы и именованные каналы). Любопытно, что, помимо разрешающих записей, списки прав доступа содержат и запрещающие записи, чтобы пользователь, которому доступ к какому-либо объекту запрещен, не смог получить его как член какой-либо группы, которой этот доступ предоставлен.

Система защиты ОС Windows NT состоит из следующих компонентов:

- Процедуры регистрации (Logon Processes), которые обрабатывают запросы пользователей на вход в систему. Они включают в себя начальную интерактивную процедуру, отображающую начальный диалог с пользователем на экране и удаленные процедуры входа, которые позволяют удаленным пользователям получить доступ с рабочей станции сети к серверным процессам Windows NT.

- Подсистемы локальной авторизации (Local Security Authority, LSA), которая гарантирует, что пользователь имеет разрешение на доступ в систему. Этот компонент - центральный для системы защиты Windows NT. Он порождает маркеры доступа, управляет локальной политикой безопасности и предоставляет интерактивным пользователям аутентификационные услуги. LSA также контролирует политику *аудита* и ведет журнал, в котором сохраняются сообщения, порождаемые диспетчером доступа.
- Менеджера учета (Security Account Manager, SAM), который управляет базой данных учета пользователей. Эта база данных содержит информацию обо всех пользователях и группах пользователей. SAM предоставляет услуги по легализации пользователей, применяющиеся в LSA.
- Диспетчера доступа (Security Reference Monitor, SRM), который проверяет, имеет ли пользователь право на доступ к объекту и на выполнение тех действий, которые он пытается совершить. Этот компонент обеспечивает легализацию доступа и политику *аудита*, определяемые LSA. Он предоставляет услуги для программ супервизорного и пользовательского режимов, для того чтобы гарантировать, что пользователи и процессы, осуществляющие попытки доступа к объекту, имеют необходимые права. Данный компонент также порождает сообщения службы *аудита*, когда это необходимо.

Microsoft Windows NT - относительно новая ОС, которая была спроектирована для поддержки разнообразных защитных механизмов, от минимальных до С2, и безопасность которой наиболее продумана. Дефолтный уровень называется минимальным, но он легко может быть доведен системным администратором до желаемого уровня.

**Заключение**

Решение вопросов безопасности операционных систем обусловлено их архитектурными особенностями и связано с правильной организацией *идентификации* и *аутентификации*, авторизации и *аудита*.

Наиболее простой подход к *аутентификации* - применение пользовательского пароля. Пароли уязвимы, значительная часть попыток несанкционированного доступа в систему связана с компрометацией паролей.

*Авторизация* связана со специфицированием совокупности аппаратных и программных объектов, нуждающихся в защите. Для защиты объекта устанавливаются *права доступа* к нему. Набор прав доступа определяет *домен безопасности*. Формальное описание модели защиты осуществляется с помощью *матрицы доступа*, которая может храниться в виде списков прав доступа или *перечней возможностей*.

*Аудит* системы заключается в регистрации специальных данных о различных событиях, происходящих в системе и так или иначе влияющих на состояние безопасности компьютерной системы.

Среди современных ОС вопросы безопасности лучше всего продуманы в ОС *Windows* NT.

## Предмет и задачи криптографии

Проблемой защиты информации при ее передаче между абонентами люди занимаются на протяжении всей своей истории. Человечеством изобретено множество способов, позволяющих в той или иной мере скрыть смысл передаваемых сообщений от противника. На практике выработалось несколько групп методов защиты секретных посланий. Назовем некоторые из них, применяющиеся так же давно, как и криптографические.

Первым способом является *физическая защита* материального носителя информации от противника. В качестве носителя данных может выступать бумага, компьютерный носитель (DVD-*диск*, флэш-карта, магнитный *диск*, *жесткий диск* компьютера и т.д.). Для реализации этого способа необходим надежный *канал связи*, недоступный для перехвата. В различное время для этого использовались почтовые голуби, специальные курьеры, радиопередачи на секретной частоте. Методы физической защиты информации используются и в современных автоматизированных системах обработки данных. Так, например, комплексные системы защиты информации невозможны без систем ограждения и физической изоляции, а также без охранных систем.

Второй способ защиты информации, известный с давних времен – *стеганографическая защита* информации. Этот способ защиты основан на попытке скрыть от противника сам факт наличия интересующей его информации. При стеганографическом методе защиты от противника прячут физический носитель данных или маскируют секретные сообщения среди открытой, несекретной информации. К таким способам относят, например, "запрятывание" микрофотографии с тайной информацией в несекретном месте: под маркой на почтовом конверте, под обложкой книги и т.д. К стеганографии относятся также такие известные приемы, как "запрятывание" секретного послания в корешках книг, в пуговицах, в каблуках, в пломбе зуба и т.д. Некоторые из методов были разработаны еще в древние времена. Так, например, греки нашли необычное решение: они брили наголо голову раба и выцарапывали на ней свое послание. Когда волосы на голове раба отрастали вновь, его посылали доставить сообщение. Получатель брил голову раба и прочитывал текст. К сожалению, на отправку сообщения и получение ответа таким способом уходило несколько недель.

В более поздние времена в этом направлении наибольшее распространение получили химические (симпатические) чернила. Текст, написанный этими чернилами между строк несекретного сообщения, невидим. Он появлялся только в результате применения определенной технологии проявления.

В условиях повсеместного использования информационных технологий возникают новые стеганографические приемы. Например, известен способ, при котором секретное сообщение прячется в файле графического изображения. При использовании этого способа младший значащий *бит* в описании каждого пикселя изображения заменяется битом сообщения. Разделив все исходное сообщение на биты и разместив эти биты *по* всему графическому файлу, мы пересылаем изображение с замаскированным сообщением получателю. Графическое изображение при этом меняется не слишком сильно, особенно если использовался режим с большим количеством цветов, например, с глубиной цвета 24 бита на *пиксел*. Это связано с тем, что человеческий глаз не может различать такое большое количество цветов. В результате в картинке размером всего 32 на 32 точки можно вместить тайное сообщение длиной 1024 бита или 128 *байт*.

Третий способ защиты информации – наиболее надежный и распространенный в наши дни – *криптографический*. Этот метод защиты информации предполагает преобразование информации для сокрытия ее смысла от противника. **Криптография** в переводе с греческого означает "тайнопись". В настоящее время *криптография* занимается поиском и исследованием математических методов преобразования информации.

Наряду с криптографией развивается и совершенствуется **криптоанализ** – наука о преодолении криптографической защиты информации. Криптоаналитики исследуют возможности расшифровывания информации без знания ключей. Успешно проведенный *криптоанализ* позволяет получить *ключ шифрования*, или *открытый текст*, или то и другое вместе. Иногда криптографию и *криптоанализ* объединяют в одну науку – **криптологию** (kryptos - тайный, logos - наука), занимающуюся вопросами обратимого преобразования информации с целью защиты от несанкционированного доступа, оценкой надежности систем шифрования и анализом стойкости шифров.

В настоящее время *криптография* прочно вошла в нашу жизнь. Перечислим лишь некоторые сферы применения криптографии в современном информатизированном обществе:

- шифрование данных при передаче по открытым каналам связи (например, при совершении покупки в Интернете сведения о сделке, такие как адрес, телефон, номер кредитной карты, обычно зашифровываются в целях безопасности);
- обслуживание банковских пластиковых карт;
- хранение и обработка паролей пользователей в сети;
- сдача бухгалтерских и иных отчетов через удаленные каналы связи;
- банковское обслуживание предприятий через локальную или глобальную сеть;
- безопасное от несанкционированного доступа хранение данных на жестком диске компьютера (в операционной системе Windows даже имеется специальный термин – шифрованная файловая система (EFS)).

До начала XX века криптографические методы применялись лишь для шифрования данных с целью защиты от несанкционированного доступа. В двадцатом веке в связи с развитием техники передачи информации на дальние расстояния интерес к криптографии значительно возрос. Благодаря созданию новых криптографических методов расширился и спектр задач криптографии. В настоящее время считается, что *криптография* предназначена решать следующие задачи:

- собственно шифрование данных с целью защиты от несанкционированного доступа;
- проверка подлинности сообщений: получатель сообщения может проверить его источник;
- проверка целостности передаваемых данных: получатель может проверить, не было ли сообщение изменено или подменено в процессе пересылки;
- обеспечение невозможности отказа, то есть невозможности как для получателя, так и для отправителя отказаться от факта передачи.

Системы шифрования варьируются от самых элементарных до очень сложных. И если первые не требуют никаких математических познаний, то в последних используются понятия, знакомые лишь специалистам в некоторых областях математики и информатики. При использовании криптографических методов должны учитываться *затраты* на защиту информации и на реализацию методов нападения. На практике стремятся к

достижению компромисса между стоимостью шифрования и требуемой степенью обеспечения безопасности.

В рамках данного учебного пособия рассматриваются как простейшие, "докомпьютерные", шифры, известные человечеству на протяжении веков, так и современные системы шифрования, разработанные только в XXI веке.

**Основные определения**

Теперь, узнав назначение криптографии, познакомимся с основными терминами, которые будем использовать при изучении криптографических методов защиты информации.

**Шифр** – совокупность заранее оговоренных способов преобразования исходного секретного сообщения с целью его защиты.

Исходные сообщения обычно называют **открытыми текстами**. В иностранной литературе для открытого текста используют термин **plaintext**.

**Символ** - это любой знак, в том числе буква, цифра или знак препинания.

**Алфавит** - конечное множество используемых для кодирования информации символов. Например, русский *алфавит* содержит 33 буквы от А до Я . Однако этих тридцати трех знаков обычно бывает недостаточно для записи сообщений, поэтому их дополняют символом пробела, точкой, запятой и другими знаками. *Алфавит* арабских цифр – это символы 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 . Этот *алфавит* содержит 10 знаков и с его помощью можно записать любое *натуральное число*. Любое сообщение может быть записано также с помощью *двоичного алфавита* , то есть с использованием только нулей и единиц.

Сообщение, полученное после преобразования с использованием любого шифра, называется **шифрованным сообщением** (закрытым текстом, криптограммой). В иностранной литературе для закрытого текста используют термин **ciphertext**.

Преобразование открытого текста в криптограмму называется **зашифрованием**. Обратное действие называется **расшифрованием**. В англоязычной литературе терминам "зашифрование/ *расшифрование*" соответствуют термины **"enciphering/deciphering"**.

**Ключ** – *информация*, необходимая для шифрования и расшифрования сообщений.

С точки зрения русского языка термины "*расшифрование*" и "*дешифрование*" являются синонимами. Однако в работах *по* криптографии последних десятилетий часто эти слова различают. Будем считать, что термины "*расшифрование*" и "*дешифрование*" не являются синонимами. Примем, что *расшифрованием* занимается легальный *получатель сообщения* (тот, кто знает *ключ*), а человек, которому послание не предназначено, пытаясь понять его смысл, занимается *дешифрованием* .

**Система шифрования**, или **шифрсистема**, – это любая система, которую можно использовать для обратимого изменения текста сообщения с целью сделать его непонятным для всех, кроме тех, кому оно предназначено.

**Криптостойкостью** называется характеристика шифра, определяющая его стойкость к дешифрованию без знания ключа (т.е. способность противостоять криптоанализу).

Таким образом, с учетом всех сделанных определений можно дать более точное *определение* науке "*криптография*". **Криптография** изучает построение и использование систем шифрования, в том числе их стойкость, слабости и степень уязвимости относительно различных методов вскрытия.

Все методы преобразования информации с целью защиты от несанкционированного доступа делятся на две большие группы: методы шифрования *с закрытым ключом* и методы шифрования *с открытым ключом* . **Шифрование с закрытым ключом** (*шифрование* с секретным ключом или *симметричное шифрование*) используется человеком уже довольно долгое время. Для шифрования и расшифрования данных в этих методах используется один и тот же *ключ*, который обе стороны стараются хранить в секрете от противника. Системы шифрования с закрытым ключом подробно рассматриваются в лекциях 2-9. **Шифрование с открытым ключом** (*асимметричное шифрование*) стало использоваться для криптографического закрытия информации лишь во второй половине XX века. В эту группу относятся методы шифрования, в которых для шифрования и расшифрования данных используются два разных ключа. При этом один из ключей (открытый *ключ*) может передаваться *по* открытому (незащищенному) каналу связи. Алгоритмам преобразования информации с открытым ключом посвящены лекции 10-14 учебного пособия.

**Электронной (цифровой) подписью** называется обычно присоединяемый к сообщению *блок данных*, полученный с использованием криптографического преобразования. *Электронная подпись* позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.

**Криптографическая система защиты информации** – система защиты информации, в которой используются криптографические методы для шифрования данных.

**Требования к криптографическим системам защиты информации**

Для разрабатываемых в настоящее время криптографических систем защиты информации сформулированы следующие общепринятые требования:

- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- знание алгоритма шифрования не должно влиять на надежность защиты;
- любой ключ из множества возможных должен обеспечивать надежную защиту информации;
- алгоритм шифрования должен допускать как программную, так и аппаратную реализацию.

Не для всех алгоритмов шифрования перечисленные требования выполняются полностью. В частности, требование отсутствия слабых ключей (ключей, которые позволяют злоумышленнику легче вскрыть зашифрованное сообщение) не выполняется для некоторых "старых" блочных шифров. Однако все вновь разрабатываемые системы шифрования удовлетворяют перечисленным требованиям.

Если все процедуры шифрования и расшифрования выполняются специальными электронными схемами *по* определенным логическим правилам, то такой способ реализации криптографического метода называется **аппаратным**. Аппаратным способом могут быть реализованы все криптоалгоритмы, рассматриваемые в данном учебном пособии. На разработку аппаратного устройства необходимы существенные *затраты*, однако при массовом выпуске устройства эти *затраты* окупаются. Аппаратная реализация криптографического метода отличается высокой производительностью, простотой в эксплуатации, защищенностью. Во всем мире выпускаются ежегодно миллионы криптографических устройств.

Повсеместное внедрение вычислительной техники, а особенно персональных компьютеров, привело к появлению **программных** реализаций алгоритмов шифрования. Интересно, что разработчики первых блочных шифров, используемых, например, в старом американском стандарте *DES*, и не предполагали, что придуманные ими алгоритмы будут реализовываться программно. Благодаря тому, что все методы криптографического преобразования могут быть представлены в виде конечной алгоритмической процедуры, они могут быть запрограммированы. Основным достоинством программных методов реализации защиты является их гибкость, т.е. возможность быстрого изменения алгоритмов шифрования или их настройки. Кроме того, программные реализации криптографических методов отличаются меньшей стоимостью. Основным же недостатком программной реализации является существенно меньшее *быстродействие по* сравнению с аппаратными средствами (в десятки раз в зависимости от алгоритма).

В настоящее время выпускаются и комбинированные модули шифрования, так называемые программно-*аппаратные средства*. В этом случае *компьютер* дополняется своеобразным "криптографическим сопроцессором" – аппаратным вычислительным блоком, ориентированным на выполнение специфических *криптографических операций*. Меняя *программное обеспечение* для такого устройства, можно выбирать тот или иной метод шифрования. Такое программно-аппаратное средство объединяет в себе достоинства программных и аппаратных методов.

**Криптографический протокол**

В современной криптографии большое внимание уделяется не только созданию и исследованию шифров, но и разработке криптографических протоколов. **Криптографический протокол** – это такая процедура взаимодействия двух или более абонентов с использованием криптографических средств, в результате которой абоненты достигают своей цели, а их противники - не достигают. В основе протокола лежит набор правил, регламентирующих использование криптографических преобразований и алгоритмов в информационных процессах. Каждый криптографический протокол предназначен для решения определенной задачи.

Любой протокол имеет следующие свойства:

- при выполнении протокола важен порядок действий; каждое действие должно выполняться в свою очередь и только по окончании предыдущего;
- протокол должен быть непротиворечивым;
- протокол должен быть полным, то есть для каждой возможной ситуации должно быть предусмотрено соответствующее действие.

Свойства протокола напоминают известные нам из курса информатики свойства алгоритма. На самом деле протокол – это и есть *алгоритм* действия нескольких сторон в определенной ситуации. Участники протокола должны знать протокол и выполнять полностью все его этапы. Участниками криптографического протокола обычно являются абоненты некоторой системы связи. Участники протокола могут не доверять друг другу, поэтому криптографические протоколы должны защищать их участников не только от внешнего противника, но и от нечестных действий партнеров.

Криптографические протоколы – сравнительно молодая отрасль криптографической науки. Первые протоколы появились во второй половине XX века. С тех пор эта область криптографии бурно развивалась, и на настоящий момент имеется уже несколько десятков различных типов криптографических протоколов. Все эти типы можно условно

разделить на две группы: прикладные протоколы и примитивные. *Прикладной протокол* решает конкретную задачу, которая возникает (или может возникнуть) на практике. *Примитивные* же протоколы используются как своеобразные "строительные блоки" при разработке прикладных протоколов. Мы в данном учебном пособии будем рассматривать только примитивные криптографические протоколы, которые при некоторой адаптации к реальным системам связи могут использоваться на практике.

Рассмотрим назначение некоторых видов протоколов.

1. *Протоколы конфиденциальной передачи сообщений*. Задача конфиденциальной передачи сообщений состоит в следующем. Имеются два участника протокола, которые являются абонентами сети связи. Участники соединены некоторой линией связи, по которой можно пересылать сообщения в обе стороны. Линию связи может контролировать противник. У одного из абонентов имеется конфиденциальное сообщение m, и задача состоит в том, чтобы это сообщение конфиденциальным же образом передать второму абоненту. Протоколы этого типа, наверно, появились раньше других криптографических протоколов, так как задача конфиденциальной передачи сообщений – исторически первая задача, которая решалась криптографией.
2. *Протоколы аутентификации и идентификации*. Они предназначены для предотвращения доступа к некоторой информации лиц, не являющихся ее пользователями, а также предотвращения доступа пользователей к тем ресурсам, на которые у них нет полномочий. Типичная сфера применения – организация доступа пользователей к ресурсам некоторой большой информационной системы.
3. *Протоколы распределения ключей* необходимы для обеспечения секретными ключами участников обмена зашифрованными сообщениями.
4. *Протоколы электронной цифровой подписи* позволяют ставить под электронными документами подпись, аналогичную обыкновенной подписи на бумажных документах. В результате выполнения протокола электронной цифровой подписи к передаваемой информации добавляется уникальное числовое дополнение, позволяющее проверить ее авторство.
5. *Протоколы обеспечения неотслеживаемости* ("Электронные деньги"). Под электронными деньгами в криптографии понимают электронные платежные средства, обеспечивающие неотслеживаемость, то есть невозможность проследить источник пересылки информации.

Рассмотрим простейший протокол для обмена конфиденциальными сообщениями между двумя сторонами, которые будем называть *абонент* №1 и *абонент* №2. Пусть *абонент* №1 желает передать зашифрованное сообщение абоненту №2. В этом случае их последовательность действий должна быть следующей.

1. Абоненты выбирают систему шифрования (например, шифр Цезаря со сдвигом на n позиций).
2. Абоненты договариваются о ключе шифрования.
3. Абонент №1 шифрует исходное сообщение с помощью ключа выбранным методом и получает зашифрованное сообщение.
4. Зашифрованное сообщение пересылается абоненту №2.
5. Абонент №2 расшифровывает зашифрованное сообщение с помощью ключа и получает *открытое сообщение*.

Этот протокол достаточно прост, однако он может действительно использоваться на практике. Криптографические протоколы могут быть простыми и сложными в зависимости от назначения. Со многими из них мы познакомимся в различных главах этого курса.

Ранее мы ввели понятие криптографической атаки и рассмотрели типы атак на криптографический *алгоритм*. Во многих случаях *атака* может быть направлена не на *алгоритм* шифрования, а на протокол. Поэтому даже наличие абсолютно надежного алгоритма шифрования не гарантирует полной безопасности абонентам системы связи. Известны случаи, когда применяемые на практике криптографические протоколы содержали изъяны, допускающие мошенничество участвующих сторон или вскрытие со стороны активного взломщика. Конечно, криптографические протоколы не должны допускать таких возможностей нарушителям. Именно поэтому в настоящее время криптографические протоколы являются предметом тщательного анализа со стороны специалистов.

**Ключевые термины**

**Ciphertext** – зашифрованное сообщение (закрытый текст, криптограмма).

**Deciphering** – *расшифрование*.

**Enciphering** – преобразование открытого текста в криптограмму (зашифрование).

**Plaintext** – исходное сообщение или *открытый текст*.

**Активная криптографическая атака** – при такой атаке противник имеет возможность модифицировать передаваемые сообщения и даже добавлять свои сообщения.

**Алфавит** - конечное множество используемых для кодирования информации символов.

**Ключ** – *информация*, необходимая для шифрования и расшифрования сообщений.

**Криптоанализ** – наука о преодолении криптографической защиты информации.

**Криптографическая система защиты информации** – система защиты информации, в которой используются криптографические методы для шифрования данных.

**Криптографический протокол** – *алгоритм* взаимодействия двух или более абонентов с использованием криптографических средств, в результате которой абоненты достигают своей цели, а их противники - не достигают.

**Криптография** изучает построение и использование систем шифрования, в том числе их стойкость, слабости и степень уязвимости относительно различных методов вскрытия.

**Криптостойкость** – характеристика шифра, определяющая его стойкость к дешифрованию без знания ключа (т.е. способность противостоять криптоанализу).

**Пассивная криптографическая атака** – *атака*, при которой противник не имеет возможности изменять передаваемые сообщения. При *пассивной атаке* возможно лишь прослушивание передаваемых сообщений, их *дешифрование* и *анализ трафика*.

**Принцип Керкхоффса** – правило разработки криптографических систем, согласно которому в засекреченном виде держится *ключ шифрования*, а остальные параметры системы шифрования могут быть открыты без снижения *стойкости алгоритма*. Другими словами, при оценке надёжности шифрования необходимо предполагать, что противник знает об используемой системе шифрования всё, кроме применяемых ключей. Впервые данный принцип сформулировал в XIX веке голландский криптограф Огюст Керкхоффс.

**Символ** - это любой знак, в том числе буква, цифра или знак препинания.

**Система шифрования, или шифрсистема**, – это любая система, которую можно использовать для обратимого изменения текста сообщения с целью сделать его непонятным для всех, кроме тех, кому оно предназначено.

**Шифр** – совокупность заранее оговоренных способов преобразования исходного секретного сообщения с целью его защиты.

**Шифрование с закрытым ключом (симметричное шифрование)** – методы обратимого преобразования данных, в которых используется один и тот же *ключ*, который обе стороны информационного обмена должны хранить в секрете от противника. Все известные из истории шифры, например, *шифр* Цезаря – это шифры с закрытым ключом.

**Шифрование с открытым ключом (асимметричное шифрование)** – методы шифрования, в которых для шифрования и расшифрования данных используются два разных ключа. При этом один из ключей (открытый *ключ*) может передаваться *по* открытому (незащищенному) каналу связи. *Шифрование* с открытым ключом используется на практике лишь со второй половины XX века.

**Электронная (цифровая) подпись** – присоединяемый к сообщению *блок данных*, полученный с использованием криптографического преобразования. *Электронная подпись* позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.

**Общая схема симметричного шифрования**

Классическая, или одноключевая *криптография* опирается на использование **симметричных алгоритмов шифрования**, в которых *шифрование* и *расшифрование* отличаются только порядком выполнения и направлением некоторых шагов. Эти алгоритмы используют один и тот же секретный элемент (*ключ*), и второе действие (*расшифрование*) является простым обращением первого (шифрования). Поэтому обычно каждый из участников обмена может как зашифровать, так и расшифровать сообщение. Схематичная структура такой системы представлена на рис. 2.1.

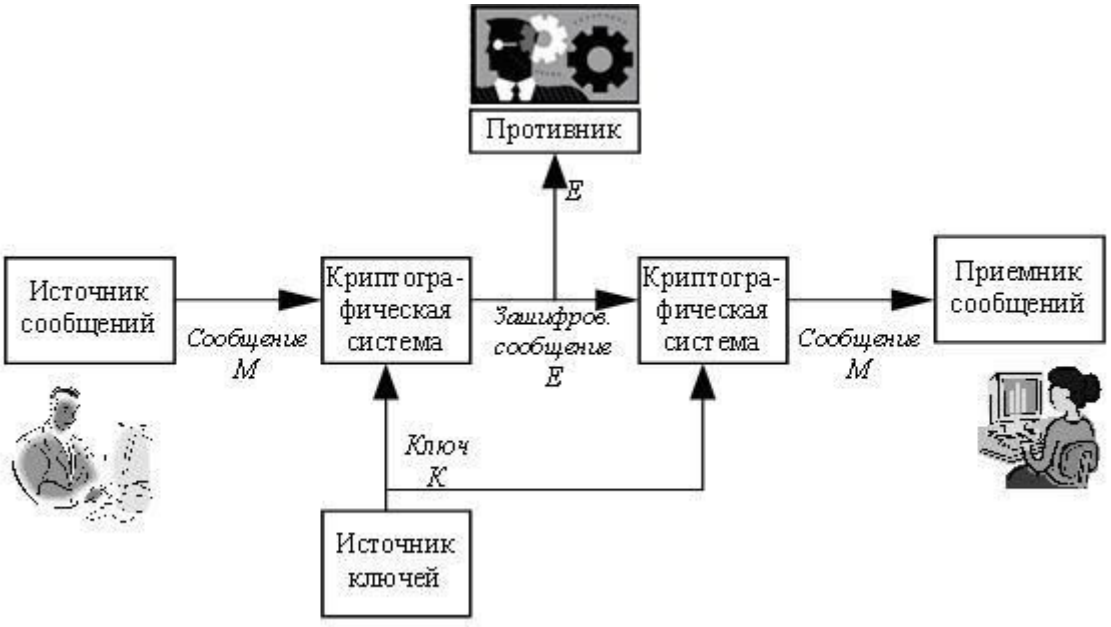


**Рис. 2.1.** Общая структура секретной системы, использующей симметричное шифрование

На передающей стороне имеются источник сообщений и источник ключей. Источник ключей выбирает конкретный *ключ* К среди всех возможных ключей данной системы. Этот *ключ* К передается некоторым способом принимающей стороне, причем предполагается, что его нельзя перехватить, например, *ключ* передается специальным курьером (поэтому *симметричное шифрование* называется также шифрованием с *закрытым ключом*). Источник сообщений формирует некоторое сообщение М, которое затем шифруется с использованием выбранного ключа. В результате процедуры шифрования получается зашифрованное сообщение Е (называемое также криптограммой). Далее криптограмма Е передается *по* каналу связи. Так как *канал связи* является открытым, незащищенным, например, *радиоканал* или компьютерная *сеть*, то передаваемое сообщение может быть перехвачено противником. На принимающей стороне криптограмму Е с помощью ключа расшифровывают и получают исходное сообщение М.

Если М – сообщение, К – *ключ*, а Е – зашифрованное сообщение, то можно записать

$E=f(M,K)$

то есть зашифрованное сообщение Е является некоторой функцией от исходного сообщения М и ключа К. Используемый в криптографической системе метод или *алгоритм* шифрования и определяет функцию f в приведенной выше формуле.

*По* причине большой избыточности естественных языков непосредственно в зашифрованное сообщение чрезвычайно трудно внести осмысленное изменение, поэтому классическая *криптография* обеспечивает также защиту от навязывания ложных данных. Если же естественной избыточности оказывается недостаточно для надежной защиты сообщения от модификации, *избыточность* может быть искусственно увеличена путем добавления к сообщению специальной контрольной комбинации, называемой *имитовставкой*.

Известны разные методы шифрования с закрытым ключом рис. 2.2. На практике часто используются алгоритмы перестановки, подстановки, а также комбинированные методы.

**Рис. 2.2.** Методы шифрования с закрытым ключом

В методах перестановки символы исходного текста меняются местами друг с другом *по* определенному правилу. В методах замены (или подстановки) символы открытого текста заменяются некоторыми эквивалентами шифрованного текста. С целью повышения надежности шифрования, текст, зашифрованный с помощью одного метода, может быть еще раз зашифрован с помощью другого метода. В этом случае получается комбинированный или композиционный *шифр*. Применяемые на практике в настоящее время блочные или поточные симметричные шифры также относятся к комбинированным, так как в них используется несколько операций для зашифрования сообщения.

Основное отличие современной криптографии от криптографии "докомпьютерной" заключается в том, что раньше криптографические алгоритмы оперировали символами естественных языков, например, буквами английского или русского алфавитов. Эти буквы переставлялись или заменялись другими *по* определенному правилу. В современных криптографических алгоритмах используются *операции* над двоичными знаками, то есть над нулями и единицами. В настоящее время основными операциями при шифровании также являются *перестановка* или *подстановка*, причем для повышения надежности шифрования эти *операции* применяются вместе (комбинируются) и помногу раз циклически повторяются.

Принципы построения современных блочных шифров сформулированы в "Принципы построения блочных шифров с закрытым ключом" , "Алгоритмы шифрования DES и AES" , "Алгоритм криптографического преобразования данных ГОСТ 28147-89" , а в этой лекции рассматриваются шифры подстановки и перестановки, применяемые человеком с древнейших времен. Мы должны познакомиться с этими шифрами, так как процедуры подстановки и перестановки используются в качестве составных операций и в современных блочных шифрах.

**Методы замены**

Методы шифрования заменой (подстановкой) основаны на том, что символы исходного текста, обычно разделенные на блоки и записанные в одном алфавите, заменяются одним или несколькими символами другого алфавита в соответствии с принятым правилом преобразования.

**Одноалфавитная замена**

Одним из важных подклассов методов замены являются *одноалфавитные* (или моноалфавитные) подстановки, в которых устанавливается однозначное соответствие между каждым знаком $a_i$ исходного алфавита сообщений A и соответствующим знаком $e_i$ зашифрованного текста E. Одноалфавитная подстановка иногда называется также простой заменой, так как является самым простым шифром замены.

Примером одноалфавитной замены является шифр Цезаря, рассмотренный ранее. В рассмотренном в "Основные понятия криптографии" примере первая строка является исходным алфавитом, вторая (с циклическим сдвигом на k влево) – вектором замен.

В общем случае при одноалфавитной подстановке происходит однозначная замена исходных символов их эквивалентами из вектора замен (или таблицы замен). При таком методе шифрования ключом является используемая таблица замен.

Подстановка может быть задана с помощью таблицы, например, как показано на рис. 2.3.

| Откр. текст | Шифр 1 | Шифр 2 | | Откр. текст | Шифр 1 | Шифр 2 | | Откр. текст | Шифр 1 | Шифр 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| А | В | ^ | | М | Т | № | | Ч | М | ∑ |
| Б | И | @ | | Н | Ц | # | | Ш | У | ∇ |
| В | О | ) | | О | . | - | | Щ | Д | ϒ |
| Г | А | + | | П | Ж | = | | Ъ | Э | ℵ |
| Д | Щ | < | | Р | Г | ( | | Ы | Н | ⊕ |
| Е | П | > | | С | Л | ? | | Ь | Ю | × |
| Ж | К | ∀ | | Т | Х | % | | Э | Ы | ω |
| З | Б | ♦ | | У | С | ⊗ | | Ю | Ш | $ |
| И | Ъ | * | | Ф | Ь | ! | | Я | Е | Δ |
| К | пробел | ♥ | | Х | Ч | № | | пробел | Ф | ∞ |
| Л | Р | ♠ | | Ц | З | ® | | . | Я | ♣ |

**Рис. 2.3.** Пример таблицы замен для двух шифров

В таблице на рис. 2.3 на самом деле объединены сразу две таблицы. Одна (шифр 1) определяет замену русских букв исходного текста на другие русские буквы, а вторая (шифр 2) – замену букв на специальные символы. Исходным алфавитом для обоих шифров будут заглавные русские буквы (за исключением букв "Ё" и "Й"), пробел и точка.

Зашифрованное сообщение с использованием любого шифра моноалфавитной подстановки получается следующим образом. Берется очередной знак из исходного сообщения. Определяется его позиция в столбце "Откр. текст" таблицы замен. В зашифрованное сообщение вставляется шифрованный символ из этой же строки таблицы замен.

Попробуем зашифровать сообщение "ВЫШЛИТЕ ПОДКРЕПЛЕНИЕ" с использованием этих двух шифров ( рис. 2.4). Для этого берем первую букву исходного сообщения "В". В таблице на рис. 2.3 в столбце "Шифр 1" находим для буквы "В" заменяемый символ. Это будет буква "О". Записываем букву "О" под буквой "В". Затем рассматриваем второй символ исходного сообщения – букву "Ы". Находим эту букву в столбце "Откр. текст" и из столбца "Шифр 1" берем букву, стоящую на той же строке, что и буква "Ы". Таким образом получаем второй символ зашифрованного сообщения – букву "Н". Продолжая действовать аналогично, зашифровываем все исходное сообщение ( рис. 2.4).

| Открытое сообщение | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| В | Ы | Ш | Л | И | Т | Е | | П | О | Д | К | Р | Е | П | Л | Е | Н | И | Е |
| Зашифрованное сообщение с использованием шифра 1 | | | | | | | | | | | | | | | | | | | |
| О | Н | У | Р | Ъ | Х | П | Ф | Ж | . | Щ | | Г | П | Ж | Р | П | Ц | Ъ | П |
| Зашифрованное сообщение с использованием шифра 2 | | | | | | | | | | | | | | | | | | | |
| ) | ⊕ | ∇ | ♠ | * | % | > | ∞ | = | - | < | ♥ | ( | > | = | ♠ | > | # | * | > |

**Рис. 2.4.** Пример шифрования методом прямой замены

Полученный таким образом текст имеет сравнительно низкий уровень защиты, так как исходный и зашифрованный тексты имеют одинаковые статистические закономерности. При этом не имеет значения, какие символы использованы для замены – перемешанные символы исходного алфавита или таинственно выглядящие знаки.

Зашифрованное сообщение может быть вскрыто путем так называемого *частотного криптоанализа*. Для этого могут быть использованы некоторые статистические данные языка, на котором написано сообщение.

Известно, что в текстах на русском языке наиболее часто встречаются символы О, И. Немного реже встречаются буквы Е, А. Из согласных самые частые символы Т, Н, Р, С. В распоряжении криптоаналитиков имеются специальные таблицы частот встречаемости символов для текстов разных типов – научных, художественных и т.д.

Криптоаналитик внимательно изучает полученную криптограмму, подсчитывая при этом, какие символы сколько раз встретились. Вначале наиболее часто встречаемые знаки зашифрованного сообщения заменяются, например, буквами О. Далее производится попытка определить места для букв И, Е, А. Затем подставляются наиболее часто встречаемые согласные. На каждом этапе оценивается возможность "сочетания" тех или иных букв. Например, в русских словах трудно найти четыре подряд гласные буквы, слова в русском языке не начинаются с буквы Ы и т.д. На самом деле для каждого естественного языка (русского, английского и т.д.) существует множество закономерностей, которые помогают раскрыть специалисту зашифрованные противником сообщения.

**Методы гаммирования**

Еще одним частным случаем многоалфавитной подстановки является **гаммирование**. В этом способе шифрование выполняется путем сложения символов исходного текста и ключа по модулю, равному числу букв в алфавите. Если в исходном алфавите, например, 33 символа, то сложение производится по модулю 33. Такой процесс сложения исходного текста и ключа называется в криптографии *наложением гаммы*.

Пусть символам исходного алфавита соответствуют числа от 0 (А) до 32 (Я). Если обозначить число, соответствующее исходному символу, x, а символу ключа – k, то можно записать правило гаммирования следующим образом:

z = x + k (mod N),

где z – закодированный символ, N - количество символов в алфавите, а сложение по модулю N - операция, аналогичная обычному сложению, с тем отличием, что если обычное суммирование дает результат, больший или равный N, то значением суммы считается остаток от деления его на N. Например, пусть сложим по модулю 33 символы Г (3) и Ю (31):

3 + 31 (mod 33) = 1,

то есть в результате получаем символ Б, соответствующий числу 1.

Наиболее часто на практике встречается двоичное гаммирование. При этом используется *двоичный алфавит*, а сложение производится по модулю два. Операция сложения по модулю 2 часто обозначается $\oplus$, то есть можно записать:

$$z = x + k(mod 2) = x \oplus k.$$

Операция сложения по модулю два в *алгебре логики* называется также "исключающее ИЛИ" или по-английски XOR.

Чем длиннее ключ, тем надежнее шифрование методом гаммирования. Связь длины ключа с вероятностью вскрытия сообщения, а также некоторые принципы дешифрования сообщений, *закрытых методом* гаммирования, обсуждаются в "Поточные шифры и генераторы псевдослучайных чисел. Часть 2" и "Шифрование, помехоустойчивое кодирование и сжатие информации". На практике длина ключа ограничена возможностями аппаратуры обмена данными и вычислительной техники, а именно выделяемыми объемами памяти под ключ, временем обработки сообщения, а также возможностями аппаратуры подготовки и записи последовательностей ключей. Кроме того, для использования ключа вначале необходимо каким-либо надежным способом доставить его обеим сторонам, обменивающимся сообщениями. Это приводит к возникновению проблемы распределения ключей, сложность решения которой возрастает с увеличением длины ключа и количества абонентов в сети передачи сообщений.

**Методы перестановки**

При использовании шифров **перестановки** *входной поток* исходного текста делится на блоки, в каждом из которых выполняется *перестановка* символов. Перестановки в классической "докомпьютерной" криптографии получались в результате записи исходного текста и чтения шифрованного текста *по* разным путям геометрической фигуры.

Простейшим примером перестановки является *перестановка с фиксированным периодом d*. В этом методе сообщение делится на блоки *по d* символов и в каждом блоке производится одна и та же *перестановка*. Правило, *по* которому производится *перестановка*, является ключом и может быть задано некоторой перестановкой первых *d* натуральных чисел. В результате сами буквы сообщения не изменяются, но передаются в другом порядке.

Например, для d=6 в качестве ключа перестановки можно взять 436215. Это означает, что в каждом блоке из 6 символов четвертый символ становится на первое *место*, третий – на второе, шестой – на третье и т.д. Пусть необходимо зашифровать такой текст:

ЭТО_ТЕКСТ_ДЛЯ_ШИФРОВАНИЯ

Количество символов в исходном сообщении равно 24, следовательно, сообщение необходимо разбить на 4 блока. Результатом шифрования с помощью перестановки 436215 будет сообщение

_ОЕТЭТ_ТЛСКДИШР_ЯФНАЯВОИ

Теоретически, если блок состоит из *d* символов, то число возможных перестановок d!=1*2*...*(d-1)*d. В последнем примере d=6, следовательно, число перестановок равно 6!=1*2*3*4*5*6=720. Таким образом, если противник перехватил зашифрованное сообщение из рассмотренного примера, ему понадобится не более 720 попыток для раскрытия исходного сообщения (при условии, что размер блока известен противнику).

Для повышения криптостойкости можно последовательно применить к шифруемому сообщению две или более перестановки с разными периодами.

Другим примером методов перестановки является *перестановка по таблице*. В этом методе производится *запись* исходного текста *по* строкам некоторой таблицы и чтение его *по* столбцам этой же таблицы. Последовательность заполнения строк и чтения столбцов может быть любой и задается ключом.

**Краткие итоги**

Симметричные шифры – способ шифрования, в котором для шифрования и расшифровывания применяется один и тот же криптографический *ключ*. *Ключ* шифрования должен сохраняться в секрете обеими сторонами.

Известны разные методы шифрования с закрытым ключом. На практике часто используются алгоритмы перестановки, подстановки, а также комбинированные методы.

В методах перестановки символы исходного текста меняются местами друг с другом *по* определенному правилу.

В методах замены (или подстановки) символы открытого текста заменяются некоторыми эквивалентами шифрованного текста. *Шифр* простой (или одноалфавитной) замены – *группа* методов шифрования, которые сводится к созданию *по* определённому алгоритму таблицы шифрования, в которой для каждой буквы открытого текста существует единственная сопоставленная ей буква *шифртекста*. Само *шифрование* заключается в замене букв согласно таблице. Для расшифровки достаточно иметь ту же таблицу, либо знать *алгоритм*, *по* которой она генерируется.

*Шифр* многоалфавитной замены – *группа* методов шифрования подстановкой, в которых для замены символов исходного текста используется не один, а несколько алфавитов *по* определенному правилу. Таким образом, при шифровании получаётся достаточно сложная последовательность, которую уже не так просто вскрыть, как один одноалфавитный *шифр*.

Частным случаем многоалфавитной подстановки является гаммирование – метод шифрования, основанный на "наложении" гамма-последовательности на *открытый текст*. Обычно это суммирование в каком-либо конечном *поле* (суммирование *по* модулю длины алфавита).

Самым важным эффектом, достигаемым при использовании многоалфавитного шифра, является маскировка частот появления тех или иных букв в тексте, на основании которой обычно очень легко вскрываются одноалфавитные шифры.

**Предпосылки создания методов шифрования с открытым ключом и основные определения**

При использовании шифрования с закрытым ключом возникают две достаточно серьезные проблемы. Первая проблема заключается в изготовлении секретных ключей и доставке их участникам информационного обмена. При большом количестве и территориальной распределенности участников информационного обмена, использующих каналы связи общего назначения, например, обычную или электронную почту, часто бывает сложно гарантировать *безопасность* доставки такого ключа и его подлинность. В "Поточные шифры и генераторы псевдослучайных чисел. Часть 2" учебного пособия проблема распределения ключей для симметричного шифрования была подробно рассмотрена.

Второй проблемой является обеспечение подлинности партнеров при электронном общении. Развитие деловой переписки и электронной коммерции требует наличия методов, при использовании которых невозможно было бы подменить кого-либо из участников обмена. Получатель корреспонденции должен иметь возможность удостовериться в подлинности документа, а создатель электронного послания должен быть в состоянии доказать свое авторство получателю или третьей стороне. Следовательно, электронные документы должны иметь аналог обычной подписи.

Многие криптографы работали над решением этих проблем, в результате чего во второй половине семидесятых годов XX века были разработаны принципиально новые подходы, позволяющие решить перечисленные выше (и некоторые другие) задачи. Основой послужило открытие так называемых **асимметричных криптоалгоритмов**, или методов, в которых процедуры прямого и обратного криптопреобразования выполняются на различных ключах и не имеют между собой очевидных и легко прослеживаемых связей, которые позволили бы по одному ключу определить другой. *Асимметричные алгоритмы* гораздо больше основаны на свойствах математических функций, чем алгоритмы симметричного шифрования, использующие в основном только *операции* перестановки и замены. Большой вклад в эти исследования внесли американские ученые У. Диффи (W. Diffie), Э. Хеллман (M. Hellman), Р. Меркль (R. Merkle). Они первыми предложили способы решения обеих задач, которые радикально отличаются от всех предыдущих подходов к шифрованию.

*Асимметричные алгоритмы* шифрования называются также **алгоритмами с открытым ключом**. В отличие от *алгоритмов симметричного шифрования* (алгоритмов шифрования с закрытым ключом), в которых для шифрования и расшифрования используется один и тот же *ключ*, в ассиметричных алгоритмах один *ключ* используется для шифрования, а другой, отличный от первого, – для расшифрования. Алгоритмы называются асимметричными, так как ключи шифрования и расшифрования разные, следовательно, отсутствует *симметрия* основных криптографических процессов. Один из двух ключей является **открытым** (*public key*) и может быть объявлен всем, а второй – **закрытым** (*private key*) и должен держаться в секрете. Какой из ключей, открытый или закрытый, используется для шифрования, а какой для расшифрования, определяется назначением криптографической системы.

В настоящее время *асимметричные алгоритмы* широко применяются на практике, например, для обеспечения информационной безопасности телекоммуникационных сетей, в том числе сетей, имеющих сложную топологию; для обеспечения информационной безопасности в глобальной сети *Internet*; в различных банковских и платежных системах (в том числе использующих *интеллектуальные карты*).

*Алгоритмы шифрования* с открытым ключом можно использовать для решения, как *минимум*, трех задач:

1. Для шифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа.
2. Для формирования цифровой подписи под электронными документами.
3. Для распределения секретных ключей, используемых потом при шифровании документов симметричными методами.

**Односторонние функции**

Все *алгоритмы шифрования* с открытым ключом основаны на использовании так называемых *односторонних функций*. **Односторонней функцией** (*one-way function*) называется *математическая функция*, которую относительно легко вычислить, но трудно найти по значению функции соответствующее *значение* аргумента. То есть, зная x легко вычислить f(x), но по известному f(x) трудно найти подходящее *значение* x. Под словом "трудно вычислить" понимают, что для этого потребуется не один год расчетов с использованием ЭВМ. *Односторонние функции* применяются в криптографии также в качестве хеш-функций (см. "Криптографические хеш-функции" ). Использовать *односторонние функции* для шифрования сообщений с целью их защиты не имеет смысла, так как обратно расшифровать зашифрованное сообщение уже не

получится. Для целей шифрования используются специальные *односторонние функции* – **односторонние функции с люком** (или с секретом) – это особый вид *односторонних функций*, имеющих некоторый секрет (*люк*), позволяющий относительно быстро вычислить обратное *значение* функции.

Для *односторонней функции* с люком f справедливы следующие утверждения:

1. зная x, легко вычислить f(x),
2. по известному значению f(x) трудно найти x,
3. зная дополнительно некоторую секретную информацию, можно легко вычислить x.

**Использование асимметричных алгоритмов для шифрования**

В 70-х годах XX века Диффи и Хеллман предложили принцип шифрования, основанный на использовании двух разных ключей, хотя и связанных между собой, но устроенных так, что вычислить по одному из них (открытому) другой (закрытый) практически невозможно. Этот принцип может быть использован для решения проблемы снабжения пользователей ключами шифрования/расшифрования, а точнее – для устранения этой проблемы. Согласно Диффи и Хеллману предварительно распределяемые закрытые ключи вообще не должны использоваться для шифрования данных (так как секрет, который известен более чем одному человеку, – уже не секрет). *Закрытый ключ* должен быть известен только одному лицу – его владельцу. Такой принцип использования *асимметричных алгоритмов* получил название *открытого шифрования* или *шифрованием с открытым ключом*.

Согласно этому принципу, любой желающий может зашифровать сообщение открытым ключом. Расшифровать сообщение сможет только владелец закрытого ключа. Пусть, например, пользователи А и Б, имеющие возможность обмениваться электронными сообщениями, используют схему открытого шифрования. Предположим, *пользователь* А должен передать секретное сообщение пользователю Б так, чтобы никто другой не смог его прочитать. Для этого необходимо выполнить следующие действия:

1. Пользователь Б посылает пользователю А свой открытый ключ U по любому каналу связи, например, по электронной почте.
2. Пользователь А шифрует свое сообщение M полученным открытым ключом U и получает зашифрованное сообщение C.
3. Зашифрованное сообщение C пересылается пользователю Б.
4. Пользователь Б расшифровывает полученное сообщение C своим закрытым ключом R.

Если операцию шифрования обозначить как F, а операцию расшифрования как $F^{-1}$, то схему протокола обмена информацией между пользователями можно изобразить, как на рис. 9.1.

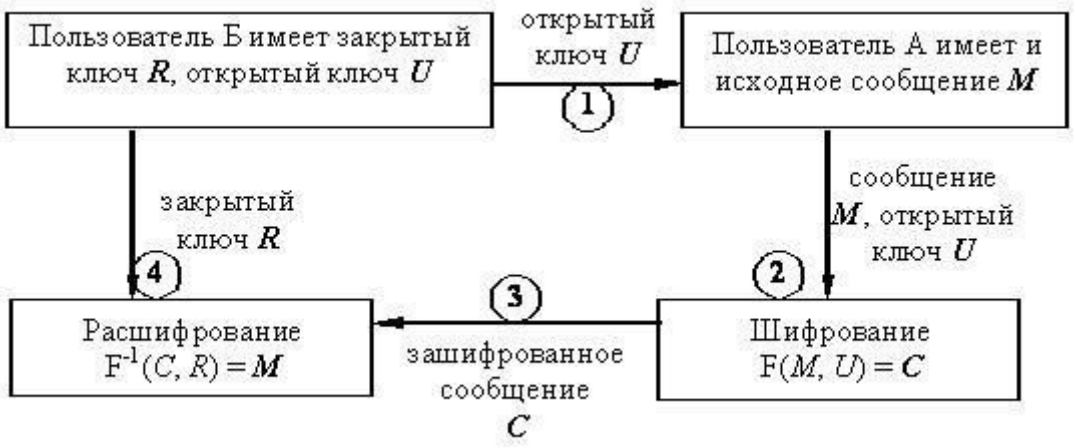


**Рис. 9.1.** Схема открытого шифрования

Использование открытого шифрования снимает проблему распределения ключей. Раньше пользователи перед обменом зашифрованными данными должны были каким-либо образом по закрытому каналу связи согласовывать используемый *секретный ключ*. Для этого они могли встретиться лично или использовать курьера. Если один из пользователей считал нужный изменить *ключ*, он должен был передать новый *ключ* своему абоненту. *Криптография* с открытыми ключами все упрощает. Теперь абоненты не должны заботиться о возможности компрометации секретного ключа. Пользователи системы связи могут совершенно свободно обмениваться открытыми ключами и зашифрованными ими сообщениями. Если *пользователь* надежно хранит свой *закрытый ключ*, никто не сможет прочитать передаваемые сообщения.

Для упрощения процедуры обмена в сети передачи сообщений обычно используется *база данных* (подробнее об этом см. в "Криптографические алгоритмы с открытым ключом и их использование" ), в которой хранятся открытые ключи всех пользователей. При необходимости любой *пользователь* системы может запросить из базы *открытый ключ* другого человека и использовать полученный *ключ* для шифрования сообщений.

**Цифровая подпись на основе алгоритмов с открытым ключом**

Как и все люди, абоненты *сети передачи данных* могут не доверять друг другу или вести себя нечестно. Они могут подделывать чужие сообщения, отрицать свое авторство или выдавать себя за другое лицо. Особенно актуальными становятся эти проблемы в связи с развитием электронной коммерции и возможностью оплаты услуг через *Интернет*. Поэтому во многих системах связи получатель корреспонденции должен иметь возможность удостовериться в подлинности документа, а создатель электронного послания должен быть в состоянии доказать свое авторство получателю или третьей стороне. Следовательно, электронные документы должны иметь аналог обычной физической подписи. При этом подпись должна обладать следующими свойствами:

1. подпись воспроизводится только одним лицом, а подлинность ее может быть удостоверена многими;
2. подпись неразрывно связывается с данным сообщением и не может быть перенесена на другой документ;
3. после того, как документ подписан, его невозможно изменить;

4. от поставленной подписи невозможно отказаться, то есть лицо, подписавшее документ, не сможет потом утверждать, что не ставило подпись.

*Асимметричные алгоритмы* шифрования могут быть использованы для формирования **цифровой (электронной) подписи** (*digital signature*) – уникального числового дополнения к передаваемой информации, позволяющего проверить ее авторство. *Электронная (цифровая) подпись* (*ЭЦП*) представляет собой последовательность *бит* фиксированной длины, которая вычисляется определенным образом с помощью содержимого подписываемой информации и секретного ключа.

При формировании цифровой подписи специальным образом шифруется или все сообщение целиком, или результат вычисления хеш-функции от сообщения. Последний способ обычно оказывается предпочтительнее, так как подписываемое сообщение может иметь разный размер, иногда довольно большой, а хеш-код всегда имеет постоянную не очень большую длину. Рассмотрим подробнее оба варианта формирования *ЭЦП*.

Самый простой способ основывается, так же как и при открытом шифровании, на использовании пары связанных между собой ключей (открытого и закрытого). Однако роли закрытого и открытого ключей меняются – *ключ* подписывания становится секретным, а *ключ* проверки – открытым. Если при этом сохраняется свойство, что по открытому ключу нельзя практически найти *закрытый ключ*, то в качестве подписи может выступать само сообщение, зашифрованное секретным ключом. Таким образом подписать сообщение может только владелец закрытого ключа, но каждый, кто имеет его *открытый ключ*, может проверить подпись.

Пусть, например, *пользователь* А хочет отправить пользователю Б подписанное сообщение. Процедура создания и проверки подписи состоит из следующих шагов:

1. Пользователь А посылает пользователю Б свой открытый ключ U по любому каналу связи, например, по электронной почте.
2. Пользователь А шифрует сообщение M своим закрытым ключом R и получает зашифрованное сообщение C.
3. Зашифрованное сообщение пересылается пользователю Б.
4. Пользователь Б расшифровывает полученное сообщение C, используя открытый ключ пользователя А. Если сообщение расшифровалось, значит, оно подписано пользователем А.

Этот протокол можно изобразить в виде схемы, как на рис. 9.2.

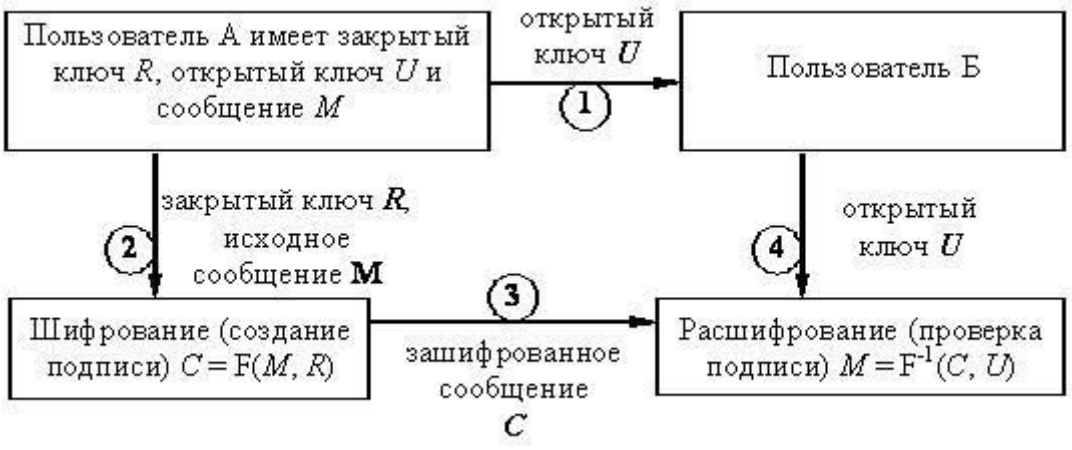


**Рис. 9.2.** Первый вариант схемы создания и проверки цифровой подписи

До тех пор, пока *пользователь* А надежно хранит свой *закрытый ключ*, его подписи достоверны. Кроме того, невозможно изменить сообщение, не имея доступа к закрытому ключу абонента А; тем самым обеспечивается *аутентичность* и *целостность* данных.

Физическое *представление* пары ключей зависит от конкретной системы, поддерживающей использование *ЭЦП*. Чаще всего *ключ* записывается в *файл*, который, в *дополнение* к самому ключу, может содержать, например, информацию о пользователе - владельце ключа, о сроке действия ключа, а также некий набор данных, необходимых для работы конкретной системы (подробнее об этом см. "Электронная цифровая подпись" ). Данные о владельце ключа позволяют реализовать другую важную функцию *ЭЦП* - установление авторства, поскольку при проверке подписи сразу же становится ясно, кто подписал то или иное сообщение. Обычно программные продукты, осуществляющие *проверку ЭЦП*, настраиваются так, чтобы результат исполнения появлялся на экране в удобном для восприятия виде с указанием поставившего подпись пользователя, например, так:

"Подпись файла приказ.doc верна (Автор: Соколов А.И.)"

На рис. 9.2представлена схема формирования так называемой **цифровой подписи с восстановлением документа**. Цифровые подписи с восстановлением документа как бы содержат в себе подписываемый документ: в процессе проверки подписи автоматически вычисляется и *тело документа*. Если при расшифровывании сообщение восстановилось правильно, значит, подпись была верной. *Цифровая подпись* с восстановлением документа может быть реализована, например, с помощью одного из самых популярных алгоритмов формирования *ЭЦП – RSA*.

В случае использования цифровой подписи с восстановлением документа все сообщение целиком подписывается, то есть шифруется. В настоящее время на практике так обычно не делается. *Алгоритмы шифрования* с открытым ключом достаточно медленные, кроме того, для подтверждения целостности сообщения требуется много памяти. К тому же практически все применяемые алгоритмы вычисления *ЭЦП* используют для расчета сообщения заранее заданной стандартной длины. Например, в российском алгоритме формирования цифровой подписи ГОСТ Р34.10-94 этот размер определен равным 32 байтам. Поэтому для экономии времени и вычислительных ресурсов, а также для удобства работы асимметричный *алгоритм* обычно используется вместе с какой-нибудь однонаправленной хеш-функцией. В этом случае вначале с помощью хеш-функции из сообщения произвольной длины вычисляется хеш-код нужного размера, а затем для вычисления *ЭЦП* производится *шифрование* полученного на предыдущем этапе хеш-кода от сообщения.

*ЭЦП*, вычисленные по хеш-коду документа, называют **присоединяемыми цифровыми подписями**. Такие цифровые подписи представляют собой некоторый числовой код, который необходимо пристыковывать к подписываемому документу. Само сообщение при этом не шифруется и передается в открытом виде вместе с цифровой подписью отправителя.

Если *пользователь* А хочет отправить пользователю Б сообщение M, дополненное присоединенной цифровой подписью, то процедура создания и проверки подписи должна состоять из следующих шагов:

1. Пользователь А посылает пользователю Б свой открытый ключ U по любому каналу связи, например, по электронной почте.
2. Пользователь А с помощью некоторой надежной хеш-функции H вычисляет хеш-код своего сообщения h = H(M).

3. Затем пользователь А шифрует хеш-код сообщения h своим закрытым ключом R и получает цифровую подпись С.
4. Исходное сообщение М вместе с цифровой подписью С пересылаются пользователю Б.
5. Пользователь Б вычисляет хеш-код h полученного сообщения М, а затем проверяет цифровую подпись С, используя открытый ключ пользователя А.

Этот протокол можно изобразить в виде схемы, как на рис. 9.3.

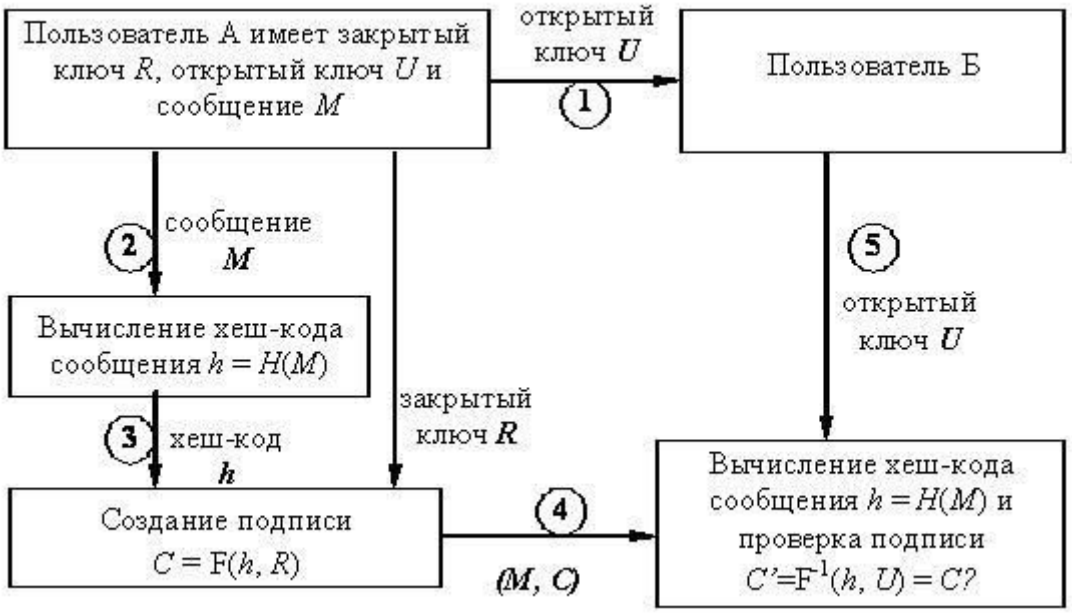


**Рис. 9.3.** Второй вариант схемы создания и проверки цифровой подписи

Хеш-*функция* не являются частью алгоритма *ЭЦП*, поэтому в схеме может быть использована любая надёжная хеш-*функция*.

Описанный процесс создания подписи не обеспечивает *конфиденциальность*. То есть сообщение, посланное таким способом, невозможно изменить, но можно прочитать. Даже если не использовать хеш-функцию, а шифровать все сообщение целиком, *конфиденциальность* не обеспечивается, так как любой может расшифровать сообщение, используя *открытый ключ* отправителя.

Во многих ситуациях приведенной схемы создания и использования цифровой подписи оказывается вполне достаточно. Однако бывают случаи, когда *пользователь* Б может смошенничать. Предположим, что пересылаемым документом был чек от пользователя А, например, за оказанные услуги. *Пользователь* Б удостоверился, что *цифровая подпись* на нем верная и использовал его для получения денег. Никто не может помешать пользователю Б снять одну или несколько копий с подписанного документа (тем более, если документ электронный) и периодически с небольшим интервалом предъявлять их в банк для получения денег.

Для устранения возможности такого жульничества в цифровые подписи часто включают *метки времени*. Дата и время подписания документа добавляются к сообщению и подписываются вместе со всем документом. При оплате чека *метка* времени может быть зафиксирована банком и занесена в базу данных. При попытке повторного предъявления чека банк это обнаружит и примет соответствующие меры.

Разновидностью цифровой подписи является *неотрицаемая цифровая подпись*. Как и обычная *цифровая подпись*, неотрицаемая *цифровая подпись* зависит от подписываемого документа и закрытого ключа автора. В отличие от обычной *ЭЦП* неотрицаемая подпись не может быть проверена без разрешения подписавшего. Таким образом, получатель корреспонденции не сможет показать подпись (или не сможет доказать правильность подписи) без согласия лица, подписавшего сообщение

**Формирование секретных ключей с использованием асимметричных алгоритмов**

На практике алгоритмы с открытым ключом редко используются для непосредственного шифрования сообщений. Этому препятствует относительная невысокая скорость *асимметричных алгоритмов* при шифровании и расшифровании больших объемов данных. Данный фактор связан с тем, что основной операцией в системах с открытым ключом является возведение в степень по большому модулю 500-1000 битовых чисел, что при программной реализации производится намного медленнее, чем *шифрование* того же объема данных классическими симметричными способами. Однако при обработке коротких блоков данных, например, ключей определенной длины, *алгоритмы шифрования* с открытым ключом могут использоваться достаточно эффективно. Поэтому часто используют следующую комбинированную схему: асимметричный *алгоритм* применяется для согласования *ключа сессии*, а затем этот *ключ* выступает в роли секретного ключа для шифрования сообщений *симметричным алгоритмом*.

Простейший протокол формирования секретного *ключа сессии* может выглядеть следующим образом (если пользователи некоторой системы связи имеют *доступ* к базе данных открытых ключей абонентов системы, предоставляемой центром распределения ключей, то они могут получать из нее открытые ключи друг друга):

1. Пользователь А получает открытый ключ пользователя Б из *центра распределения ключей* или непосредственно от пользователя Б.
2. Пользователь А генерирует случайный сеансовый ключ и зашифровывает его полученным открытым ключом.
3. Зашифрованный сеансовый ключ пересылается пользователю Б.
4. Пользователь Б расшифровывает полученный пакет своим закрытым ключом.
5. Пользователи А и Б используют согласованный сеансовый ключ для обмена шифрованными сообщениями.

Схему формирования парой пользователей А и В общего секретного ключа К для шифрования - расшифрования можно изобразить следующим образом ( рис. 9.4).

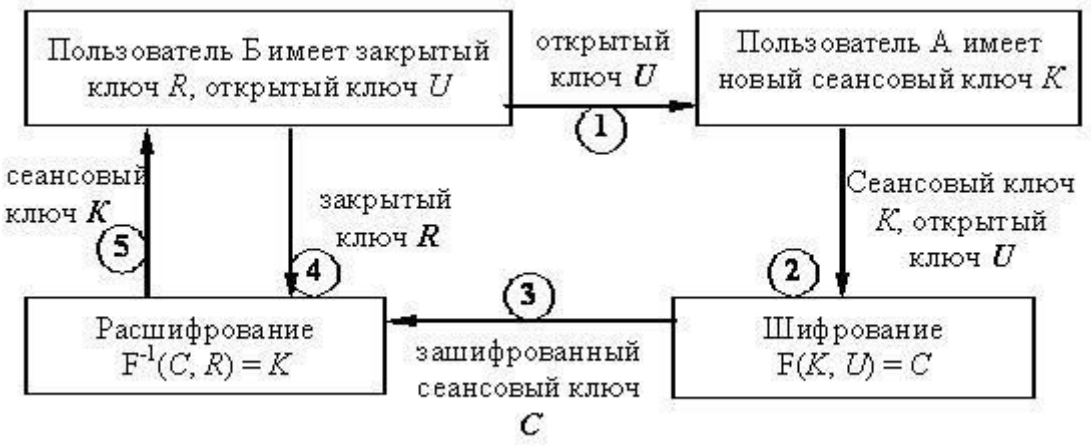


**Рис. 9.4.** Схема формирования общего секретного ключа

Эта схема напоминает рис. 9.1, и это неудивительно, ведь в ней используется тот же самый режим шифрования открытым ключом. Отличие заключается в том, *что* шифруется. В схеме на рис. 9.4 производится *шифрование* небольшого по размеру сеансового ключа, который в дальнейшем будет использован в качестве секретного ключа при шифровании *симметричным алгоритмом*. *Шифрование* небольшого по размеру блока данных выполняется достаточно быстро и не замедляет телекоммуникационные процессы даже в системе с многими тысячами пользователей.

Существуют и более сложные протоколы распределения ключей, обеспечивающие взаимное подтверждение подлинности участников сеанса связи, подтверждение достоверности сеанса *механизмом запроса*-ответа или другие требования.

**Требования к алгоритмам шифрования с открытым ключом**

Рассмотрев основные способы применения алгоритмов шифрования с открытым ключом, изучим требования, которым должен, по мнению основоположников теории шифрования с открытым ключом Диффи и Хеллмана, удовлетворять *алгоритм* шифрования с открытым ключом. Эти требования следующие:

1. Вычислительно легко создавать пару (открытый ключ, закрытый ключ).
2. Вычислительно легко зашифровать сообщение открытым ключом.
3. Вычислительно легко расшифровать сообщение, используя закрытый ключ.
4. Вычислительно невозможно, зная открытый ключ, определить соответствующий закрытый ключ.
5. Вычислительно невозможно, зная только открытый ключ и зашифрованное сообщение, восстановить исходное сообщение.

Из этих общих требований видно, что реализация конкретного алгоритма с открытым ключом зависит от соответствующей *односторонней функции*.

Математиками и криптографами предложено большое количество алгоритмов шифрования с открытым ключом, основанных на различных односторонних функциях. Некоторые алгоритмы можно задействовать тремя, рассмотренными ранее в данной лекции способами, в то время как другие могут использоваться только одним или двумя способами. Мы рассмотрим четыре алгоритма с открытым ключом, три из которых достаточно давно применяются на практике, а четвертый вид алгоритмов совсем недавно начал применяться в системах защиты информации. Эти алгоритмы используются обычно для различных целей, что отражено в следующей таблице.

| Название алгоритма | Возможность использования | | |
|---|---|---|---|
| | Шифрование / расшифрование данных | Цифровая подпись | Согласование или формирование ключа |
| RSA | Да | Да | Да |
| *Алгоритм Диффи-Хеллмана* | Нет | Нет | Да |
| Алгоритм Эль-Гамаля | Да | Да | Да |
| Алгоритмы с использованием эллиптических кривых | Да | Да | Да |

Перед изучением наиболее известных алгоритмов шифрования с открытым ключом, необходимо напомнить, что все *асимметричные алгоритмы* основаны на свойствах тех или иных математических функций. Доказательства правильности работы рассматриваемых алгоритмов могут быть достаточно сложными, поэтому мы ограничимся изучением основных принципов их работы. Многие криптографические алгоритмы базируются на результатах классической теории чисел. Основные факты и положения этой теории, необходимые для понимания алгоритмов и выполнения упражнений, сформулированы в "Основные положения теории чисел, используемые в криптографии с открытым ключом" .

**Ключевые термины**

**Алгоритм шифрования с открытым ключом** (или **асимметричные криптоалгоритмы** ) – криптографический *алгоритм*, в котором для шифрования и расшифрования используются разные ключи.

**Закрытый ключ** – *ключ*, используемый в асимметричных криптографических алгоритмах, который должен храниться в секрете.

**Односторонняя функция** – *математическая функция*, которую относительно легко вычислить, но трудно найти по значению функции соответствующее *значение* аргумента. То есть, зная x легко вычислить f(x), но по известному f(x) трудно найти подходящее *значение* x.

**Односторонняя функция с люком** (или **с секретом** ) – это особый вид *односторонних функций*, имеющих некоторый секрет (*люк*), позволяющий относительно быстро вычислить обратное *значение* функции.

**Открытый ключ** – *ключ*, используемый в асимметричных криптографических алгоритмах, который может не храниться в секрете.

**Присоединяемые цифровые подписи** – подписи, вычисленные по хеш-коду документа. Такие цифровые подписи представляют собой некоторый числовой код, который необходимо пристыковывать к подписываемому документу. Само сообщение при этом не шифруется и передается в открытом виде вместе с цифровой подписью отправителя.

**Цифровая (электронная) подпись** (*digital signature*) – уникальное числовое *дополнение* к передаваемой информации, позволяющее проверить ее авторство. *Электронная (цифровая) подпись* (*ЭЦП*) представляет собой последовательность *бит* фиксированной длины, которая вычисляется определенным образом с помощью содержимого подписываемой информации и секретного ключа.

**Цифровые подписи с восстановлением документа** – подписи, которые как бы содержат в себе подписываемый документ: в процессе проверки подписи автоматически

вычисляется и *тело документа*. Если при расшифровывании сообщение восстановилось правильно, значит, подпись была верной.

**Краткие итоги**

*Асимметричные алгоритмы* шифрования (или алгоритмы с открытым ключом) – криптографические алгоритмы, в которых один *ключ* используется для шифрования, а другой, отличный от первого, – для расшифрования. Алгоритмы называются асимметричными, так как ключи шифрования и расшифрования разные, следовательно, отсутствует *симметрия* основных криптографических процессов. Один из двух ключей является открытым (*public key*) и может быть объявлен всем, а второй – закрытым (*private key*) и должен держаться в секрете. Какой из ключей, открытый или закрытый, используется для шифрования, а какой для расшифрования, определяется назначением криптографической системы.

Все *алгоритмы шифрования* с открытым ключом основаны на использовании *односторонних функций*. Односторонней функцией называется *математическая функция*, которую относительно легко вычислить, но трудно найти по значению функции соответствующее *значение* аргумента. Использовать *односторонние функции* для шифрования сообщений с целью их защиты не имеет смысла, так как обратно расшифровать зашифрованное сообщение уже не получится. Для целей шифрования используются *односторонние функции с люком* (или с секретом) – особый вид *односторонних функций*, имеющих некоторый секрет (*люк*), позволяющий относительно быстро вычислить обратное *значение* функции.

*Алгоритмы шифрования* с открытым ключом можно использовать для решения следующих задач:

1. Для шифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа.
2. Для формирования цифровой подписи под электронными документами.
3. Для распределения секретных ключей, используемых потом при шифровании документов симметричными методами.

Цифровая (электронная) подпись – уникальное числовое *дополнение* к передаваемой информации, позволяющее проверить ее авторство. *Электронная (цифровая) подпись* (*ЭЦП*) представляет собой последовательность *бит* фиксированной длины, которая вычисляется определенным образом с помощью содержимого подписываемой информации и секретного ключа. Различают присоединяемые цифровые подписи и цифровые подписи с восстановлением документа. Присоединяемые цифровые подписи – подписи, вычисленные по хеш-коду документа. Такие цифровые подписи представляют собой некоторый числовой код, который необходимо пристыковывать к подписываемому документу. Само сообщение при этом не шифруется и передается в открытом виде вместе с цифровой подписью отправителя. Цифровые подписи с восстановлением документа – подписи, которые как бы содержат в себе подписываемый документ: в процессе проверки подписи автоматически вычисляется и *тело документа*. Если при расшифровывании сообщение восстановилось правильно, значит, подпись была верной.

**Симметричная или асимметричная криптография?**

Однозначного ответа на вопрос о том, какие алгоритмы - симметричные или асимметричные - предпочтительнее, конечно же, нет. Основным достоинством

симметричной криптографии является высокая скорость обработки данных. Проблемы криптосистем с закрытым ключом обсуждались подробно в лекции 8. Попробуем теперь оценить особенности алгоритмов шифрования с открытым ключом.

Главным достоинством асимметричной криптографии является отсутствие необходимости в предварительном доверенном обмене ключевыми элементами при организации секретного обмена сообщениями. К основным недостаткам асимметричных криптосистем, мешающим им вытеснить симметричные методы шифрования, относят следующие.

1. Алгоритмы с открытым ключом работают намного (в сотни раз) медленнее классических алгоритмов с закрытым ключом. Это их самый главный недостаток. Связан он с тем, что основной операцией в системах с открытым ключом является возведение в степень по большому модулю 500-1000 битовых чисел, что при программной реализации производится намного медленнее, чем шифрование того же объема данных классическими способами.
2. Алгоритмы с открытым ключом требуют обеспечения достоверности открытых ключей, что порой превращается в довольно сложную задачу. То же самое относится и к протоколам цифровой подписи. Для управления открытыми ключами используют специальную инфраструктуру открытых ключей, обеспечивающую функции управления открытыми ключами.
3. Алгоритмы с открытым ключом чувствительны к атакам по выбранному открытому тексту.

Таким образом, с практической точки зрения системы с открытым ключом и *асимметричным шифрованием* целесообразно использовать лишь для распределения секретных ключей и организации цифровых подписей, так как для решения этих задач не требуется шифрования больших блоков данных.

Использование *асимметричных алгоритмов* позволяет создавать сеансовые ключи шифрования, которые удаляются после окончания сеанса связи. Это значительно снижает риск вскрытия зашифрованных сообщений, так как, если каждое передаваемое сообщение шифруется уникальным сеансовым ключом, задача взломщика существенно усложняется. Причем пользователям совсем необязательно выполнять *протокол обмена ключом* перед симметричным шифрованием. Вот возможный вариант протокола передачи зашифрованных данных одновременно с передачей ключа.

1. Пользователь А генерирует случайный *сеансовый ключ* К и зашифровывает им с помощью *симметричного алгоритма* $F_{сим}$ свое сообщение М:
   $Ст = F_{сим}(M, K)$
2. Пользователь А получает из базы данных открытый ключ U пользователя Б и зашифровывает им *сеансовый ключ* К:
   $Ck = F_{асим}(K, U)$
3. Пользователь А посылает своему абоненту зашифрованное сообщение Ст и зашифрованный *сеансовый ключ* Ck. Для защиты от вскрытия "человек-в-середине" передаваемые данные могут быть дополнены цифровой подписью.
4. Пользователь Б расшифровывает полученный *сеансовый ключ* Ck с помощью своего закрытого ключа R:
   $K = F_{асим}^{-1}(Ck, R)$
5. Пользователь Б расшифровывает сообщение с помощью сеансового ключа К:
   $M = F_{сим}^{-1}(Ст, K)$

Такая *криптографическая система* называется *смешанной*, так как в ней используется и асимметричное, и *симметричное шифрование*. Смешанные криптосистемы широко применяются на практике: в банковских и платежных сетях передачи данных, в мобильной связи, в системах электронной почты и др. Для лучшего обеспечения безопасности они могут быть дополнены цифровыми подписями пользователей и удостоверяющего центра, метками времени. *Цифровая подпись* в сочетании с открытым распределением ключей позволяют организовывать защищенный обмен электронными документами.