**Тема 28-29 Средства и протоколы шифрования сообщений**

1 Протокол обмена сообщениями с использованием симметричного шифрования

2 Протокол обмена сообщениями с использованием шифрования с открытым ключом

3 Гибридные криптосистемы

4 Цифровая подпись

5 Подписание документов при помощи симметричных криптосистем и арбитра

**1 Протокол обмена сообщениями с использованием симметричного шифрования**

**Взаимная аутентификация**

Данные протоколы применяются для взаимной *аутентификации* участников и для обмена *ключом сессии*.

Основной задачей таких протоколов является обеспечение конфиденциального распределения *ключа сессии* и гарантирование его своевременности, то есть протокол не должен допускать повторного использования старого *ключа сессии*. Для обеспечения конфиденциальности *ключи сессии* должны передаваться в зашифрованном виде. Вторая задача, обеспечение своевременности, важна, потому что существует угроза перехвата передаваемого сообщения и повторной его пересылки. Такие повторения в худшем случае могут позволять взломщику использовать скомпрометированный *ключ сессии*, при этом успешно подделываясь под другого участника. Успешное повторение может, как минимум, разорвать операцию *аутентификации* участников.

Такие повторы называются *replay-атаками*. Рассмотрим возможные примеры подобных *replay-атак*:

1. Простое повторение: противник просто копирует сообщение и повторяет его позднее.
2. Повторение, которое не может быть определено: противник уничтожает исходное сообщение и посылает скопированное ранее сообщение.

Один из возможных подходов для предотвращения *replay-атак* мог бы состоять в присоединении последовательного номера (*sequence number*) к каждому сообщению, используемому в *аутентификационном* обмене. Новое сообщение принимается только тогда, когда его последовательный номер правильный. Трудность данного подхода состоит в том, что каждому участнику требуется поддерживать значения *sequence number* для каждого участника, с которым он взаимодействует в данный момент. Поэтому обычно *sequence number* не используются для *аутентификации* и обмена ключами. Вместо этого применяется один из следующих способов:

1. *Отметки времени*: участник А принимает сообщение как не устаревшее только в том случае, если оно содержит *отметку времени*, которая, по мнению А, соответствует текущему времени. Этот подход требует, чтобы часы всех участников были синхронизированы.
2. Запрос/ответ: участник А посылает в запросе к В случайное число ( ***nonce*** - number only once ) и проверяет, чтобы ответ от В содержал корректное значение этого *nonce*.

Считается, что подход с *отметкой времени* не следует использовать в приложениях, ориентированных на соединение, потому что это технически трудно, так как таким протоколам, кроме поддержки соединения, необходимо будет поддерживать синхронизацию часов различных процессоров. При этом возможный способ осуществления успешной атаки может возникнуть, если временно будет отсутствовать синхронизация часов одного из участников. В результате различной и непредсказуемой природы сетевых задержек распределенные часы не могут поддерживать точную синхронизацию. Следовательно, процедуры, основанные на любых *отметках времени*, должны допускать окно времени, достаточно большое для приспособления к сетевым задержкам, и достаточно маленькое для минимизации возможности атак.

С другой стороны, подход запрос/ответ не годится для приложений, не устанавливающих соединения, так как он требует предварительного рукопожатия перед началом передач, тем самым отвергая основное *свойство транзакции* без установления соединения. Для таких приложений доверие к некоторому безопасному серверу часов и постоянные попытки каждой из частей синхронизировать свои часы с этим сервером может быть оптимальным подходом.

**Использование симметричного шифрования**

Для обеспечения *аутентификации* и распределения *ключа сессии* часто используется двухуровневая иерархия ключей *симметричного шифрования*. В общих чертах эта стратегия включает использование доверенного *центра распределения ключей* ( *KDC* ). Каждый участник разделяет секретный ключ, называемый также *мастер-ключом*, с *KDC*. *KDC* отвечает за создание ключей, называемых *ключами сессии*, и за распределение этих ключей с использованием *мастер-ключей*. *Ключи сессии* применяются в течение короткого времени для шифрования только данной сессии между двумя участниками.

Большинство алгоритмов распределения *секретного ключа* с использованием *KDC*, включает также возможность *аутентификации* участников.

Протокол Нидхэма и Шредера

Предполагается, что секретные *мастер-ключи* KA и KB разделяют соответственно А и *KDC* и В и *KDC* . Целью протокола является безопасное распределение *ключа сессии* KS между А и В. Протокол представляет собой следующую последовательность шагов:

1. A -> KDC: IDA || IDB || N1

2. KDC -> A: EKa [KS || IDB || N1 ||

EKb [KS || IDA] ]

3. A -> B: EKb [KS || IDA]

4. B -> A: EKS [N2]

5. A -> B: EKS [f (N2)]

1. А запрашивает у *KDC* *ключ сессии* для установления защищенного соединения с В. Сообщение включает идентификацию А и В и уникальный идентификатор данной транзакции, который обозначен как N1 и называется *nonce*. *Nonce* может быть временной меткой, счетчиком или случайным числом; минимальное требование состоит в том, чтобы он отличался для каждого запроса. Кроме того, для предотвращения подделки желательно, чтобы противнику было трудно предугадать *nonce*. Таким образом, случайное число является лучшим вариантом для *nonce*.
2. *KDC* отвечает сообщением, зашифрованным ключом KА. Таким образом, только А может расшифровать сообщение, и А уверен, что оно получено от *KDC* , так как предполагается, что кроме А и *KDC* этот ключ не знает никто. Это сообщение включает следующие элементы, предназначенные для А:
   * Одноразовый *ключ сессии*.
   * Идентификатор В.
   * *nonce*, который идентифицирует данную сессию .

А должен убедиться, что полученный *nonce* равен значению *nonce* из первого запроса. Это доказывает, что ответ от *KDC* не был модифицирован при пересылке и не является повтором некоторого предыдущего запроса. Кроме того, сообщение включает два элемента, предназначенные для В:

* + Одноразовый *ключ сессии* KS.
  + Идентификатор А IDA.

Эти два последних элемента шифруются *мастер-ключом*, который *KDC* разделяет с В. Они посылаются В при установлении соединения и доказывают идентификацию А.

1. А сохраняет у себя *ключ сессии* и передает В информацию от *KDC* , предназначенную В: ЕKb [KS || IDA]. Так как эта информация зашифрована KВ, она защищена от просмотра. Теперь В знает *ключ сессии* ( KS ), знает, что другим участником является А, ( IDA ) и что начальная информация передана от *KDC* , т.к. она зашифрована с использованием KB.

В этой точке *ключ сессии* безопасно передан от А к В, и они могут начать безопасный обмен. Тем не менее, существует еще два дополнительных шага:

1. Используя созданный *ключ сессии*, В пересылает A *nonce* N2.
2. Также используя KS, А отвечает f (N2), где f - функция, выполняющая некоторую модификацию N2.

Эти шаги гарантируют B, что сообщение, которое он получил, не изменено и не является повтором предыдущего сообщения.

Заметим, что реальное распределение ключа включает только шаги 1 - 3, а шаги 4 и 5, как и 3, выполняют функцию *аутентификации*.

А безопасно получает *ключ сессии* на шаге 2. Сообщение на шаге 3 может быть дешифровано только B. Шаг 4 отражает знание В ключа KS, и шаг 5 гарантирует В знание участником А ключа KS и подтверждает, что это не устаревшее сообщение, так как используется *nonce* N2. Шаги 4 и 5 призваны предотвратить общий тип *replay-атак*. В частности, если противник имеет возможность захватить сообщение на шаге 3 и повторить его, то это должно привести к разрыву соединения.

Разрывая рукопожатие на шагах 4 и 5, протокол все еще уязвим для некоторых форм атак повторения. Предположим, что противник Х имеет возможность скомпрометировать старый *ключ сессии*. Маловероятно, чтобы противник мог сделать больше, чем просто копировать сообщение шага 3. Потенциальный риск состоит в том, что Х может заставить взаимодействовать А и B, используя старый *ключ сессии*. Для этого Х просто повторяет сообщение шага 3, которое было перехвачено ранее и содержит скомпрометированный *ключ сессии*. Если В не запоминает идентификацию всех предыдущих *ключей сессий* с А, он не сможет определить, что это повтор. Далее Х должен перехватить сообщение рукопожатия на шаге 4 и представиться А в ответе на шаге 5.

Протокол Деннинга

Деннинг предложил преодолеть эту слабость модификацией протокола Нидхэма и Шредера, которая включает дополнительную *отметку времени* на шагах 2 и 3:

1. A -> KDC: IDA || IDB

2. KDC -> A: EKa [KS || IDB || T ||

EKb [KS || IDA || T] ]

3. A -> B: EKb [KS || IDA || T]

4. B -> A: EKS [N1]

5. A -> B: EKS [f (N1)]

Т - это *отметка времени*, которая гарантирует А и B, что *ключ сессии* является только что созданным. Таким образом, и А, и В знают, что *распределенный ключ* не является старым. А и В могут верифицировать *временную отметку* проверкой, что

|Clock - T| < \Delta t_{1} + \Delta t_{2}

где \Delta  t_{1} - оцениваемое нормальное расхождение между часами *KDC* и локальными часами (у А или B ) и t2 - ожидаемая сетевая задержка времени. Каждый участник может установить свои часы, ориентируясь на определенный доверенный источник. Поскольку *временная отметка* Т шифруется с использованием секретных *мастер-ключей*, взломщик, даже зная старый *ключ сессии*, не сможет достигнуть цели повторением шага 3 так, чтобы В не заметил искажения времени.

Шаги 4 и 5 не были включены в первоначальное представление, но были добавлены позднее. Эти шаги подтверждают А, что В получил *ключ сессии*.

Протокол Деннинга обеспечивает большую степень безопасности по сравнению с протоколом Нидхэма и Шредера. Однако данная схема требует доверия к часам, которые должны быть синхронизированы в сети. В этом есть определенный риск, который состоит в том, что распределенные часы могут рассинхронизироваться в результате диверсии или повреждений. Проблема возникает, когда часы отправителя спешат по отношению к часам получателя. В этом случае противник может перехватить сообщение от отправителя и повторить его позднее, когда *отметка времени* в сообщении станет равной времени на узле получателя. Это повторение может иметь непредсказуемые последствия.

Один способ вычисления атак повторения состоит в требовании, чтобы участники регулярно сверяли свои часы с часами *KDC* . Другая альтернатива, при которой нет необходимости всем синхронизировать часы, состоит в доверии *протоколам рукопожатия*, использующим *nonce*.

Протокол аутентификации с использованием билета

Данный протокол пытается преодолеть проблемы, возникшие в предыдущих двух протоколах. Он выглядит следующим образом:

1. A -> B: IDA || Na

2. B -> KDC: IDB || Nb || EKb [IDA || Na || Tb]

3. KDC -> A: EKa [IDB || Na || KS || Tb] ||

EKb [IDA || KS || Tb] || Nb

4. A -> B: EKb [IDA || KS || Tb] || EKS [Nb]

1. А инициализирует *аутентификационный* обмен созданием *nonce* Na и посылкой его и своего идентификатора к В в незашифрованном виде. Этот *nonce* вернется к А в зашифрованном сообщении, включающем *ключ сессии*, гарантируя А, что *ключ сессии* не старый.
2. B сообщает *KDC* , что необходим *ключ сессии*. Это сообщение к KDС включает идентификатор В и *nonce* Nb. Данный *nonce* вернется к В в зашифрованном сообщении, которое включает *ключ сессии*, гарантируя B, что *ключ сессии* не устарел. Сообщение В к *KDC* также включает блок, зашифрованный *секретным ключом*, разделяемым В и *KDC* . Этот блок используется для указания *KDC* , когда заканчивается время жизни данного *ключа сессии*. Блок также специфицирует намеченного получателя и содержит *nonce*, полученный от А. Этот блок является своего рода "верительной грамотой" или "билетом" для А.
3. *KDC* получил *nonces* от А и В и блок, зашифрованный *секретным ключом*, который В разделяет с *KDC* . Блок служит билетом, который может быть использован А для последующих *аутентификаций*. *KDC* также посылает А блок, зашифрованный *секретным ключом*, разделяемым А и *KDC* . Этот блок доказывает, что В получил начальное сообщение А (IDB), что в нем содержится допустимая *отметка времени* и нет повтора ( Na ). Этот блок обеспечивает А *ключом сессии* ( KS ) и устанавливает ограничение времени на его использование ( Тb ).
4. А посылает полученный билет В вместе с *nonce* B, зашифрованным *ключом сессии*. Этот билет обеспечивает В *ключом сессии*, который тот использует для *дешифрования* и проверки *nonce*. Тот факт, что *nonce* B *расшифрован* *ключом сессии*, доказывает, что сообщение пришло от А и не является повтором.

Данный протокол аутентифицирует А и В и распределяет *ключ сессии*. Более того, протокол предоставляет в распоряжение А билет, который может использоваться для его последующей *аутентификации*, исключая необходимость повторных контактов с *аутентификационным* сервером. Предположим, что А и В установили сессию с использованием описанного выше протокола и затем завершили эту сессию. Впоследствии, но до истечения лимита времени, установленного протоколом, А может создать новую сессию с B. Используется следующий протокол:

1. A -> B: EKb [IDA || KS || Tb], Na'

2. B -> A: Nb', ES [Na']

3. A -> B: ES [Nb']

Когда В получает сообщение на шаге 1, он проверяет, что билет не просрочен. Заново созданные *nonces* Na' и Nb' гарантируют каждому участнику, что не было атак повтора. Время Tb является временем относительно часов B. Таким образом, эта временная метка не требует синхронизации, потому что В проверяет только им самим созданные *временные отметки*.

**2 Протокол обмена сообщениями с использованием шифрования с открытым ключом**

**Основные требования к алгоритмам асимметричного шифрования**

Создание *алгоритмов асимметричного шифрования* является величайшим и, возможно, единственным революционным достижением в истории криптографии.

*Алгоритмы шифрования* с *открытым ключом* разрабатывались для того, чтобы решить две наиболее трудные задачи, возникшие при использовании *симметричного шифрования*.

Первой задачей является *распределение ключа*. При симметричном шифровании требуется, чтобы обе стороны уже имели общий *ключ*, который каким-то образом должен быть им заранее передан. Диффи, один из основоположников шифрования с *открытым ключом*, заметил, что это требование отрицает всю суть криптографии, а именно возможность поддерживать всеобщую секретность при коммуникациях.

Второй задачей является необходимость создания таких механизмов, при использовании которых невозможно было бы подменить кого-либо из участников, т.е. нужна *цифровая подпись*. При использовании коммуникаций для решения широкого круга задач, например в коммерческих и частных целях, электронные сообщения и документы должны иметь эквивалент подписи, содержащейся в бумажных документах. Необходимо создать метод, при использовании которого все участники будут убеждены, что электронное сообщение было послано конкретным участником. Это более сильное требование, чем *аутентификация*.

Диффи и Хеллман достигли значительных результатов, предложив способ решения обеих задач, который радикально отличается от всех предыдущих подходов к шифрованию.

Сначала рассмотрим общие черты *алгоритмов шифрования* с *открытым ключом* и требования к этим алгоритмам. Определим требования, которым должен соответствовать *алгоритм*, использующий один *ключ* для шифрования, другой *ключ* - для *дешифрования*, и при этом вычислительно невозможно определить дешифрующий *ключ*, зная только *алгоритм* шифрования и шифрующий *ключ*.

Кроме того, некоторые алгоритмы, например *RSA*, имеют следующую характеристику: каждый из двух ключей может использоваться как для шифрования, так и для *дешифрования*.

Сначала рассмотрим алгоритмы, обладающие обеими характеристиками, а затем перейдем к алгоритмам *открытого ключа*, которые не обладают вторым свойством.

При описании *симметричного шифрования* и шифрования с *открытым ключом* будем использовать следующую терминологию. *Ключ*, используемый в *симметричном шифровании*, будем называть *секретным ключом*. Два ключа, используемые при шифровании с *открытым ключом*, будем называть ***открытым ключом*** и ***закрытым ключом*** . *Закрытый ключ* держится в секрете, но называть его будем *закрытым ключом*, а не секретным, чтобы избежать путаницы с ключом, используемым в *симметричном шифровании*. *Закрытый ключ* будем обозначать KR, *открытый ключ* - KU.

Будем предполагать, что все участники имеют *доступ* к *открытым ключам* друг друга, а *закрытые ключи* создаются локально каждым участником и, следовательно, распределяться не должны.

В любое время участник может изменить свой *закрытый ключ* и опубликовать составляющий пару *открытый ключ*, заменив им старый *открытый ключ*.

Диффи и Хеллман описывают требования, которым должен удовлетворять *алгоритм шифрования* с *открытым ключом*.

1. Вычислительно легко создавать пару ( *открытый ключ* KU, *закрытый ключ* KR ).
2. Вычислительно легко, имея *открытый ключ* и незашифрованное сообщение М, создать соответствующее зашифрованное сообщение:

С = ЕKU[М]

1. Вычислительно легко дешифровать сообщение, используя *закрытый ключ*:

М = DKR[C] = DKR[EKU[M]]

1. Вычислительно невозможно, зная *открытый ключ* KU, определить *закрытый ключ* KR.
2. Вычислительно невозможно, зная *открытый ключ* KU и зашифрованное сообщение С, восстановить исходное сообщение М.

Можно добавить шестое требование, хотя оно не выполняется для всех алгоритмов с *открытым ключом*:

1. Шифрующие и дешифрующие функции могут применяться в любом порядке:

М = ЕKU[DKR[M]]

Это достаточно сильные требования, которые вводят понятие *односторонней функции с люком*. ***Односторонней функцией*** называется такая функция, у которой каждый аргумент имеет единственное обратное значение, при этом вычислить саму функцию легко, а вычислить обратную функцию трудно.

|  |
| --- |
| Y = f(X) - легко |
| X = f-1(Y) - трудно |

Обычно "легко" означает, что проблема может быть решена за полиномиальное время от длины входа. Таким образом, если *длина* входа имеет n битов, то время вычисления функции пропорционально na, где а - фиксированная константа. Таким образом, говорят, что *алгоритм* принадлежит классу полиномиальных алгоритмов Р. Термин "трудно" означает более сложное понятие. В общем случае будем считать, что проблему решить невозможно, если усилия для ее решения больше полиномиального времени от величины входа. Например, если *длина* входа n битов, и время вычисления функции пропорционально 2n, то это считается вычислительно невозможной задачей. К сожалению, тяжело определить, проявляет ли конкретный *алгоритм* такую сложность. Более того, традиционные представления о вычислительной сложности фокусируются на худшем случае или на среднем случае сложности алгоритма. Это неприемлемо для криптографии, где требуется невозможность инвертировать функцию для всех или почти всех значений входов.

Вернемся к определению ***односторонней функции с люком***, которую, подобно *односторонней функции*, легко вычислить в одном направлении и трудно вычислить в обратном направлении до тех пор, пока недоступна некоторая дополнительная информация. При наличии этой дополнительной информации *инверсию* можно вычислить за полиномиальное время. Таким образом, *односторонняя функция* с люком принадлежит семейству *односторонних функций* fk таких, что

|  |
| --- |
| Y = fk(X) - легко, если k и Х известны |
| X = fk-1(Y) - легко, если k и Y известны |
| Х = fk-1(Y) - трудно, если Y известно, но k неизвестно |

Мы видим, что разработка конкретного алгоритма с *открытым ключом* зависит от открытия соответствующей *односторонней функции с люком*.

**Криптоанализ алгоритмов с открытым ключом**

Как и в случае *симметричного шифрования*, *алгоритм шифрования* с *открытым ключом* уязвим для лобовой атаки. Контрмера стандартная: использовать большие ключи.

*Криптосистема* с *открытым ключом* применяет определенные неинвертируемые *математические функции*. Сложность вычислений таких функций не является линейной от количества битов ключа, а возрастает быстрее, чем ключ. Таким образом, размер ключа должен быть достаточно большим, чтобы сделать лобовую атаку непрактичной, и достаточно маленьким для возможности практического шифрования. На практике размер ключа делают таким, чтобы лобовая атака была непрактичной, но в результате скорость шифрования оказывается достаточно медленной для использования алгоритма в общих целях. Поэтому шифрование с *открытым ключом* в настоящее время в основном ограничивается приложениями управления ключом и подписи, в которых требуется шифрование небольшого блока данных.

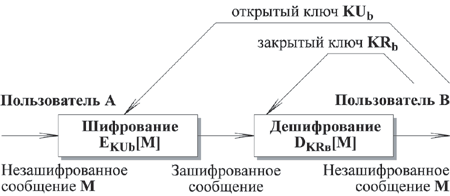
Другая форма атаки состоит в том, чтобы найти способ вычисления *закрытого ключа*, зная *открытый ключ*. Невозможно математически доказать, что данная форма атаки исключена для конкретного алгоритма *открытого ключа*. Таким образом, любой алгоритм, включая широко используемый *алгоритм RSA*, является подозрительным.

Наконец, существует форма атаки, специфичная для способов использования систем с *открытым ключом*. Это атака вероятного сообщения. Предположим, например, что посылаемое сообщение состоит исключительно из 56-битного *ключа сессии* для алгоритма симметричного шифрования. Противник может зашифровать все *возможные ключи*, используя *открытый ключ*, и может дешифровать любое сообщение, соответствующее передаваемому зашифрованному тексту. Таким образом, независимо от размера ключа схемы *открытого ключа*, атака сводится к лобовой атаке на 56-битный *симметричный ключ*. Защита от подобной атаки состоит в добавлении определенного количества случайных битов в простые сообщения.

**Основные способы использования алгоритмов с открытым ключом**

Основными способами использования алгоритмов с *открытым ключом* являются шифрование/*дешифрование*, создание и проверка подписи и обмен ключа.

**Шифрование** с *открытым ключом* состоит из следующих шагов:

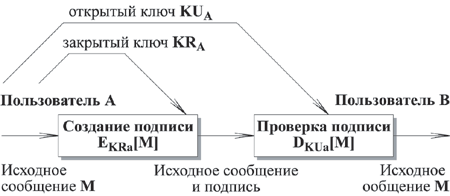


**Рис. 7.1.**Шифрование с открытым ключом

1. Пользователь В создает пару ключей KUb и KRb, используемых для шифрования и *дешифрования* передаваемых сообщений.
2. Пользователь В делает доступным некоторым надежным способом свой ключ шифрования, т.е. *открытый ключ* KUb. Составляющий пару *закрытый ключ* KRb держится в секрете.
3. Если А хочет послать сообщение В, он шифрует сообщение, используя *открытый ключ* В KUb.
4. Когда В получает сообщение, он дешифрует его, используя свой *закрытый ключ* KRb. Никто другой не сможет дешифровать сообщение, так как этот *закрытый ключ* знает только В.

Если пользователь (конечная система) надежно хранит свой *закрытый ключ*, никто не сможет подсмотреть передаваемые сообщения.

Создание и проверка подписи состоит из следующих шагов:



**Рис. 7.2.**Создание и проверка подписи

1. Пользователь А создает пару ключей KRA и KUA, используемых для создания и проверки подписи передаваемых сообщений.
2. Пользователь А делает доступным некоторым надежным способом свой ключ проверки, т.е. *открытый ключ* KUA. Составляющий пару *закрытый ключ* KRA держится в секрете.
3. Если А хочет послать подписанное сообщение В, он создает подпись EKRa[M] для этого сообщения, используя свой *закрытый ключ* KRA.
4. Когда В получает подписанное сообщение, он проверяет подпись DKUa[M], используя *открытый ключ* А KUA. Никто другой не может подписать сообщение, так как этот *закрытый ключ* знает только А.

До тех пор, пока пользователь или прикладная система надежно хранит свой *закрытый ключ*, их подписи достоверны.

Кроме того, невозможно изменить сообщение, не имея доступа к *закрытому ключу* А ; тем самым обеспечивается аутентификация и *целостность данных*.

В этой схеме все сообщение подписывается, причем для подтверждения целостности сообщения требуется много памяти. Каждое сообщение должно храниться в незашифрованном виде для использования в практических целях. Кроме того, копия сообщения также должна храниться в зашифрованном виде, чтобы можно было проверить в случае необходимости подпись. Более эффективным способом является шифрование небольшого блока битов, который является функцией от сообщения. Такой блок, называемый *аутентификатором*, должен обладать свойством невозможности изменения сообщения без изменения *аутентификатора*. Если *аутентификатор* зашифрован *закрытым ключом* отправителя, он является ***цифровой подписью***, с помощью которой можно проверить исходное сообщение. Далее эта технология будет рассматриваться в деталях.

Важно подчеркнуть, что описанный процесс создания подписи не обеспечивает конфиденциальность. Это означает, что сообщение, посланное таким способом, невозможно изменить, но можно подсмотреть. Это очевидно в том случае, если подпись основана на аутентификаторе, так как само сообщение передается в явном виде. Но даже если осуществляется шифрование всего сообщения, конфиденциальность не обеспечивается, так как любой может расшифровать сообщение, используя *открытый ключ* отправителя.

**Обмен ключей**: две стороны взаимодействуют для обмена *ключом сессии*, который в дальнейшем можно использовать в алгоритме симметричного шифрования.

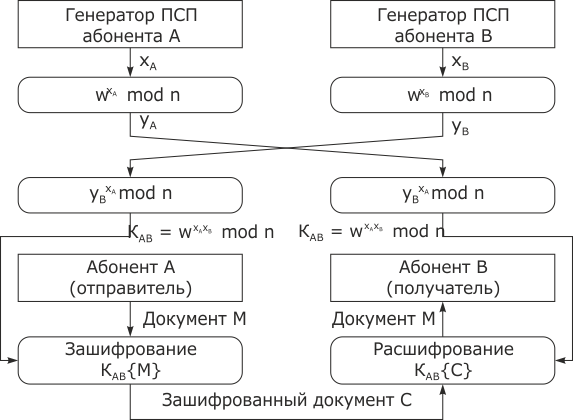
Некоторые алгоритмы можно задействовать тремя способами, в то время как другие могут использоваться одним или двумя способами.

Перечислим наиболее популярные алгоритмы с *открытым ключом* и возможные способы их применения.

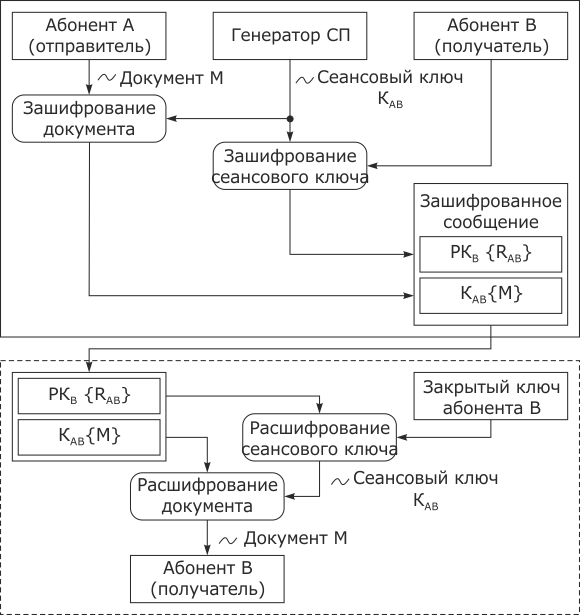
|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм** | **Шифрование / *дешифрование*** | ***Цифровая подпись*** | **Обмен ключей** |
| *RSA* | Да; непригоден для больших блоков | Да | Да |
| *DSS* | Нет | Да | Нет |
| *Диффи-Хеллман* | Нет | Нет | Да |

**3 Гибридные криптосистемы**

Несмотря на все преимущества **криптосистем с открытым ключом**, ни одна из известных на сегодняшний день их реализаций не может конкурировать по быстродействию с **криптосистемами с секретным ключом**. Так, например, быстродействие системы RSA в тысячи раз ниже быстродействия DES. В результате при шифровании длинных информационных последовательностей может случиться так, что применение асимметричного алгоритма недопустимо снижает скорость информационного обмена, а применение симметричного невозможно из-за отсутствия общего секретного ключа у участников этого обмена или по каким-то другим причинам. Выходом из этой ситуации является использование гибридной криптосистемы, оба варианта которой приведены на [рис. 3.23](https://www.intuit.ru/studies/courses/3580/822/lecture/30592?page=4#image.3.23) и [3.24](https://www.intuit.ru/studies/courses/3580/822/lecture/30592?page=4#image.3.24).



**Рис. 3.23.**Первый вариант схемы гибридного шифрования



**Рис. 3.24.**Второй вариант схемы гибридного шифрования

В схеме, показанной на [рис. 3.23](https://www.intuit.ru/studies/courses/3580/822/lecture/30592?page=4#image.3.23), на начальном этапе участники информационного обмена (абоненты А и В), используя протокол выработки общего секретного ключа, формируют общую секретную информацию (сеансовый ключ K_{АВ}). На следующем этапе для обмена зашифрованными сообщения используется криптосистема с секретным ключом.

Схема, показанная на [рис. 3.24](https://www.intuit.ru/studies/courses/3580/822/lecture/30592?page=4#image.3.24), предполагает наличие у каждого участника информационного обмена двух ключей: открытого PK и закрытого SK. Рассмотрим процесс пересылки некоего документа M. Отправитель (абонент А) вырабатывает секретный ключ - случайное число, используемое только один раз и поэтому называемое одноразовым или сеансовым ключом. Этот ключ используется для зашифрования документа M при помощи симметричного криптоалгоритма. Сеансовый ключ зашифровывается на открытом ключе получателя (абонент В) и присоединяется к ранее зашифрованному документу. Сформированное таким образом сообщение отсылается получателю. Последний, получив сообщение, повторяет те же процедуры, но в обратном порядке. С помощью своего секретного ключа получатель восстанавливает сеансовый ключ, а затем с его помощью расшифровывает и сам документ.

**4 Цифровая подпись**

Мы все знакомы с понятием подписи. Человек подписывает документ, чтобы показать, что нечто сделано им самим или было им одобрено. Подпись - *доказательство* получателю, что документ исходит от истинного объекта. Когда клиент подписывает чек, банк должен убедиться, что чек принадлежит клиенту и никому другому. Другими словами, подпись на документе является признаком установления подлинности автора - подлинности документа. Рассмотрим картину, подписанную художником. Подпись на предмете искусства, если она подлинная, означает, что этот предмет, вероятно, подлинный.

Когда Алиса передает сообщение Бобу, Боб должен проверить подлинность передатчика; он должен убедиться, что сообщение исходит от Алисы, а не от Евы. Боб может попросить, чтобы Алиса подписала сообщение с помощью электроники. Другими словами, *электронная подпись* может доказать подлинность Алисы как передатчика сообщения. Мы называем этот тип подписи *цифровая подпись*.

В этой лекции мы сначала рассмотрим некоторые проблемы, связанные с цифровыми подписями, а затем перейдем к рассмотрению различных *схем цифровой подписи*.

**3.1. Сравнение**

Рассмотрение начнем с отличия между обычными подписями и цифровыми подписями.

**Включение**

Обычная подпись включена в документ; это - часть документа. Когда мы пишем чек, подпись находится на чеке; это - не отдельный документ. Но когда мы подписываем документ в цифровой форме, мы передаем подпись как отдельный документ. Передатчик передает два документа: сообщение и подпись. Получатель получает оба документа и проверяет, что подпись принадлежит предполагаемому передатчику. Если это доказано, сообщение сохраняется; иначе оно отклоняется.

**Метод проверки**

Второе отличие между двумя типами подписей - метод подтверждения подписи. В случае обычной подписи, когда получатель получает документ, он сравнивает подпись на документе с подписью в архиве. Если они одинаковые, документ подлинный. Получатель должен иметь в архиве копию для сравнения этой подписи. При цифровой подписи - получатель получает сообщение и подпись, но копия подписи не хранится нигде. Получатель должен применить методику проверки комбинации сообщения и подписи, чтобы проверить подлинность.

**Отношения**

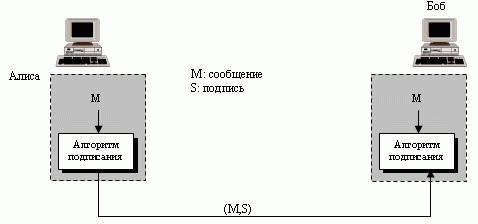
В случае обычной подписи есть отношения "один ко многим" между подписью и документами. Человек использует ту же самую подпись, чтобы подписать много документов. Для цифровой подписи есть непосредственные отношения между подписью и сообщением. Каждое сообщение имеет свою собственную подпись. Подпись одного сообщения не может использоваться в другом сообщении. Если Боб получает два сообщения, один за другим, от Алисы, он не может использовать подпись первого сообщения, чтобы проверить второе. Каждое сообщение нуждается в новой подписи.

**Резервное копирование**

Другая разность между двумя типами подписей - качество, названное *резервное копирование*. При обычной подписи ее копия на поступившем документе может отличаться от копии, сохраненной в архиве. В цифровой подписи нет такого различия, если нет указателя времени (такого как метка времени) на документе. Например, предположите, что Алиса передает документ, инструктирующий Боба заплатить Еве. Если Ева перехватывает документ и сохраняет подпись, она может воспользоваться этим позже (скопировать подпись), чтобы снова получить деньги от Боба.

**3.2. Процесc**

[Рис. 3.1](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=1#image.3.1) показывает процесс цифровой подписи. Передатчик использует *алгоритм подписания*, чтобы подписать сообщение. Сообщение и подпись передают приемнику, приемник получает сообщение и подпись и применяет *алгоритм* подтверждения к комбинации "сообщение - подпись". Если результат истинен, сообщение принято; иначе - отклонено.



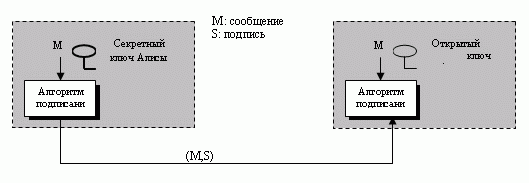
**Рис. 3.1.**Процесс передачи цифровой подписи

**Потребность в ключах**

Обычная подпись походит на частный "*ключ*", принадлежащий лицу, которое подписывает документ. Подписывающее лицо использует его, чтобы подписать документы; никто другой не имеет эту подпись. Копия подписи находится в архиве подобно открытому ключу; любой может использовать его, чтобы проверить документ, сравнить подпись с первоначальной подписью.

В цифровой подписи подписывающее лицо использует свой секретный *ключ* - прикладной *алгоритм подписания* - для подписания документа. С другой стороны проверяющий использует открытый *ключ* подписывающего лица - прикладной *алгоритм* подтверждения - для того, чтобы проверить документ.

Мы можем добавить секретный и открытый ключи к [рис. 3.1](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=1#image.3.1), чтобы дать законченное понятие цифровой подписи (см. [рис. 3.2](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=1#image.3.2)). Обратите внимание, что когда документ подписан, любой, включая Боба, может проверить это, потому что каждый имеет *доступ* к открытому ключу Алисы. Алиса не должна использовать свой открытый *ключ* для того, чтобы подписывать документ, потому что тогда любой сможет подделать ее подпись.



**Рис. 3.2.**Дополнение передачи цифровой подписи ключом Алисы

Можем ли мы использовать секретный (симметричный) *ключ*, чтобы и подписать и проверить подпись? Ответ отрицателен по нескольким причинам.

Первая: *ключ* засекречивания известен только двум объектам (например, Алисе и Бобу). Так, если Алиса должна подписать другой документ и передать его Тэду, она должна использовать другой *ключ* засекречивания.

Вторая: как мы увидим, создавая *ключ* засекречивания для сеанса, в него включают признаки установления подлинности, которые используют цифровую подпись. Мы получим порочный круг.

Третье: Боб может использовать *ключ* засекречивания, который он использует между собой и Алисой, подписать документ, передать его Тэду и имитировать, что он прибыл от Алисы.

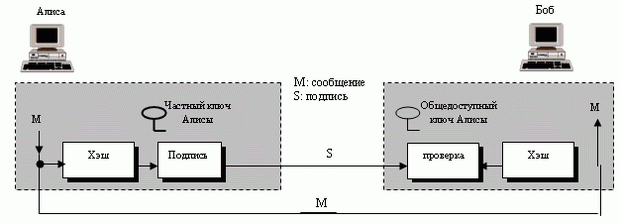
*Цифровая подпись* **нуждается в применении системы открытого ключа. Подписывающее лицо подписывается своим секретным ключом; принимающий проверяет его общедоступным ключом подписывающего лица**.

Мы должны понимать различие между секретным и открытым ключами, используемыми в цифровых подписях, и секретным и открытым ключами, которые применяются в *криптографической системе* для конфиденциальности. В последних секретный и открытый ключи приемника используются в процессе. Передатчик применяет открытый *ключ* приемника, чтобы зашифровать сообщение; приемник использует свой собственный секретный *ключ*, чтобы расшифровать сообщение. В цифровой подписи нужен и секретный и открытый ключи передатчика. Передатчик использует свой секретный *ключ*, приемник использует открытый *ключ* передатчика.

**Криптографическая система использует секретный и открытый ключи приемника**, *цифровая подпись* **- секретный и открытый ключи передатчика**.

**Подписание дайджеста**

В ["B. Стандарты и организации по стандартизации"](https://www.intuit.ru/studies/courses/553/409/lecture/17859)мы узнали, что криптосистемы с асимметричными ключами очень неэффективны, когда имеют дело с длинными сообщениями. В системе цифровой подписи сообщения обычно длинные, но мы должны использовать асимметрично-ключевые схемы. Решение состоит в том, что подпись *дайджеста сообщения* - намного короче, чем сообщение. Как мы узнали в ["С. Набор протоколов TCP/IP"](https://www.intuit.ru/studies/courses/553/409/lecture/17860), тщательно выбранный *дайджест сообщения* имеет непосредственное отношение к сообщению. Передатчик может подписать *дайджест сообщения*, а приемник может проверить *дайджест сообщения*, - эффект тот же самый. [рис. 3.3](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=1#image.3.3) показывает подписание дайджеста в системе цифровой подписи.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_03.gif)  
**Рис. 3.3.**Подписание дайджеста

Дайджест получается из сообщения на стороне Алисы и проходит процесс подписания, используя секретный ключ Алисы. Алиса затем передает сообщение и подпись Бобу. Как мы увидим позже в этой лекции, есть варианты в процессе подписания, которые зависят от системы. Например, могут быть проведены дополнительные вычисления, прежде чем получается дайджест, или может быть использована другая система засекречивания. В некоторых системах подпись имеет множество значений.

На стороне Боба используется та же самая общедоступная хэш-функция, и сначала из полученного сообщения создается дайджест. Далее вычисляются подпись и дайджест. Процесс подтверждения также проверяет прикладные критерии по результатам вычисления, чтобы определить подлинность подписи. Если подпись подлинна, то сообщение принимается; в противном случае она отклоняется.

**3.3. Услуги**

Мы обсуждали несколько услуг безопасности в ["Целостность сообщения и установление подлинности сообщения"](https://www.intuit.ru/studies/courses/553/409/lecture/9375), включая *конфиденциальность сообщения, установление подлинности сообщения, целостность сообщения* и *исключение отказа от сообщения*. *Цифровая подпись* может непосредственно обеспечить последние три; для конфиденциальности сообщения мы все еще нуждаемся в шифровании/дешифровании.

**Установление подлинности сообщения**

Безопасность *схемы цифровой подписи*похожа на безопасность обычной подписи (например, она не может быть легко скопирована) и способна обеспечить установление подлинности сообщения (также иногда называется установлением подлинности происхождения данных). Боб может проверить, что сообщение передала Алиса, потому что при проверке используется общедоступный ключ Алисы. Общедоступный ключ Алисы не сможет проверить подпись, подписанную секретным ключом Евы.

*Цифровая подпись* **обеспечивает установление подлинности сообщения**.

**Целостность сообщения**

Целостность сообщения сохраняется, даже если мы подписываем все сообщение, потому что мы не можем получить ту же самую подпись, если сообщение изменено. *Схемы цифровой подписи*сегодня используют хэш-функцию при подписании и подтверждении, - алгоритмы, которые сохраняют целостность сообщения.

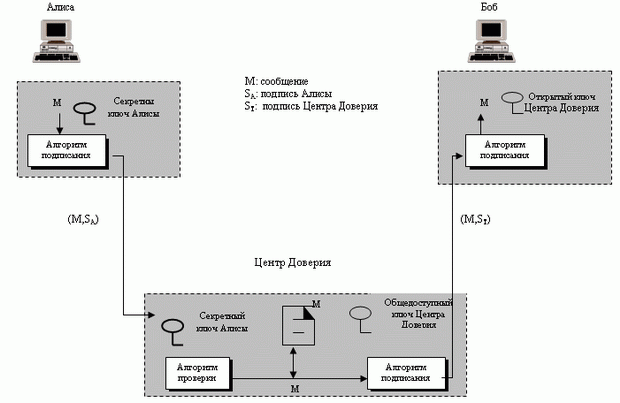
*Цифровая подпись* **обеспечивает целостность сообщения**.

**Исключение отказа от сообщения**

Если Алиса подписывает сообщение и затем отрицает это, может ли Боб доказать, что Алиса фактически подписала его? Например, Алиса передает сообщение банку (Боб) и просит перечислить 10000 $ с ее счета на счет Тэда. Может ли Алиса потом отрицать, что она передала это сообщение? Согласно схеме, которую мы рассматривали до сих пор, Боб мог бы иметь проблему. Боб должен был сохранить подпись в архиве и далее использовать открытый ключ Алисы, чтобы создать первоначальное сообщение, и доказать, что сообщение в архиве и недавно полученное сообщение является одним и тем же. Это невыполнимо, потому что Алиса может изменить за это время свой секретный или открытый ключ. Она может также утверждать, что сообщение в архиве, содержащее подпись, не подлинное.

Одно из решений - третье лицо (доверенное лицо). Люди могут по договоренности иметь сторону, которой они доверяют. В будущих лекциях мы увидим, что сторона, которой доверяют, может решить много других проблем относительно служб безопасности и замены ключей. [рис. 3.4](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=2#image.3.4) показывает, как сторона, которой доверяют, может препятствовать отрицанию Алисой передачи данного сообщения.

Алиса создает подпись из своего сообщения ( SA ) и передает центру сообщение, которое содержит ее опознавательные признаки, опознавательные признаки Боба, а также подпись. Центр, после проверки правильности открытого ключа Алисы, проверяет с помощью этого ключа Алисы, что сообщение прибыло от Алисы. Затем Центр сохраняет копию сообщения с опознавательными признаками передатчика, опознавательными признаками получателя, а также с меткой времени, в своем архиве. Центр использует свой секретный ключ, чтобы создать из сообщения другую подпись ( ST ). Затем центр передает сообщение, новую подпись, опознавательные признаки Алисы и опознавательные признаки Боба - Бобу. Боб проверяет сообщение, используя общедоступный ключ центра, которому он доверяет.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_04.gif)

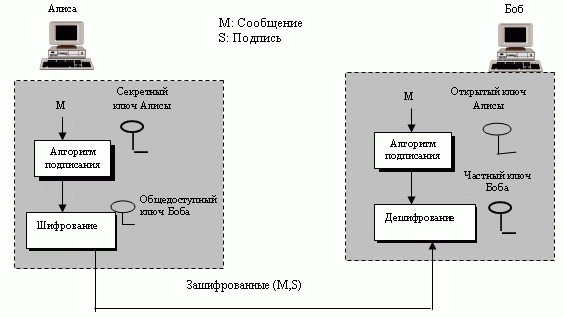
[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_04.gif)  
**Рис. 3.4.**Использование Центра Доверия для исключения отказа от сообщения

Если в будущем Алиса отрицает, что она передала сообщение, центр может предъявить копии сохраненного сообщения. Если сообщение Боба - дубликат сообщения, сохраненного в Центре, Алиса проиграет спор. Чтобы обеспечить всему этому конфиденциальность, можно добавить к схеме уровень шифрования/дешифрования, как это будет показано в следующей секции.

**Исключение отказа от сообщения может быть обеспечено участием третьей стороны, которой доверяют**.

**Конфиденциальность**

*Цифровая подпись* не обеспечивает конфиденциальную связь. Если конфиденциальность требуется, то сообщение и подпись должны быть зашифрованы с использованием любого ключа засекречивания (криптосистема с открытым ключом). [Рисунок 3.5](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=2#image.3.5) показывает, как этот дополнительный уровень можно добавить к простой *схеме цифровой подписи*.



**Рис. 3.5.**Дополнение конфиденциальности к схеме цифровой подписи

Мы показали асимметрично-ключевое шифрование/дешифрование только для того, чтобы обратить ваше внимание на типы ключей, используемые в каждом конце передачи. Шифрование/дешифрование может также быть сделано симметричным ключом.

**Цифровая подпись не обеспечивает секретность. Если есть потребность в секретности, должен быть применен уровень шифрования/дешифрования**.

**3.4. Атаки цифровой подписи**

Эта секция описывает некоторые атаки цифровых подписей и определяет типы подделки.

**Типы атаки**

Мы рассмотрим три вида атак цифровых подписей:

* *атака только на ключ*,
* *атака при известном сообщении*,
* *атака по выбранному сообщению*.

*Атака только на ключ*

В *атаке только на ключ* Ева имеет доступ только к общедоступной информации, которую передает Алиса. Для того чтобы подделать сообщение, Ева должна создать подпись Алисы, чтобы убедить Боба, что сообщение прибывает от Алисы. Это можно рассматривать как атаку только зашифрованного текста, которую мы обсуждали при шифровании.

**Атака при известном сообщении**

В *атаке при известном сообщении* Ева имеет доступ к одной или более паре "подпись - сообщения". Другими словами, она имеет доступ к некоторым документам, предварительно подписанным Алисой. Ева пробует создать другое сообщение и подделать подпись Алисы. Это подобно атаке знания исходного текста, которую мы обсуждали при шифровании.

*Атака по выбранному сообщению*

В *атаке по выбранному сообщению* Ева так или иначе заставила Алису подписать одно или более сообщений для нее. Ева теперь имеет пару **"** выбранное сообщение / подпись". Через некоторое время она создает другое сообщение, с содержанием, которое она выбирает в своих интересах, и подделывает подпись Алисы. Это подобно атаке с выборкой исходного текста, которую мы обсуждали при шифровании.

**Типы подделки**

Если атака успешна, то в результате появляется подделка. Мы можем иметь два типа подделки: *экзистенциальная* и *селективная*.

**Экзистенциальная подделка**

В *экзистенциальной подделке* Ева способна создать правильную пару "подпись - сообщение", но ни одну из подделок она не может реально использовать. Другими словами, документ был подделан, но содержание восстановлено беспорядочно. Этот тип подделки вероятен, но, к счастью, Ева не может извлечь из этого выгоду. Ее подделанное сообщение синтаксически или семантически непонятно.

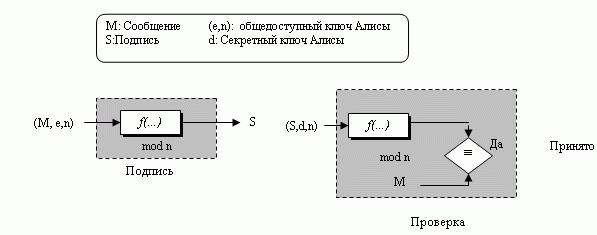
**Селективная подделка**

В *селективной подделке* Ева способна подделать подпись Алисы на сообщении с содержанием, выбранным Евой. Это выгодно для Евы и может быть очень вредно для Алисы, но вероятность такой подделки мала, хотя имеет реальную величину.

Несколько *схем цифровой подписи* были распространены в течение прошлых нескольких десятилетий. Некоторые из них были реализованы. В этой секции мы поговорим об этих схемах. В следующей секции мы обсуждаем одну, которая, вероятно, станет стандартом.

**Схема цифровой подписи RSA**

В ["B. Стандарты и организации по стандартизации"](https://www.intuit.ru/studies/courses/553/409/lecture/17859)мы обсуждали, как использовать криптографическую систему RSA для обеспечения секретности. Идея RSA может также применяться для того, чтобы подписать и подтвердить сообщение. В этом случае это называется *схемой цифровой подписи RSA*. *Схема цифровой подписи* меняет роли секретных и открытых ключей. Первое: используются секретный и открытый ключи передатчика, а не приемника. Второе: передатчик использует свой собственный секретный ключ для подписи документа; приемник использует открытый ключ передатчика, чтобы проверить этот документ. Если мы сравним схему с обычным способом подписи, мы видим, что секретный ключ играет роль передатчика собственной подписи, а открытый ключ - роль передатчика копии подписи, которая является общедоступной. Очевидно, Алиса не может использовать открытый ключ Боба, чтобы подписать сообщение, потому что тогда любой другой человек мог бы сделать то же самое. [рис. 3.6](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=3#image.3.6) дает общую идею *схемы цифровой подписи RSA*.



**Рис. 3.6.**Общая идея схемы цифровой подписи RSA

Подписание и подтверждение сайтов используют ту же самую функцию, но с различными параметрами. Верификатор сравнивает сообщение и вывод функции для сравнения. Если результат истинен, сообщение принято.

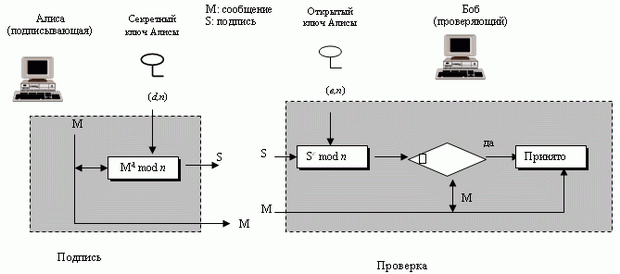
**Генерация ключей**

Генерация ключей в *схеме цифровой подписи RSА* точно такая же, как и генерация ключей в *криптографической системе* RSА (см. ["B. Стандарты и организации по стандартизации"](https://www.intuit.ru/studies/courses/553/409/lecture/17859)). Алиса выбирает два простых числа p и q и вычисляет n = p x q. Алиса вычисляет \varphi  (n) = (p - 1) (q - 1). Затем она выбирает e, для общедоступного ключа и вычисляет d для частного ключа, такое, что e x d = 1 mod \varphi \  (n). Алиса сохраняет d и публично объявляет n и e.

В *схеме цифровой подписи RSA*. d **является частным;**. e **и**. n **- общедоступными**.

**Подписание и проверка**

[Рис 3.7](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=3#image.3.7) показывает *схему цифровой подписи RSA*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_07.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_07.gif)  
**Рис. 3.7.**Схема цифровой подписи RSA

**Подписание**. Алиса на основе сообщения создает подпись, используя частный (секретный) ключ, S = Mdmod n, и передает сообщение и подпись Бобу.

**Проверка**. Боб получает М и S. Он применяет общедоступный ключ Алисы к подписи, чтобы создать копию сообщения М' = Se mod n. Боб сравнивает значение М' со значением М. Если два значения совпадают, Боб принимает сообщение. Чтобы доказать правильность этой процедуры, мы применяем критерии проверки:

M' \equiv  M(mod \ n) \to  S^{e} \equiv  M(mod \ n) \to  M^{d x e} \equiv  M(mod \ n)

Последняя конгруэнтность справедлива, потому что d x e = 1 mod \ \psi (n) (см. теорему Эйлера в ["A. ASCII"](https://www.intuit.ru/studies/courses/553/409/lecture/17858)).

**Пример 3.1**

Для безопасности подписи значения p и q должны быть очень большими. Как тривиальный пример, предположим, что Алиса выбирает p = 823 и q = 953 и вычисляет n = 784319. Значение \varphi  (n) - 782544. Теперь она выбирает e = 313 и вычисляет d = 160009. В этой точке генерация ключей закончена. Теперь вообразим, что Алиса хочет передать сообщение со значением M = 19070 Бобу. Она использует свой частный ключ 160009 для того, чтобы подписать сообщение:

M = 19070 -> S = (19070160009) mod 784319 = 210625 mod 784319

Алиса передает сообщение и подпись Бобу. Боб получает сообщение и подпись. Он вычисляет

М' = 210625313 \ mod \ 784319 = 19070 \ mod \ 784319 \ ? \ М 
\\
 М' \equiv \ mod \ n

Боб принимает сообщение, потому что он проверил подпись Алисы.

**Атаки подписи RSA**

Есть некоторые атаки, к которым Ева может обратиться для подделки *схемы цифровой подписи RSА*Алисы.

*Атака только на ключ*. Ева имеет доступ только к открытому ключу Алисы, перехватывает пару ( М, S ) и пробует создать другое сообщение М.', такое, что М' = Se mod n. Эта проблема по сложности решения равна проблеме *дискретного логарифма*, которую мы рассмотрели в ["A. ASCII"](https://www.intuit.ru/studies/courses/553/409/lecture/17858). Это - *экзистенциальная подделка* и обычно бесполезна для Евы.

*Атака при известном сообщении*. Здесь Ева использует *мультипликативное свойство*RSА. Предположим, что Ева перехватила две пары подписи сообщения - (M1, S1) и (M2, S2), которые используют один и тот же секретный ключ. Если М. = M1 x M2 mod n, тогда S. = S1 x S2 mod n. Это просто доказать, потому что мы имеем

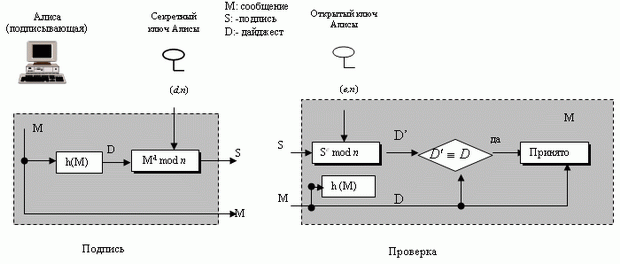
S = S1 x S2 mod n = (M1d x M2d) mod n = (M1 x M2)d mod n = Md mod n

Ева может создать М. = M1 x M2 mod n и может создать S. = S1 x S2 mod n ; глупый Боб поверит, что S - подпись Алисы на сообщении М. Эта атака, которая называется иногда *мультипликативной атакой*, проводится очень просто. Однако это - *экзистенциальная подделка* так, как сообщение, М является произведением двух предыдущих сообщений, созданных Алисой, а не Евой; сообщение М. обычно бесполезно.

*Атака по выбранному сообщению*. Эта атака также использует мультипликативное свойство RSA. Ева может так или иначе попросить, чтобы Алиса подписала два законных сообщения М1 и М2. С помощью их она позже создает новое сообщение М. = M1 x M2 . Ева может позже утверждать, что Алиса подписала M. Такую атаку называют так же, как и предыдущую - *мультипликативная атака*. Это очень серьезная атака *схемы цифровой подписи RSA*, потому что это - *селективная подделка*. (Ева может перемножать М1 и М2, чтобы получить полезный M ).

**Подпись RSA на дайджесте сообщения**

Как мы обсуждали прежде, подписание *дайджеста сообщения* и использование сильного *алгоритма хэширования* имеет несколько преимуществ. В случае RSA процессы подписания и проверки можно сделать намного быстрее, потому что *схема цифровой подписи RSА* - не что иное, как шифрование с секретным ключом и дешифрование с открытым ключом. Использование сильной графической функции хэширования также делает атаку подписи намного труднее, что мы коротко объясним. [рис. 3.8](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=3#image.3.8) иллюстрирует схему.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_08.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_08.gif)  
**Рис. 3.8.**Подпись RSА на дайджесте сообщения

Алиса, подписывающее лицо, при первом использовании согласует хэш-функцию, чтобы создать *дайджест сообщения* D = h(M). Затем она подписывает дайджест, S = Dd mod n. Сообщение и подпись передают Бобу. Боб, проверяющий, получает сообщение и подпись. Он сначала использует открытый ключ Алисы, чтобы извлечь (файл) дайджест, D' = Se mod n. Затем он применяет хэш-алгоритм для того, чтобы получить сообщение D = h(M).

Боб теперь сравнивает эти два дайджеста, D и D'. Если они являются сравнимыми по модулю n, он принимает сообщение.

**Атаки на подписанные дайджесты RSA**

Насколько восприимчива *схема цифровой подписи RSA* к нападению, когда дайджест уже подписан?

*Атака только на ключ*. Мы можем иметь три случая этой атаки.

1. Ева перехватывает пару (S, M) и пробует найти другое сообщение М', которое создает тот же самый дайджест, h (M) = h (М'). Как мы узнали в ["Целостность сообщения и установление подлинности сообщения"](https://www.intuit.ru/studies/courses/553/409/lecture/9375), если *алгоритм хэширования* *обладает устойчивостью ко второму прообразу*, эта атака очень трудна.
2. Ева находит два сообщения, М. и М', такие, что h(M) = h(M'). Она соблазняет Алису подписать h(M), чтобы найти S ; теперь Ева имеет пару (М', S), которое прошло подтверждающий тест, но это - подделка. Мы изучали в ["Целостность сообщения и установление подлинности сообщения"](https://www.intuit.ru/studies/courses/553/409/lecture/9375), что если *алгоритм хэширования* *устойчив к коллизиям*, эта атака очень трудна.
3. Ева случайным подбором может найти *дайджест сообщения* D, который может соответствовать случайной подписи S. Она тогда находит сообщение М, такое, что D = h (M). Как мы узнали в ["Целостность сообщения и установление подлинности сообщения"](https://www.intuit.ru/studies/courses/553/409/lecture/9375), если хэш-функция *устойчива к прообразу*, эта атака очень трудно осуществима.

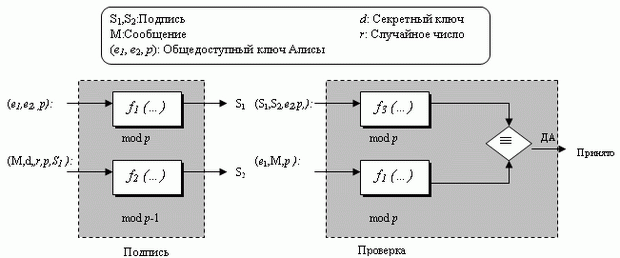
*Атака при известном сообщении*. Предположим, что Ева имеет две пары подписи сообщения - (М1,S1) и (М2, S2), которые были созданы с использованием одного и того же секретного ключа. Ева вычисляет S = S1 x S2. Если она сможет найти сообщение М, такое, что h(M) = h(M1) x h(M2), она сможет подделать новое сообщение. Однако нахождение М по данному h (M) - это очень трудный процесс, если *алгоритм хэширования* *устойчив к прообразу*.

*Атаки по выбранному сообщению*. Ева может попросить, чтобы Алиса подписала два законных сообщения - М1,и М2 для нее. Она может создать новую подпись S = S1 x S2. Так как Ева может вычислить h(M) = h(M1) x h(M2), если она может найти сообщение M данному h (M), это новое сообщение - подделка. Однако нахождение М по данному h (M) -очень трудный процесс, если *алгоритм хэширования* *устойчив к прообразу*.

**Когда дайджест подписан непосредственно вместо сообщения, восприимчивость**. *схемы цифровой подписи RSA* **зависят от свойств алгоритма хэширования**.

**Схема цифровой подписи Эль-Гамаля**

Криптосистема Эль-Гамаля была обсуждена в ["B. Стандарты и организации по стандартизации"](https://www.intuit.ru/studies/courses/553/409/lecture/17859). *Схема цифровой подписи Эль-Гамаля* использует те же самые ключи, но алгоритм различен. [рис. 3.9](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=4#image.3.9) дает общую идею *схемы цифровой подписи Эль-Гамаля*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_09.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_09.gif)  
**Рис. 3.9.**Общая идея схемы цифровой подписи Эль-Гамаля

В процессе подписания две функции создают две подписи. На стороне подтверждения обрабатывают выходы двух функций и сравнивают между собой для проверки. Обратите внимание, что одна и та же функция применяется и для подписания, и для проверки, но использует различные входы. Рисунок показывает входы каждой функции. Сообщение - часть входа, для обеспечения функционирования при подписании; оно же - часть входа к функции 1 при подтверждении. Обратите внимание, что вычисления в функциях 1 и 3 проводятся по модулю p, а функции 2 - по модулю p - 1.

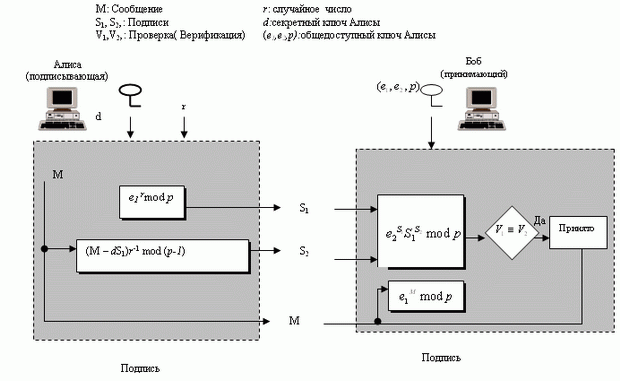
**Генерация ключей**

Процедура генерации ключей здесь точно такая же, как та, которая используется в *криптографической системе*. Выберем достаточно большое простое число p, чтобы в поле Zp\* проблема *дискретного логарифма* была достаточно трудной. Пусть e1 - простой элемент в Zp\*. Алиса выбирает свой секретный ключ d, чтобы он был меньше, чем p - 1. Она вычисляет e2 = e1d. Открытый ключ Алисы - кортеж (e1, e2, p) ; секретный ключ Алисы - d.

**В** *схеме цифровой подписи Эль-Гамаля* (e1, e2, p) **- открытый ключ Алисы;**d **-секретный ключ Алисы**.

**Подтверждение и проверка**

[Рисунок 3.10](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=4#image.3.10) показывает *схему цифровой подписи Эль-Гамаля*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_10.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_10.gif)  
**Рис. 3.10.**Схема цифровой подписи Эль-Гамаля

**Подписывающаяся** Алиса может подписать *дайджест сообщения*, направленный к любому объекту, включая Боба.

1. Алиса выбирает секретное случайное число r. Обратите внимание, что хотя открытые и секретные ключи могут использоваться неоднократно, Алиса каждый раз нуждается в новом r, когда она подписывает новое сообщение.
2. Алиса вычисляет первую подпись S1= er mod p.
3. Алиса вычисляет вторую подпись S2= (М - d x S1) x r-1 mod (p - 1),где r  - мультипликативная инверсия r по модулю p - 1.
4. Алиса передает М, S1 и S2 Бобу.

**Проверка**. Объект, например Боб, получает М, S1 и S2 и может проверить их следующим образом.

1. Боб проверяет, что 0 < S1 < p.
2. Боб проверяет, что 0 < S2 < p - 1.
3. Боб вычисляет V1 = e1M mod p.
4. Боб вычисляет V2 = e2S1 x e2S2mod p.
5. Если V1 является конгруэнтным V2, сообщение принято; иначе оно будет отклонено. Мы можем доказать правильность этого критерия проверки, используя e2 = e1d и S1 = e1r:

V_{1} \equiv  V_{2} (mod \ p) \to  e_{1}^{M}  \equiv  e_{2}^{M} \times S_{1}^{M} mod \ p  \equiv  (e_{1}^{d1})^{S1} \times  (e_{1}^{r})^{S2} (mod \ p)  \equiv  e_{1}^{dS1+rS2} mod \ p

Мы имеем e_{1}^{M } \equiv  e_{1}^{dS1 + rS2} mod \ p

Поскольку e1 - *первообразный корень*, может быть доказано, что вышеупомянутое сравнение справедливо тогда и только тогда, когда M \equiv  [dS_{1} + rS_{2}] mod \  (p-1)  S_{2} \equiv  [(M - d  \times  S_{1}) \times r^{-1}] mod \ (p-1) , и результат сравнения есть тот же самый S2, с которого мы начали процесс подписания.

**Пример 3.2**

Ниже приводится тривиальный пример. Алиса выбрала p = 3119, e1= 2, d = 127 и вычислила e2= 2127 mod 3119 = 1702. Она выбрала r равным 307. Она объявила e1, e2 и p ; она сохранила в тайне d. Далее показано, как Алиса может подписать сообщение.

M = 320

S1 = e1r = 2307 = mod3119

S2 = (M - d x S1) x r-1 = (320 - 127 x 2083) x 307-1 = 2105 mod 3118

Алиса передает М, S1 и S2 Бобу. Боб использует открытый ключ, чтобы вычислить, что сообщение подписано Алисой, потому что никто, кроме Алисы, не имеет секретного ключа d.

V1 = e1M = 2320 = 3006 mod 3119

V2 = dS1 x S2S1 = 17022083 x 20832105 = 3006 mod 3119

Поскольку V1 и V2 являются конгруэнтными, Боб принимает сообщение, и он предполагает, что сообщение было подписано Алисой, потому что никто, кроме нее, не имеет секретного ключа Алисы d.

**Пример 3.3**

Теперь вообразите, что Алиса хочет передать другое сообщение, М = 3000, Тэду. Она выбирает новое r = 107. Алиса передает М., S1 и S2 Тэду. Тэд использует общедоступные ключи, чтобы вычислить V1 и V2.

M = 3000

S1 = e1r = 2107 = 2732mod3119

S2 = (M - d x S1) x r-1 = (3000 - 127 x 2083) x 107-1 = 2526 mod 3118

V1 = e1M = 23000 = 704 mod 3119

V2 = dS1 x S1S2 = 17022732 x 20832526 = 704 mod 3119

Поскольку V1 и V2 являются конгруэнтными, Тэд принимает сообщение; он предполагает, что сообщение подписано Алисой, потому что никто, кроме нее, не имеет секретного ключа Алисы d. Обратите внимание, что сообщение может получить любой человек. Цель состоит не в том, чтобы скрыть сообщение, но в том, чтобы доказать, что его передает Алиса.

**Подделка цифровой подписи в схеме Эль-Гамаля**

*Схема цифровой подписи Эль-Гамаля* уязвима к *экзистенциальной подделке*, но *селективную подделку* на этой схеме сделать очень трудно.

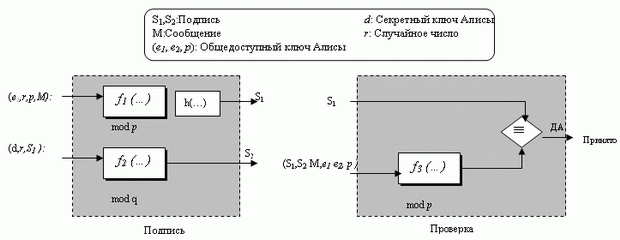
**Подделка только ключа**. В этом типе подделки Ева имеет доступ только к открытому ключу. Возможны два варианта:

1. Ева имеет заранее заданное сообщение М. Она должна подделать подпись Алисы на этом сообщении. Ева должна найти две правильных подписи S1 и S2 для этого сообщения. Это - *селективная подделка*.
   * Ева может выбрать S2 и вычислить S1. Она должна иметь d_{1}^{S1} x S_{1}^{S2} \equiv  e_{1}^{M} (mod \ p). Другими словами, S_{1}^{S2} \equiv  e_{1}^{M} d_{1}^{-S1} (mod\  p) или S_{2} \equiv  log_{S1}(e_{1}^{M} d_{1}^{-S1}) (mod \ p). Это означает вычисление *дискретного логарифма*, что является очень трудным.
   * Ева может выбрать S2 и вычислить S1. Это намного труднее, чем выполнить часть a
2. Ева может методом случайного подбора найти три значения, М, S1 и S2, такие, что подпись первого используется для второго. Если Ева может найти два новых параметра x и y, такие, что М = xS2 mod (p - 1) и S1 = -y S2 mod (p - 1), то она может подделать сообщение, но серьезной выгоды не получит, поскольку это - *экзистенциальная подделка*.

**Подделка при известном сообщении**. Если Ева перехватила сообщение М и его две подписи S1 и S2, она может найти другое сообщение М', с той же самой парой подписей S1 и S2. Однако обратите внимание, что это - *экзистенциальная подделка*, которая не помогает Еве.

**Схема цифровой подписи Шнорра**

Проблема *схемы цифровой подписи Эль-Гамаля* - в том, что p должно быть очень большим, чтобы сделать трудной проблему *дискретного логарифма* Zp\*. Рекомендуется длина p по крайней мере 1024 битов. Можно сделать подпись размером 2048 бит. Чтобы уменьшить размер подписи, Шнорр предложил новую схему, основанную на *схеме Эль-Гамаля*, но с уменьшенным размером подписи. [рис. 3.11](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=5#image.3.11) дает общую идею *схемы цифровой подписи Шнорра*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_11.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_11.gif)  
**Рис. 3.11.**Общая идея схемы цифровой подписи Шнорра

В процессе подписания две функции создают две подписи; в процессе проверки выход одной функции сравнивается с первой подписью для проверки. [рис. 3.11](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=5#image.3.11) показывает входы к каждой функции. Важно то, что схема использует два модуля: p и q. Функции 1 и 3 используют p ; функция 2 использует q. Детали входов и функций будут коротко обсуждены далее.

**Генерация ключей**

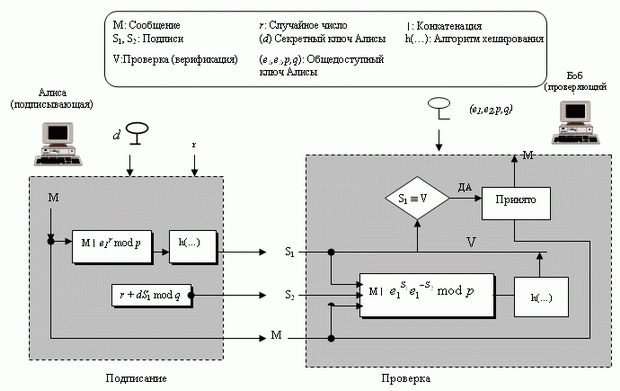
Перед подписанием сообщения Алиса должна генерировать ключи и объявить всем общедоступные ключи.

1. Алиса выбирает простое число p, которое обычно равно по длине 1024 битам.
2. Алиса выбирает другое простое число *q*, которое имеет тот же самый размер, что и дайджест, созданный функцией криптографического хэширования (в настоящее время 160 битов, но это может измениться в будущем). Простое число q должно делиться на (p - 1). Другими словами, (p - 1) = 0 mod q.
3. Алиса выбирает e1, q -тый корень которого был бы равен 1 mod p. Чтобы сделать это, Алиса выбирает примитивный элемент в Zp, e0 (см. ["G. Список неприводимых и примитивных полиномов"](https://www.intuit.ru/studies/courses/553/409/lecture/17864)) и вычисляет e1 = e0(p-1)/q mod p.
4. Алиса выбирает целое число, d, как свой секретный ключ.
5. Алиса вычисляет e2 = e1dmod p.
6. Общедоступный ключ Алисы - (e1, e2, p, q), ее секретный ключ - (d).

**В***схеме цифровой подписи Шнорра* **общедоступный ключ Алисы -**(e1, e2, p, q) ; **ее секретный ключ -**(d) .

**Подписание и проверка**

[Рисунок 3.12](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=5#image.3.12) показывает *схему цифровой подписи Шнорра*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_12.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_12.gif)  
**Рис. 3.12.**Схема цифровой подписи Шнорра

**Подписание**

1. Алиса выбирает случайное число r. Обратите внимание, что открытый и секретный ключи могут использоваться для подписи многих сообщений. Но Алиса должна изменять r каждый раз, когда она передает новое сообщение. Обратите внимание также на то, что r должен иметь значение между 1 и q.
2. Алиса вычисляет первую подпись S1= h(M | e1r mod p) . Сообщение присоединяется (конкатенируется) спереди к значению e1r mod p, затем применяется хэш-функция, чтобы создать дайджест. Обратите внимание, что хэш-функция непосредственно не применяется к сообщению, но вместо этого она получается из последовательного соединения М. и e1r mod p.
3. Алиса вычисляет вторую подпись S2 = r + d x S1 mod q. Обратите внимание, что эта часть - вычисление S2 - делается в арифметике по модулю q.
4. Алиса передает М., S1 и S2.

**Верификация (проверка) сообщения**. Приемник, например Боб, получает М., S1 и S2.

1. Боб вычисляет V = h (М | e1S2 e2-S1 mod p).
2. Если S2 конгруэнтно V по модулю p, сообщение принято; иначе оно отклоняется.

**Пример 3.4**

Вот тривиальный пример. Предположим, что мы выбираем q = 103 и p = 2267. Обратите внимание на то, что p = 22 x q + 1. Мы выбираем e0 = 2, которое является элементом в Z2267\*. Тогда (p - 1) / q = 22, так что мы имеем e1 = 222 mod 2267 = 354.

Мы выбираем d = 30, тогда e2 = 35430 mod 2267 = 1206. Секретный ключ Алисы -теперь - (d), ее общедоступный ключ - (e1,e2,p,q).

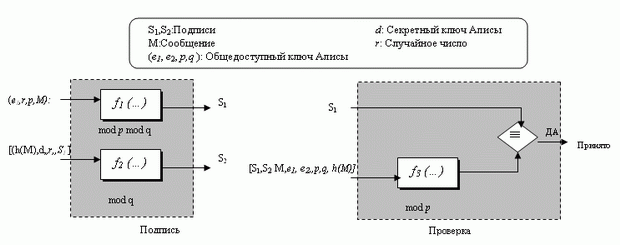
Алиса хочет передать сообщение М. Она выбирает r = 11 и вычисляет er = 35411 = 630 mod 2267. Предположим, что сообщение - 1000, и конкатенация (последовательное соединение) означает 1000630. Также предположим, что хэширование этого значения дает дайджест h (1000630) = 200. Это означает S1 = 200. Алиса вычисляет S2 = r + d x S1 mod q = 11 + 1026 200 mod 103 = 11 + 24 = 35. Алиса передает сообщение М =1000, S1= 200 и S2 = 35. Проверку оставляем как упражнение.

Подделка по *схеме подписи Шнорра*

Похоже, что все атаки на *схему Эль-Гамаля* могут быть применены к *схеме Шнорра*. Однако *схема Шнорра* находится в лучшем положении, потому что S1 = h (М. | e1r(mod p)). Это означает, что хэш-функция применяется к комбинации сообщение и e1r, в которой r является секретным.

**Стандарт цифровой подписи (DSS)**

*Стандарт цифровой подписи***(DSS - Digital Signature Standard)** был принят национальным Институтом Стандартов и Технологии (NIST) в 1994 г. NIST издал DSS как FIPS-186 (FEDERAL INFORMATION PROCESSING STANDARD 186). DSS применяет *алгоритм цифровой подписи* (*DSA*), основанный на *схеме Эль-Гамаля*, с использованием некоторых идей из *схемы Шнорра*. DSS критиковался со времени его издания. Главная претензия - оценка безопасности проекта DSS. Вторая - размер простого числа, 512 битов. Позже NIST сделал это число переменного размера, чтобы ответить на эту претензию. [рис. 3.13](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=6#image.3.13) дает общую идею схемы DSS.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_13.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_13.gif)  
**Рис. 3.13.**Общая идея схемы DSS

В процессе подписания две функции создают две подписи; в процессе проверки выход одной функции сравнивается с первой подписью для проверки. Это подобно *схеме Шнорра*, но входы различны. Другое отличие: эта схема использует *дайджест сообщения* (не само сообщение) как часть входов к функциям 1 и 3. Интересно то, что схема применяет два общедоступных модуля: p и q. Функции 1 и 3 используют оба модуля p и q, функция 2 - только q. Детали входов и функций коротко рассматриваются ниже.

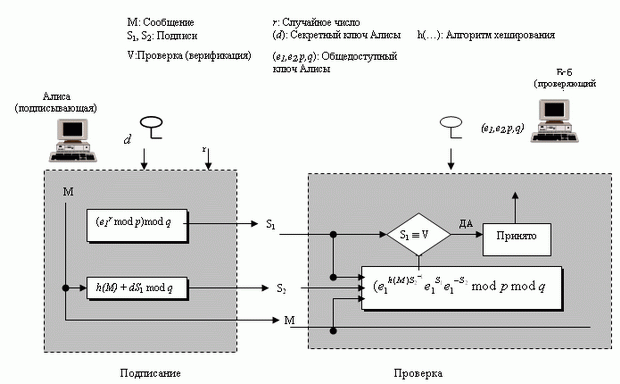
**Генерация ключей**

Перед подписанием сообщения к любому объекту Алиса должна генерировать ключи и объявить общедоступные ключи.

1. Алиса выбирает простое число длиной p. между 512 и 1024 битами. Число битов в p должно быть кратно числу 64.
2. Алиса выбирает простое число на 160 битов q с таким условием, чтобы оно делилось на (p - 1).
3. Алиса использует две группы умножения, <Zp\*, x > и <Zq\* ,x > ; вторая - подгруппа первой.
4. Алиса создает e1, такое, чтобы оно было q -тым корнем 1 по модулю p (e1p = 1 mod p). Она поступает так: выбирает элемент в Zp, e0, и вычисляет e1 = e0(p-1)/q mod p.
5. Алиса выбирает d как секретный ключ и вычисляет e2 = e1d
6. Общедоступный ключ Алисы - (e1, e2, p, q), ее секретный ключ - (d).

**Подписание и проверка**

[Рисунок 3.14](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=6#image.3.14) показывает схему DSS.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_14.gif)

[увеличить изображение](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_14.gif)  
**Рис. 3.14.**Схема DSS

**Подписание**. Ниже показаны шаги подписания сообщения.

1. Алиса выбирает случайное число (1 <= r <= q). Обратите внимание, что хотя открытые и секретные ключи могут быть выбраны один раз и использоваться для того, чтобы подписать много сообщений, Алиса должна выбирать каждый раз новый r, когда она должна подписать новое сообщение.
2. Алиса вычисляет первую подпись S1 = (e1 mod p) mod q. Обратите внимание: значение первой подписи не зависит от М (сообщения).
3. Алиса создает *дайджест сообщения* h (M).
4. Алиса вычисляет вторую подпись S2 = (h(M) + dS1)r-1mod q. Обратите внимание, что вычисление S2 делается по модулю q.
5. Алиса посылает M, S1 и S2 Бобу.

**Проверка (верификация)**. Для проверки сообщения обычно применяются следующие шаги, когда получены М, S1 и S2.

1. Боб проверяет S1 0 < S1 < q.
2. Боб проверяет S2 0 < S2 < q.
3. Боб вычисляет дайджест М, применяя *алгоритм хэширования*, используемый Алисой.
4. Боб вычисляет V = [(e1h(M)/S2 e2S1/S2) mod p] mod q.
5. Если S конгруэнтен V, сообщение принимается; иначе - отклоняется.

**Пример 3.5**

Алиса выбирает q = 101 и p = 8081. Алиса выбирает e0= 3 и вычисляет e1 = e0(p-1)/q mod p = 6968. Алиса выбирает d = 61 в качестве секретного ключа и вычисляет e2 = e1d mod p = 2038. Теперь Алиса может передать сообщение Бобу. Предположим, что h (M) = 5000, и Алиса выбирает r = 61:

h(M) = 5000 r =61

S1 = (e1r mod p) mod q = 54

S2 = (h(M) + dS1)r-1 mod q = 40

S2-1 = 48 mod 101

V = [(69685000 x 48 x 203854 x 48 x 203854 x 48)mod 8081]mod 101 = 54

Поскольку Sj и V являются подходящими, Боб принимает сообщение.

**Сравнение DSS и RSА**

Вычисление DSS-подписи быстрее, чем вычисление подписей RSА, при использовании того же самого p.

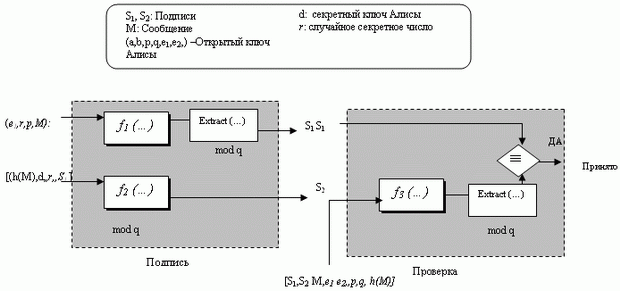
**Сравнение DSS и схемы Эль-Гамаля**

DSS-подпись - меньше, чем подписи в *схеме Эль-Гамаля*, потому что q меньше, чем p.

**Схема цифровой подписи эллиптической кривой**

Наша последняя схема - *схема цифровой подписи эллиптической кривой* **(ECDSS - Elliptic Curve Digital Signature Scheme)**, которая основана на применении *эллиптических кривых*, - их мы обсуждали в ["B. Стандарты и организации по стандартизации"](https://www.intuit.ru/studies/courses/553/409/lecture/17859). Схема иногда упоминается как ECDSA (Elliptic Curve *Digital Signature Algorithm*). [рис. 3.15](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=6#image.3.15) показывает общую идею ECDSS.

В процессе подписания две функции и экстрактор (извлекающее устройство) создают две подписи; в процессе проверки (верификации) обрабатывают выход одной функции (после прохождения через экстрактор) и сравнивают ее с первой подписью для проверки. Функции f1 и f3 фактически создают точки на кривой. Первая создает новую точку для секретного ключа подписывающего лица. Вторая - новую точку из двух общедоступных ключей подписывающего лица. Каждый экстрактор извлекает первые координаты соответствующей точки в модульной арифметике. Детали входов и функций коротко обсуждаются далее.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_15.gif)

**Рис. 3.15.**Общая идея схемы DSS

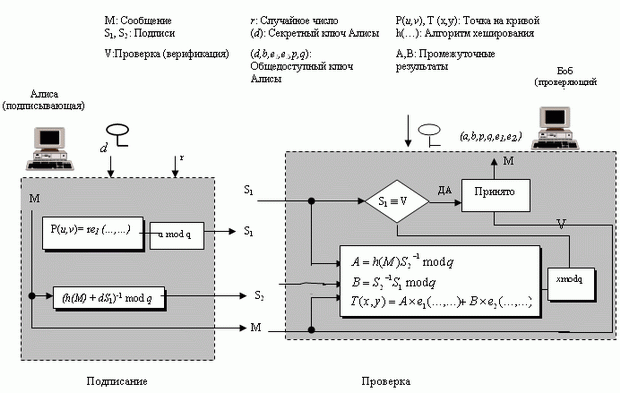
**Генерация ключей**

Генерация ключей осуществляется следующими шагами:

1. Алиса выбирает *эллиптическую кривую* *Ep* (a,b) с простым числом p.
2. Алиса выбирает другое простое число q, чтобы использовать для вычисления.
3. Алиса выбирает секретный ключ d, целое число.
4. Алиса выбирает точку на кривой e1(.,....)
5. Алиса вычисляет e2 (..,....) = d x e1 (......), другую точку на кривой.
6. Общедоступный ключ Алисы - (a, b, p, q, e1, e2), ее секретный ключ - d.

**Подписание и проверка (верификация)**

[Рис. 3.16](https://www.intuit.ru/studies/courses/553/409/lecture/9379?page=6#image.3.16) показывает *схему цифровой подписи эллиптической кривой*.

[](https://www.intuit.ru/EDI/18_11_15_6/1447798956-19904/tutorial/577/objects/3/files/03_16.gif)  
**Рис. 3.16.**Схема EDSS

**Подписание**. Процесс подписания состоит главным образом из выбора секретного случайного числа, создания третьей точки на кривой, вычисления двух подписей и передачи сообщения и подписей.

1. Алиса выбирает секретное случайное число, r, между 1 и q - 1.
2. Алиса выбирает третью точку на кривой, P (U, v) = r x e1 (......).
3. Алиса использует первые координаты P (u, v), чтобы вычислить первую подпись S1 . Это означает S1= u mod q.
4. Алиса использует *дайджест сообщения*, свой секретный ключ и секретное случайное число r и S1, чтобы вычислить вторую подпись S2 = (h (M) + d x S1) r-1 mod q,
5. Алиса передает М, S1 и S2.

**Проверка (верификация)**.Процесс проверки состоит главным образом из восстановления третьей точки и подтверждения, что первая координата эквивалентна S1 по модулю q. Обратите внимание, что третья точка была создана подписывающим лицом, использующим секретное случайное число r. Верификатор не имеет этого значения. Ему нужно создать третью точку из *дайджеста сообщения*, S1 , и S2.

1. Боб применяет М, S1 и S2 для создания двух промежуточных результатов A и B:

A = h(M)S2-1 mod q

B = S2-1S1 mod q

Затем Боб восстанавливает третью точку T(x,y) = A x e1( ... , ...) + B x e2( ... , ...)

1. Боб использует первую координату из T(x,y), чтобы проверить сообщение. Если x = S1mod q, подпись принимается, иначе - отклоняется.

**5 Подписание документов при помощи симметричных криптосистем и арбитра**

Первые варианты цифровой подписи были реализованы при помощи симметричных криптосистем, в которых абоненты, участвующие в обмене сообщениями, используют один и тоже секретный ключ для простановки и проверки подписи под документом. В качестве алгоритма криптографи­ческого преобразования может использоваться любая из симметричных криптосистем, обладающая специальными режимами функционирования (например, DES, ГОСТ 28147-89 и т.п.).

Однако, данная схема применима только в тех сетях, в которых можно дать стопроцентную гарантию надежности каждого из абонентов, т.к. в противном случае существует потенциальная возможность мошенничества со стороны одного из абонентов, владеющих секретным ключом.

Для устранения указанного недостатка была предложена схема с до­веренным арбитром.

В случае возникновения спорных ситуаций, например, связанных с от­казом отправителя от факта подписания документа, используются услуги третьей, независимой стороны, так называемого арбитра, который рассле­дует каждую такую ситуацию и принимает решение в пользу одной из сторон, участвующих в обмене данными.

Можно заметить, что самым критичным звеном этой схемы является именно арбитр. Во-первых, он должен быть действительно независимой ни от кого стороной. А во-вторых, арбитр должен быть абсолютно без­ошибочным. Ошибка в рассмотрении даже одной из нескольких тысяч спорных ситуаций подорвет доверие не только к арбитру, но и ко всем предыдущим подписанным документам, достоверность которых удостове­рялась арбитром.

Именно поэтому, несмотря на теоретическую возможность применения симметричных криптосистем, на практике для выработки подписи они не используются, поскольку требуют дорогостоящих и громоздких меропри­ятий по поддержанию достаточного уровня достоверности передаваемых данных.