# РАЗДЕЛ 5

# Вопросы безопасности вычислительных систем

# Безопасная система обладает свойствами:

- *Конфиденциальности* – гарантия того, что секретные данные будут доступны только тем пользователям, которым этот доступ разрешен (авторизованные пользователи);

- *Доступности* – гарантия того, что авторизованные пользователи всегда получат доступ к данным;

- *Целостности* – гарантия сохранности данными правильных значений, которая обеспечивается запретом для неавторизованных пользователей каким-либо образом модифицировать, разрушать или создавать данные.

**Угроза** – любое действие, направленное на нарушение конфиденциальности, целостности и/или доступности информации, а также нелегальное использование других ресурсов сети.

**Атака** – реализованная угроза.

**Риск** – вероятностная оценка величины возможного ущерба, нанесенного успешно проведенной атакой.

# Классификация угроз безопасности

# Средства обеспечения безопасности:

- ✔ Морально-этические
- ✔ Законодательные
- ✔ Административные
- ✔ Психологические

# Политика безопасности

- Какую информацию защищать?
- Какой ущерб понесет предприятие при потере или раскрытии тех или иных данных?
- Кто или что является возможным источником угрозы, какие атаки возможны в системе?
- Какие средства использовать для защиты каждого вида информации?

# Базовые принципы:

- Минимальный уровень привилегий
- Комплексный подход к обеспечению безопасности
- Баланс надежности защиты всех уровней
- Использование средств, переходящих при отказе в состояние максимальной защиты
- Единый контрольно-пропускной пункт
- Баланс возможного ущерба от реализации угрозы и затрат на ее предотвращение

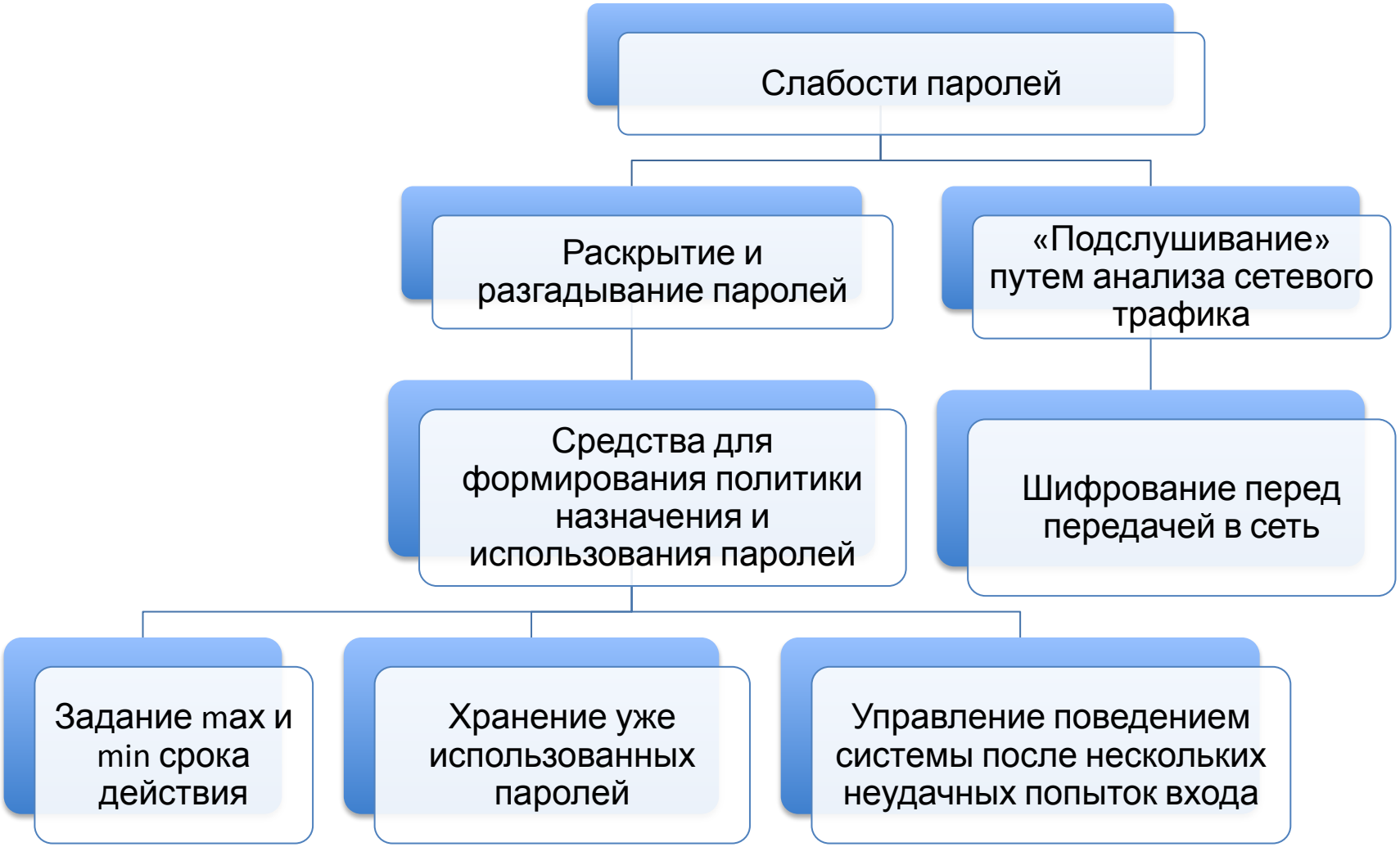
# Базовые технологии безопасности

- Аутентификация
- Авторизация
- Аудит
- Технология защищенного канала

# Аутентификация – установление подлинности

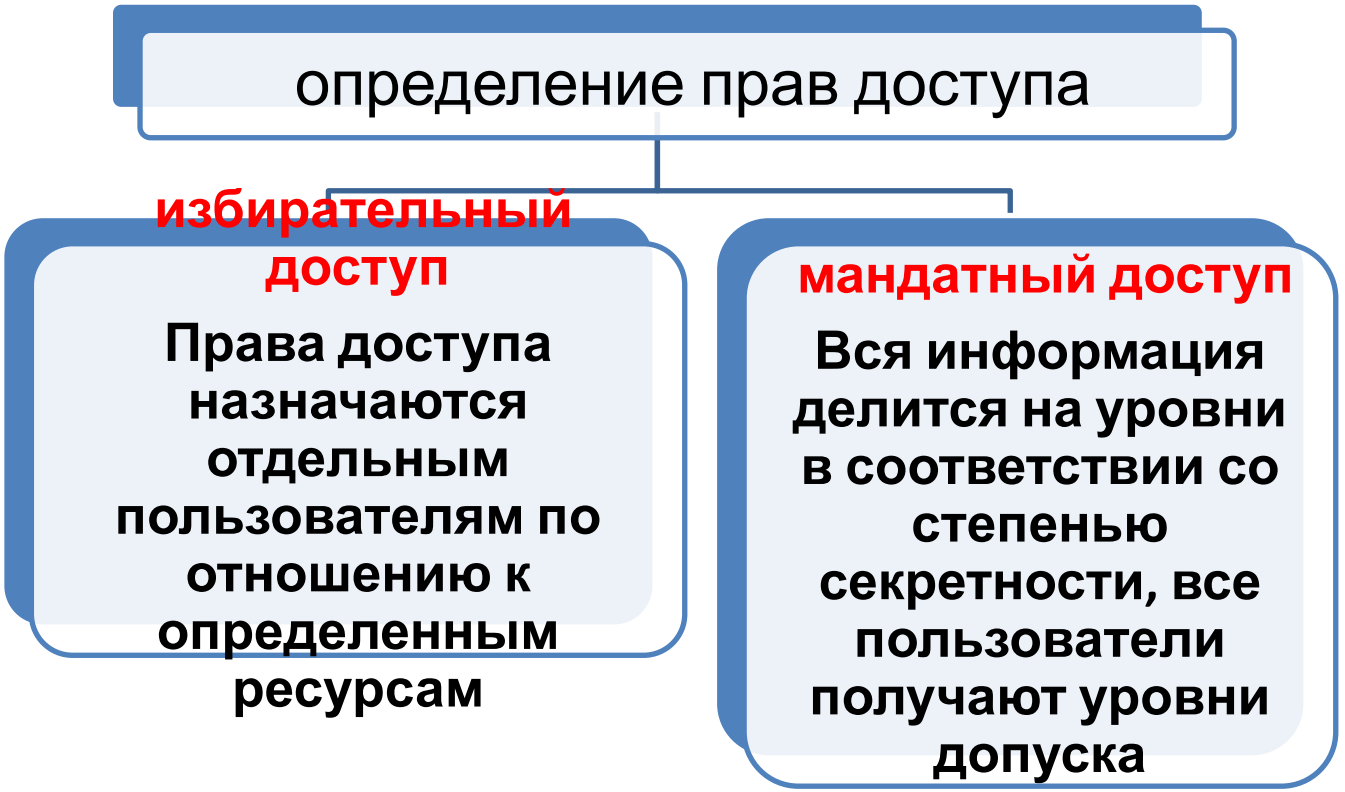
- Предотвращает доступ к сети нежелательных лиц и разрешает вход для легальных пользователей
- Для доказательства аутентичности можно использовать:

- знание некоего общего секрета: слова (пароля) или факта;
- владение неким уникальным предметом (физическим ключом);
- различные биохарактеристики: отпечатки пальцев, рисунок радужной оболочки глаз)


Слабости паролей
Раскрытие и разгадывание паролей
«Подслушивание» путем анализа сетевого трафика
Средства для формирования политики назначения и использования паролей
Шифрование перед передачей в сеть
Задание max и min срока действия
Хранение уже использованных паролей
Управление поведением системы после нескольких неудачных попыток входа

# Авторизация доступа

- Средства авторизации контролируют доступ легальных пользователей к ресурсам системы, предоставляя каждому из них именно те права, которые ему были определены администратором.



# Аудит

фиксация в системном журнале событий, связанных с безопасностью

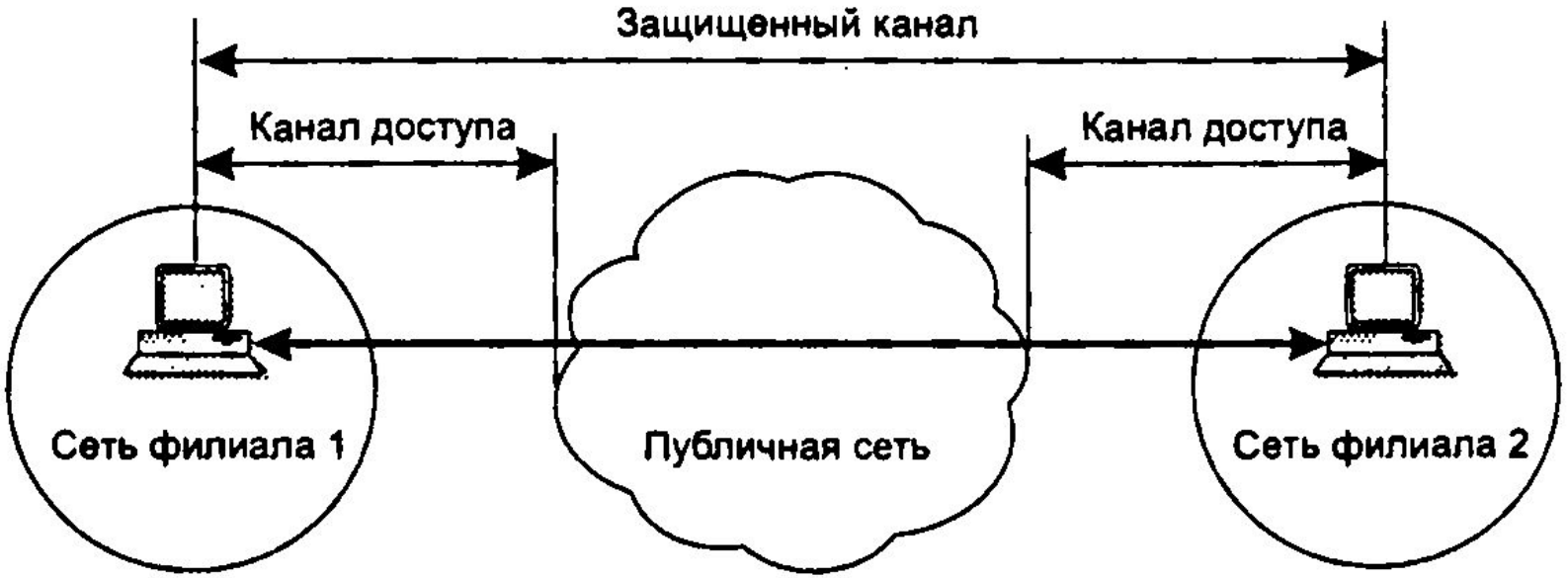
# Технология защищенного канала

используется для обеспечения безопасности передачи данных в публичных сетях.

Выполняет три основные функции:

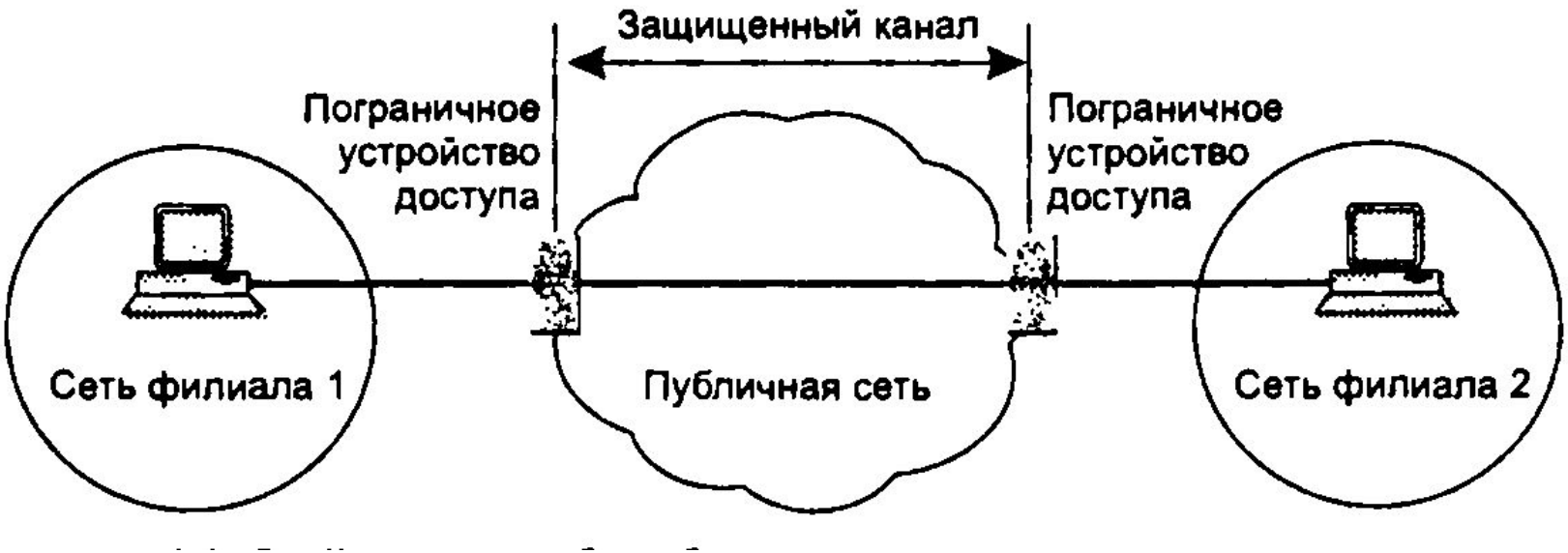
- взаимную аутентификацию абонентов при установлении соединения, которая может быть выполнена, например, путем обмена паролями;
- защиту передаваемых по каналу сообщений от несанкционированного доступа, например, путем шифрования;
- подтверждение целостности поступающих по каналу сообщений, например, путем передачи одновременно с сообщением его дайджеста.

# Схема с конечными узлами, взаимодействующими через публичную сеть




*а*

# Схема с оборудованием поставщика услуг публичной сети, расположенным на границе между частной и публичной сетью



# Шифрование

- Криптосистема – пара процедур шифрование + дешифрирование.
- Современные алгоритмы шифрования предусматривают наличие параметра – секретного ключа.

Правило Керкхоффа:

«Стойкость шифра должна определяться только секретностью ключа».

Алгоритм шифрования считается раскрытым, если найдена процедура, позволяющая подобрать ключ за реальное время. Сложность алгоритма раскрытия называется криптостойкостью.


Криптосистемы
Симметричные (классическая криптография)
Секретный ключ зашифровки совпадает с секретным ключом расшифровки
Асимметричные (криптография с открытым ключом)
Открытый ключ зашифровки не совпадает с секретным ключом расшифровки

# Симметричные алгоритмы шифрования


Модель симметричного шифрования

Теоретические основы классической модели симметричной криптосистемы были изложены Клодом Шенноном в 1949 году. Модель является универсальной – если зашифрованные данные никуда не передаются, отправитель и получатель совмещаются в одном лице, а в роли злоумышленника выступает некто, имеющий доступ к компьютеру в отсутствии владельца.

# Стандартный симметричный алгоритм шифрования DES (Data Encryption Standard)

Разработан IBM и в 1976 году рекомендован Национальным бюро стандартов к использованию в открытых секторах экономики


Схема шифрования по алгоритму DES

Данные шифруются поблочно. На вход шифрующей функции поступает блок данных размером 64 бита, он делится пополам на левую (L) и правую (R) части.

1. На место левой части результирующего блока помещается правая часть исходного блока.
2. Правая часть результирующего блока вычисляется как сумма по модулю два левой и правой части исходного блока.
3. На основе случайной двоичной последовательности по определенной схеме в полученном результате выполняются побитные замены и перестановки.

Используемая двоичная последовательность имеет длину 64 бита, из которых 56 действительно случайны, а 8 предназначены для контроля. Эта последовательность и является *КЛЮЧОМ.*

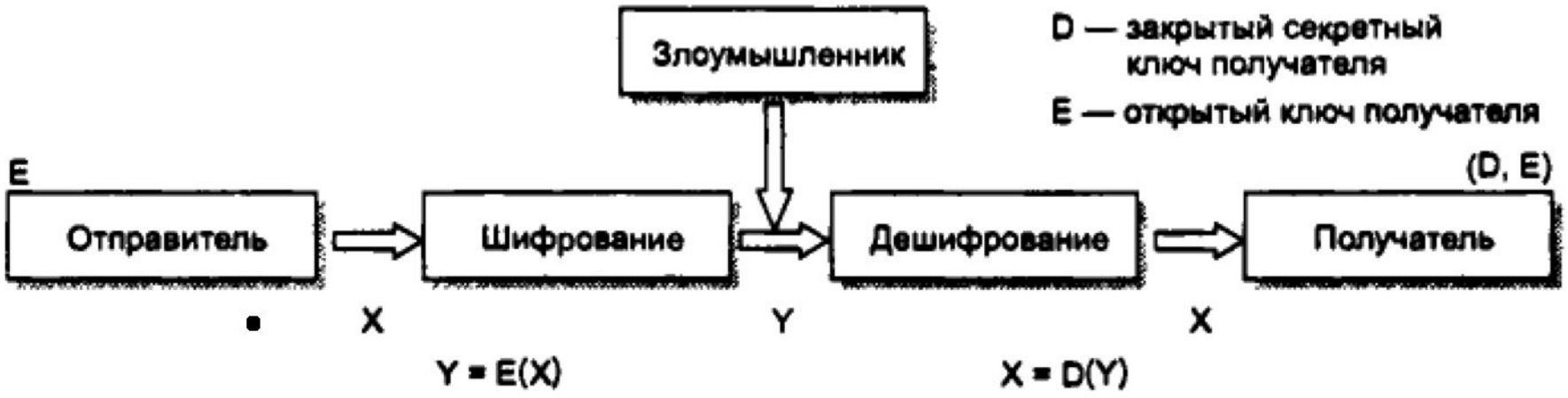
Для повышения криптостойкости иногда используют тройной алгоритм DES – троекратное шифрование с использованием 2 ключей. Производительность снижается.

AES (advanced Encryption Standard): 128 разрядные ключи (есть возможность использования 192- и 256-разрядных), за один цикл кодируется 128-разрядный блок.

В симметричных алгоритмах планирования главную проблему представляют ключи.
- Криптостойкость симметричных алгоритмов во многом зависит от качества ключа.
- Надежность канала передачи ключа второму участнику секретных переговоров. При наличии n абонентов, желающих обмениваться секретными данными по принципу «каждый с каждым», потребуется nx(n-1)/2 ключей, которые должны быть сгенерированы и распределены надежным способом. Количество ключей пропорционально квадрату количества абонентов, что при большом количестве абонентов делает задачу чрезвычайно сложной.

# Несимметричные алгоритмы шифрования

Винфилд Диффи и Мартин Хеллман в середине 70-х описали принципы шифрования с открытыми ключами.
Одновременно генерируется уникальная пара ключей, такая, что текст, зашифрованный одним ключом, может быть расшифрован только с использованием второго ключа, и наоборот.




Модель криптосхемы с открытым ключом

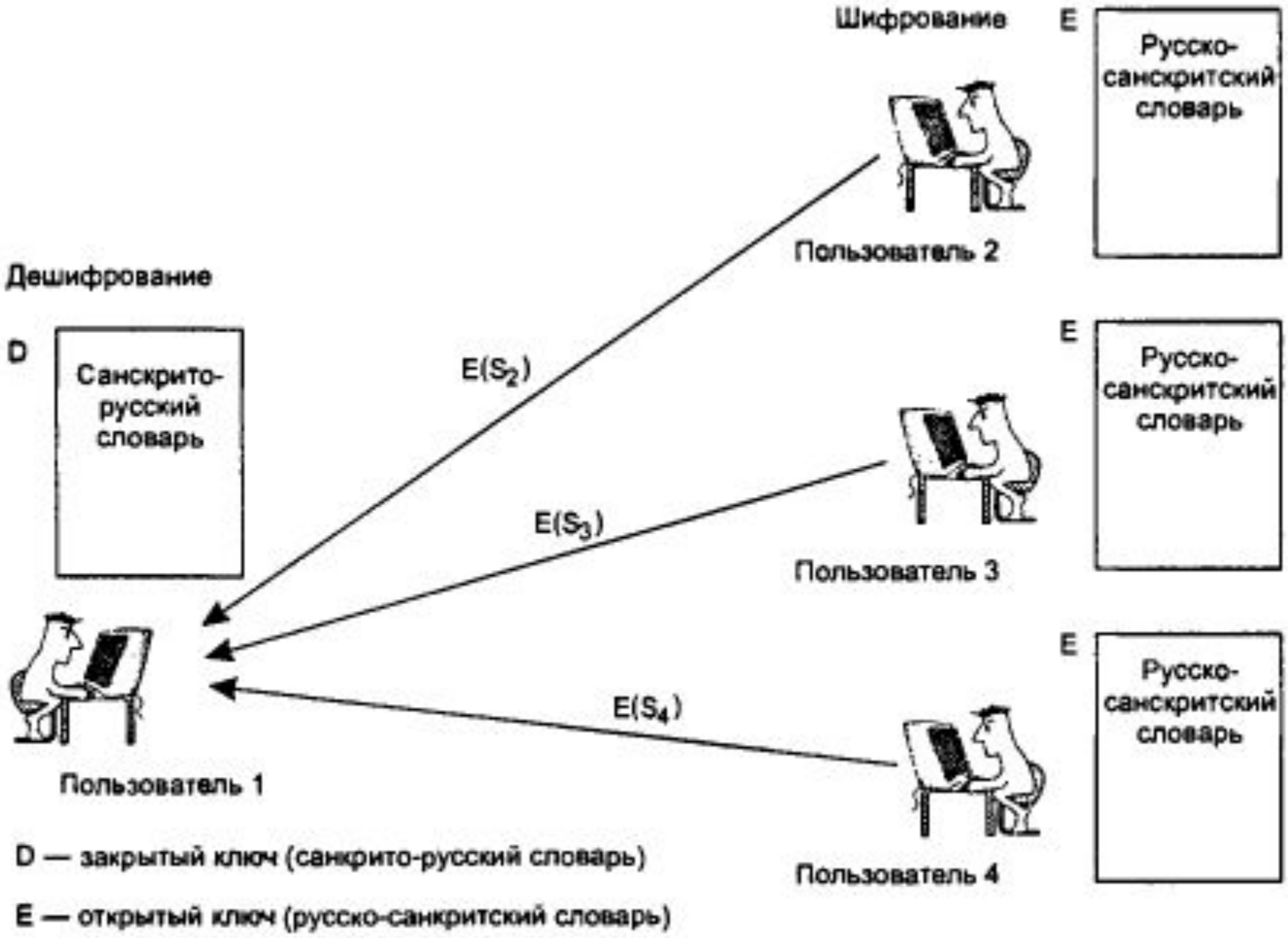
Задача отправителя заключается в том, чтобы по открытому каналу связи передать некоторое сообщение в защищенном виде.

Получатель на своей стороне генерирует два ключа: открытый Е и закрытый Д.

Закрытый ключ абонент должен хранить в защищенном месте, а открытый передает всем, с кем хочет поддерживать защищенные отношения.

Открытый ключ используется для шифрования текста, но расшифрован он может быть только с помощью закрытого ключа.

Открытый и закрытый ключи не могут быть независимы между собой, следовательно, существует теоретическая возможность вычисления закрытого ключа по открытому, но это связано с огромным объемом и временем вычислений.


Шифрование
E
Русско-санскритский словарь
Пользователь 2
Дешифрование
D
Санскрито-русский словарь
E(S2)
E
Русско-санскритский словарь
E(S3)
Пользователь 3
E
Русско-санскритский словарь
E(S4)
Пользователь 1
Пользователь 4
D — закрытый ключ (санкрито-русский словарь)
E — открытый ключ (русско-санкритский словарь)

а

# Аутентификация или электронная подпись

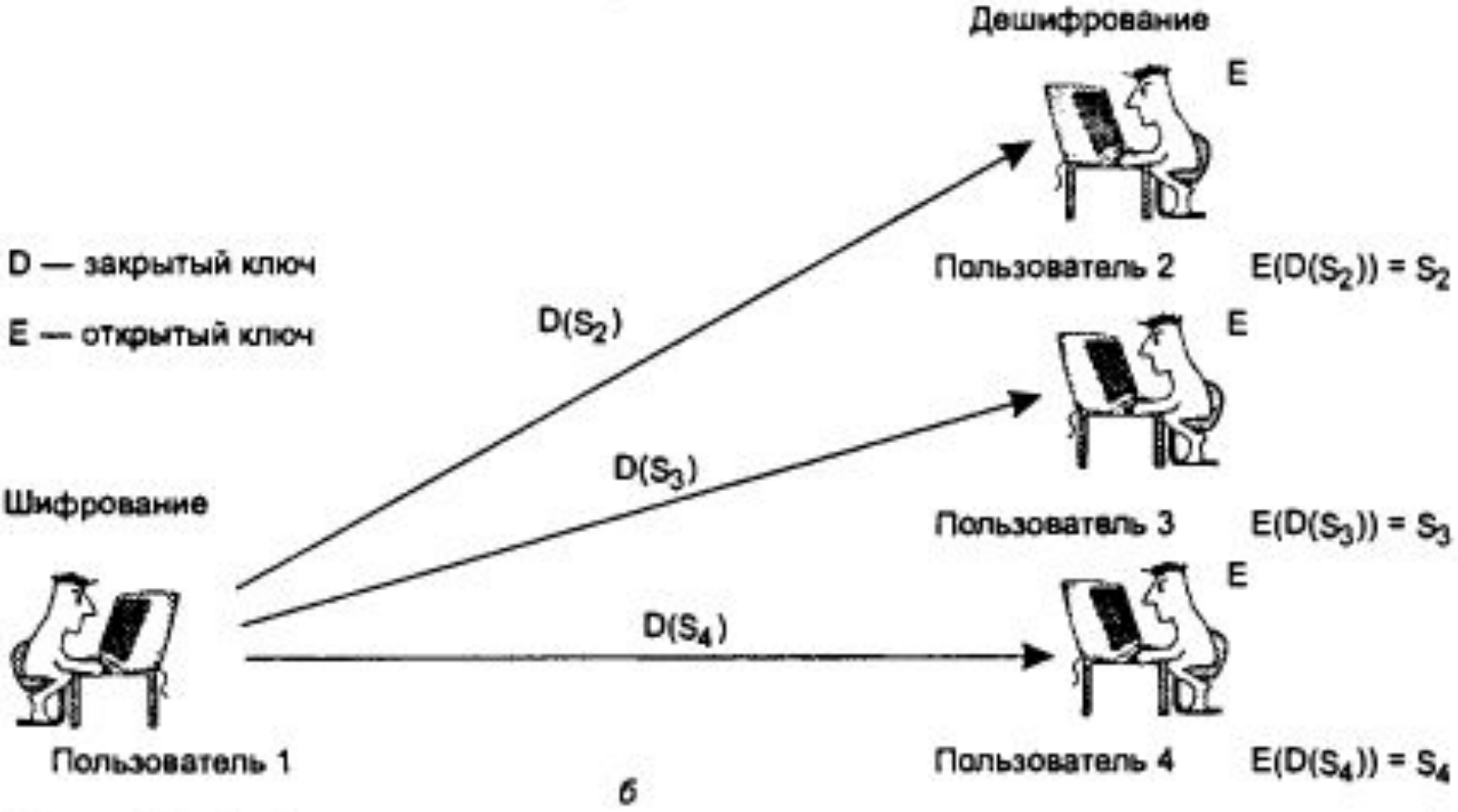



**Рис. 11.4.** Две схемы использования открытого и закрытого ключей

Если нужна взаимная аутентификация и двунаправленный секретный обмен сообщениями, то каждая из общающихся сторон генерирует свою пару ключей и посылает открытый ключ своему абоненту.

В сети из n абонентов всего будет 2n ключей: n открытых ключей для шифрования и n секретных ключей для дешифрирования.

Таким образом решается проблема масштабируемости – квадратичная зависимость количества ключей от числа абонентов в симметричных алгоритмах заменяется линейной зависимостью в несимметричных алгоритмах. Исчезает задача секретной доставки ключа. Злоумышленнику нет смысла стремиться захватить секретный ключ, поскольку это не дает возможности расшифровать сообщение или вычислить закрытый ключ.

Хотя информация об открытом ключе не является секретной, ее нужно защищать от подлогов, чтобы злоумышленник под видом легального пользователя не навязал свой открытый ключ, после чего он сможет дешифрировать сообщения своим закрытым ключом и рассылать свои сообщения от имени легального пользователя.

Решение этой проблемы – технология цифровых сертификатов.

Сертификат – это электронный документ, который связывает конкретного пользователя с конкретным ключом.

# Криптоалгоритм RSA

Наиболее популярный в настоящее время криптоалгоритм с открытым ключом. Разработан в 1978 году.
**RSA** (аббревиатура от фамилий Rivest, Shamir и Adleman).
RSA стал первым алгоритмом такого типа, пригодным и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений.
После работы над более чем 40 возможными вариантами, им удалось найти алгоритм, основанный на различии в том, насколько легко находить большие простые числа и насколько сложно раскладывать на множители произведение двух больших простых чисел, получивший впоследствии название RSA.

1. Случайно выбираются два очень больших простых числа p и q.
2. Вычисляются два произведения $n=p\times q$ и $m=(p-1)\times(q-1)$.
3. Выбирается случайное целое число E, не имеющее общих сомножителей с m.
4. Находится D, такое, что $DE=1$ по модулю m.
5. Исходный текст, X, разбивается на блоки таким образом, чтобы $0<X<n$.
6. Для шифрования сообщения необходимо вычислить $C=X^E$ по модулю n.
7. Для дешифрирования вычисляется $X=C^D$ по модулю n.

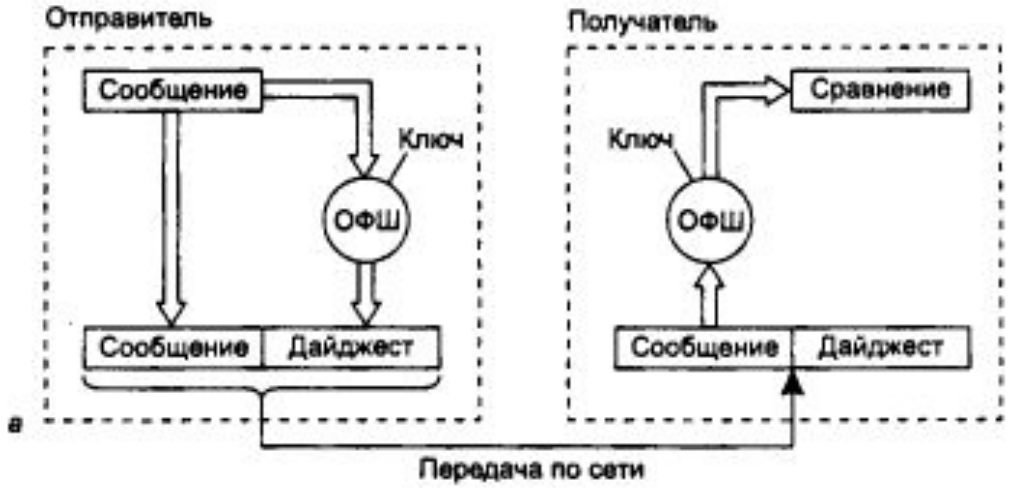
Таким образом, чтобы зашифровать сообщение, необходимо знать пару чисел (E, n), а чтобы дешифрировать – пару чисел (D, n). Первая пара – это открытый ключ, а вторая – закрытый.

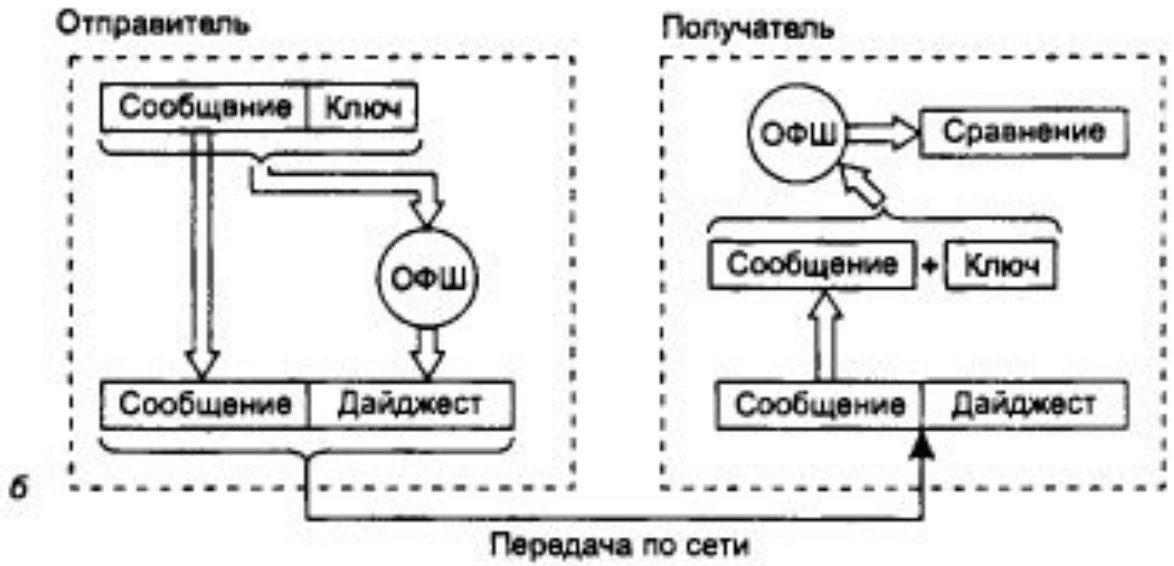
Зная открытый ключ (E, n), можно вычислить значение закрытого ключа D. Необходимым промежуточным действием в этом преобразовании является нахождение чисел p и q, для чего нужно разложить на простые множители очень большое число n, а на это требуется очень много времени. Именно с огромной вычислительной сложностью разложения большого числа на простые множители связана высокая криптостойкость алгоритма RSA. В некоторых публикациях приводятся следующие оценки: для того чтобы найти разложение 200-значного числа, понадобится 4 миллиарда лет работы компьютера с быстродействием миллион операций в секунду. Однако следует учесть, что в настоящее время активно ведутся работы по совершенствованию методов разложения больших чисел, поэтому в алгоритме RSA стараются применять числа длиной более 200 десятичных разрядов.

Программная реализация криптоалгоритмов типа RSA значительно сложнее и менее производительна, чем реализация классических криптоалгоритмов типа DES. Вследствие сложности реализации операций модульной арифметики криптоалгоритм RSA часто используют только для шифрования небольших объемов информации, например для рассылки классических секретных ключей или в

# Односторонние функции шифрования

Шифрование с помощью односторонней функции (хэш-функции, дайджест-функции).
Эта функция, примененная к шифруемым данным, дает в результате значение (дайджест), состоящее из фиксированного небольшого числа байтов. Дайджест передается с исходным сообщением. Получатель сообщения, зная какая ОФШ, была применена для получения дайджеста, заново вычисляет его, использую незащищенную часть сообщения. Если полученный и вычисленный дайджесты совпадают, значит, полученное сообщение не подвергалось изменениям.






Построение односторонних функций является трудной задачей. Такого рода функции должны удовлетворять двум условиям:

- по дайджесту, вычисленному с помощью данной функции, невозможно каким-либо образом вычислить исходное сообщение;
- должна отсутствовать возможность вычисления двух разных сообщений, для которых с помощью данной функции могли быть вычислены одинаковые дайджесты.

# Криптографический протокол SSL/TLS

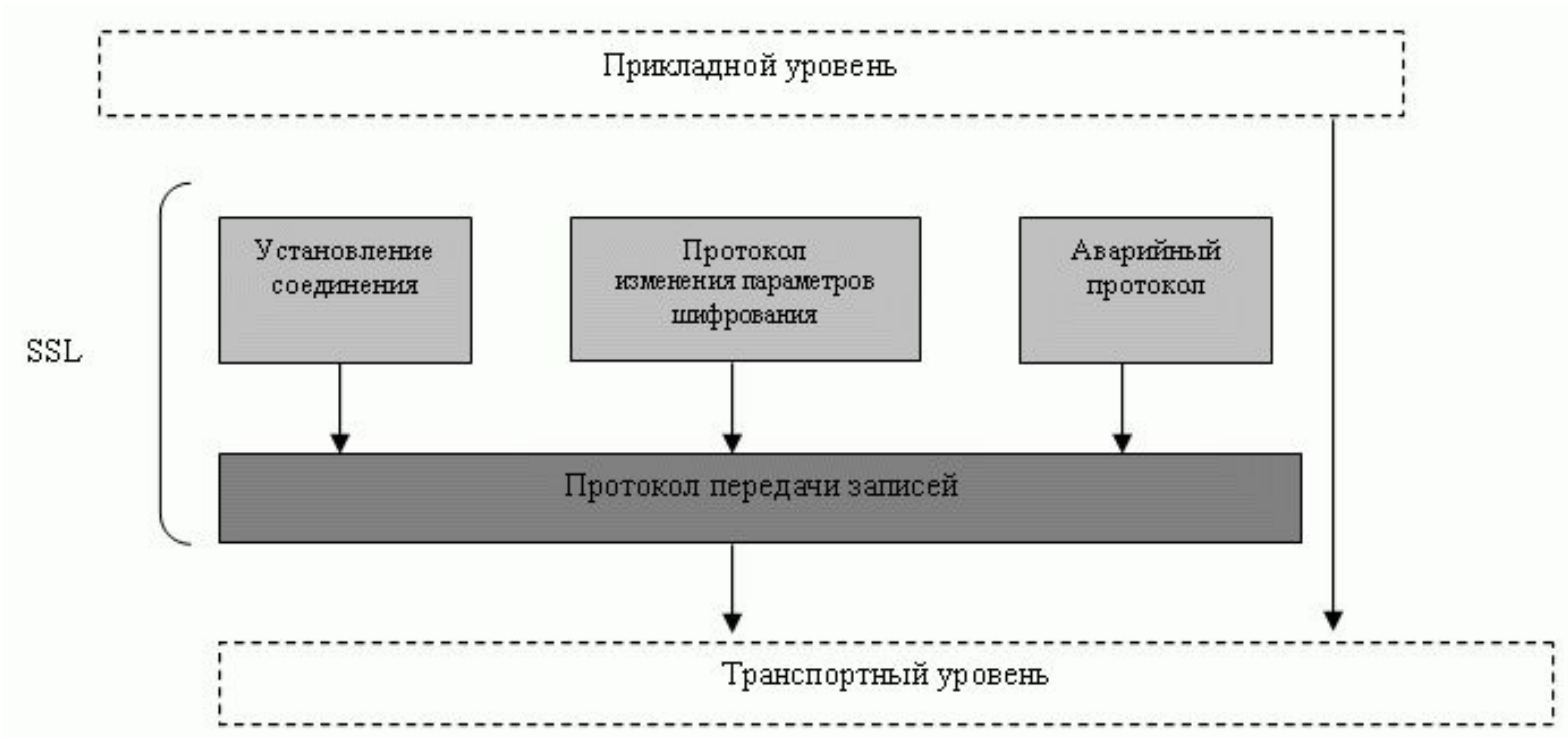
- ✔ Основная функцфия протокола SSL/TLS состоит в обеспечении конфиденциальности и целостности данных прикладного уровня, передаваемых между двумя взаимодействующими приложениями, одно из которых является клиентом, а другое - сервером.

- ✔ Протокол TLS (Transport Layer Security) разрабатывался на основе спецификации протокола SSL 3.0 (Secure Socket Layer), опубликованного корпорацией Netscape. Различия между данным протоколом и SSL 3.0 несущественны, но важно заметить, что TLS 1.0 и SSL 3.0 несовместимы, хотя в TLS 1.0 предусмотрен механизм, который позволяет TLS иметь обратную совместимость с SSL 3.0.

- ✔ Для обеспечения криптозащиты используется связка из закрытых ключей, сертификатов и открытых ключей шифрования.

# Подпротоколы SSL в модели TCP/IP

SSL - это многоуровневый протокол, который состоит из четырех подпротоколов на двух уровнях.



# Протокол записи

Нижним уровнем является протокол Записи (Record Layer), который переносит на транспортный уровень сообщения от других подпротоколов SSL, а также данные, поступающие от прикладного уровня. Сообщения из протокола Записи - это полезная нагрузка для транспортного уровня, обычно TCP.

Протокол Записей обеспечивает безопасность соединения, которая основана на следующих двух свойствах:

Конфиденциальность соединения

Для защиты данных используется один из алгоритмов симметричного шифрования. Ключ для этого алгоритма создается для каждой сессии и основан на секрете, о котором договариваются в протоколе Рукопожатия. Протокол Записи также может использоваться без шифрования.

Целостность соединения

Обеспечивается проверка целостности сообщения с помощью MAC с ключом. Для вычисления MAC используются хэш-функции SHA-1 и MD5. Протокол Записи может выполняться без вычисления MAC.

# Протокол Установления соединения

Протокол Установления соединения (протокол Рукопожатия – Handshake Protocol) используется для согласования данных сессии между клиентом и сервером. Он устанавливает набор шифров и задает ключи и параметры безопасности.

Протокол Рукопожатия обеспечивает безопасность соединения, которая основана на следующих свойствах:

- Участники **аутентифицированы** с использованием криптографии с открытым ключом. Эта аутентификация может быть необязательной, но обычно требуется по крайней мере для сервера.
- Переговоры о разделяемом секрете **безопасны,** т.е. этот общий секрет невозможно подсмотреть.
- Переговоры о разделяемом секрете **надежны**, если выполнена аутентификация хотя бы одной из сторон. В таком случае атакующий, расположенный в середине соединения, не может модифицировать передаваемый секрет незаметно для участников соединения.

## Протокол изменения параметров шифрования (Change Cipher Spec Protocol) используется для изменения данных ключа - информации, необходимой для создания ключей шифрования.

## Аварийный протокол

(Alert Protocol) нужен, чтобы известить о ситуациях, отклоняющихся от нормы. Обычно предупреждение отсылается тогда, когда подключение закрыто и получено неправильное сообщение, сообщение невозможно расшифровать или пользователь отменяет операцию.

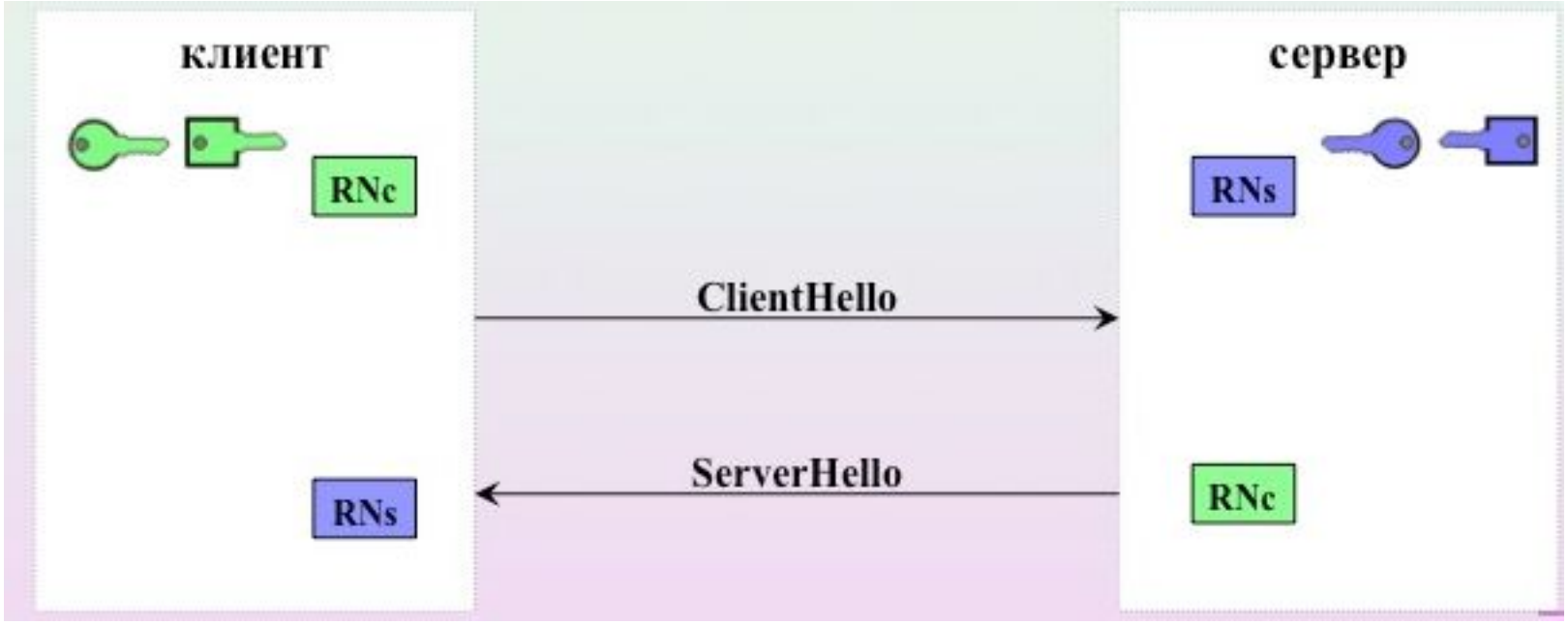
# Протокол установления соединения (рукопожатия)

При обмене сообщениями во время протокола рукопожатия достигаются следующие цели:

- Устанавливается, какой вариант протокола из поддерживаемых ( SSLv3, TLSv1, TLSv1.1, TLSv1.2)— будет использоваться. Всегда выбирается самый последний из возможных вариантов, то есть у TLSv1 всегда будет приоритет перед SSLv3 в случае, если и клиент и сервер поддерживают оба варианта. Клиент предлагает список вариантов, а сервер выбирает из предложенного списка.
- Отправляются аутентификационные данные. Обычно сервер посылает аутентификационную информацию в форме сертификата X.509 (SSL), но протокол поддерживает и другие методы.
- Устанавливается идентификатор (ID) сессии, таким образом, при необходимости сессия может быть перезапущена.
- Происходят переговоры о наборе шифров, состоящем из алгоритма обмена ключами вместе с типом алгоритма шифрования объёмных данных и типом MAC, которые будут использоваться в последующей сессии обмена данными (протокол записи). Обычно в качестве алгоритма обмена ключами используется асимметричный алгоритм, такой как RSA, DSA или ECC (Elliptic Curve Cipher, шифр на основе эллиптических кривых, смотрите RFC 5289). Асимметричные алгоритмы потребляют очень много ресурсов процессора и потому для последующего шифрования объемных данных (протокол записи) используются симметричные шифры. Алгоритмы обмена ключами применяются для передачи информации, на основании которой могут быть независимо вычислены сеансовые ключи для

# Установление соединения: первая фаза

1. Согласование алгоритмов
   - клиент сообщает версию протокола, случайное число и список поддерживаемых алгоритмов – ClientHello
   - сервер сообщает выбранную версию протокола, своё случайное число и выбранные алгоритмы – ServerHello



**ClientHello (1):** Сообщение ClientHello предлагает список поддерживаемых версий/вариантов протоколов, поддерживаемые наборы шифров в порядке предпочтения и список алгоритмов сжатия (обычно NULL). Клиент также посылает случайное значение размером в 32 байта (отметка текущего времени), которое позднее будет использоваться для вычисления симметричного ключа, и идентификатор сессии, который будет равен нулю, если не было предыдущих сессий, либо ненулевому значению, если клиент считает, что предыдущая сессия существует.

Каждый набор шифров состоит из алгоритма обмена ключами, алгоритма шифрования объёмных данных и алгоритма MAC (хэширования).

Набор шифров, используемый и клиентом и сервером, при первоначальном соединении таков: TLS_NULL_WITH_NULL_NULL (0x00, 0x00)

# первый NULL — алгоритм обмена ключами

# следующий за ним WITH_NULL определяет алгоритм шифрования объёмных данных

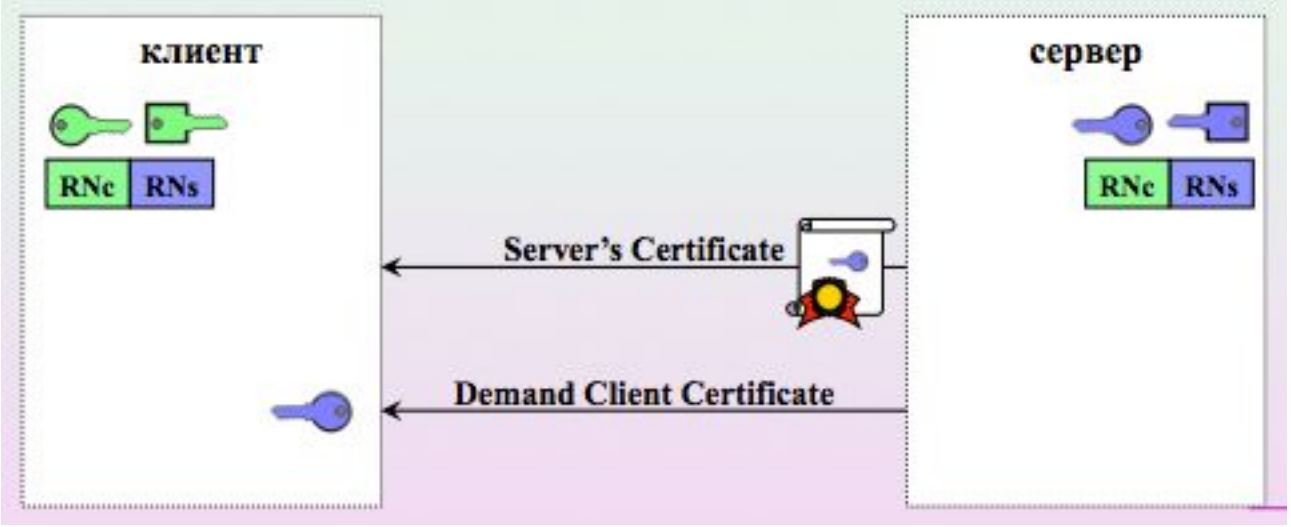
# последний NULL определяет MAC

Это значение говорит о том, что шифрование выполняться не будет и потому все сообщения до Client Key Exchange (ClientKeyMessage) будут отправляться открытым текстом.

**ServerHello (2):** Сообщение ServerHello возвращает выбранные вариант/номер версии протокола, набор шифров и алгоритм сжатия. Сервер посылает случайное значение размером в 32 байта (отметка текущего времени), которое позднее будет использоваться для вычисления симметричных ключей. Если идентификатор сессии в сообщении ClientHello был равен нулю, сервер создаст и вернёт идентификатор сессии. Если в сообщении ClientHello был предложен идентификатор предыдущей сессии, известный данному серверу, то протокол рукопожатия будет проведён по упрощённой схеме. Если клиент предложил неизвестный серверу идентификатор сессии, сервер возвращает новый идентификатор сессии и протокол рукопожатия проводится по полной схеме. В RFC 7250 определено расширение **server_certificate_format**, которое может применяться для указания формата используемого сертификата. Это может быть нормальный формат X.509 или формат **RawPublicKey**, при котором в последующих сообщениях передачи сертификата протокола рукопожатия сертификат сокращается до одного атрибута subjectPublicKeyInfo.

# Установление соединения: вторая фаза

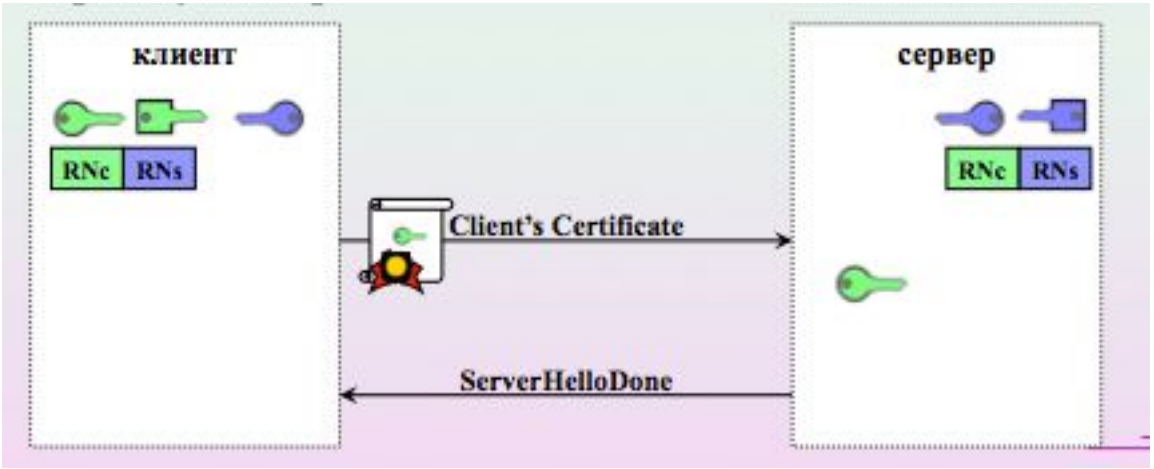
2. Аутентификация сервера
   - сервер высылает свой сертификат (X.509 или OpenPGP)
   - сервер может запросить сертификат клиента, чтобы аутентифицировать его
   - клиент проверяет сертификат сервера, используя PKI



**Certificate (3):** Сервер посылает свой сертификат X.509, содержащий открытый ключ сервера, алгоритм которого должен совпадать с алгоритмом обмена ключами в выбранном наборе шифров. Протокол предлагает и другие методы доставки открытого ключа, — можно, например, просто указать на запись DNS KEY RR, — но сертификат X.509 является стандартным общепринятым методом. Цель данного сообщения — получение клиентом из доверенного источника открытого ключа сервера, который затем может использоваться этим клиентом для отправки зашифрованного сообщения. RFC 7250 определяет упрощённый формат сертификата, в котором открытый ключ в чистом виде инкапсулируется в обёртку, состоящую из атрибута SubjectPublicKeyInfo (необходимого для описания свойств открытого ключа). Это скромный шаг в сторону более разумного способа обретения открытого ключа непосредственно от аутентифицированного источника, такого как защищённая DNS-запись DNSSEC.

# Установление соединения: вторая фаза

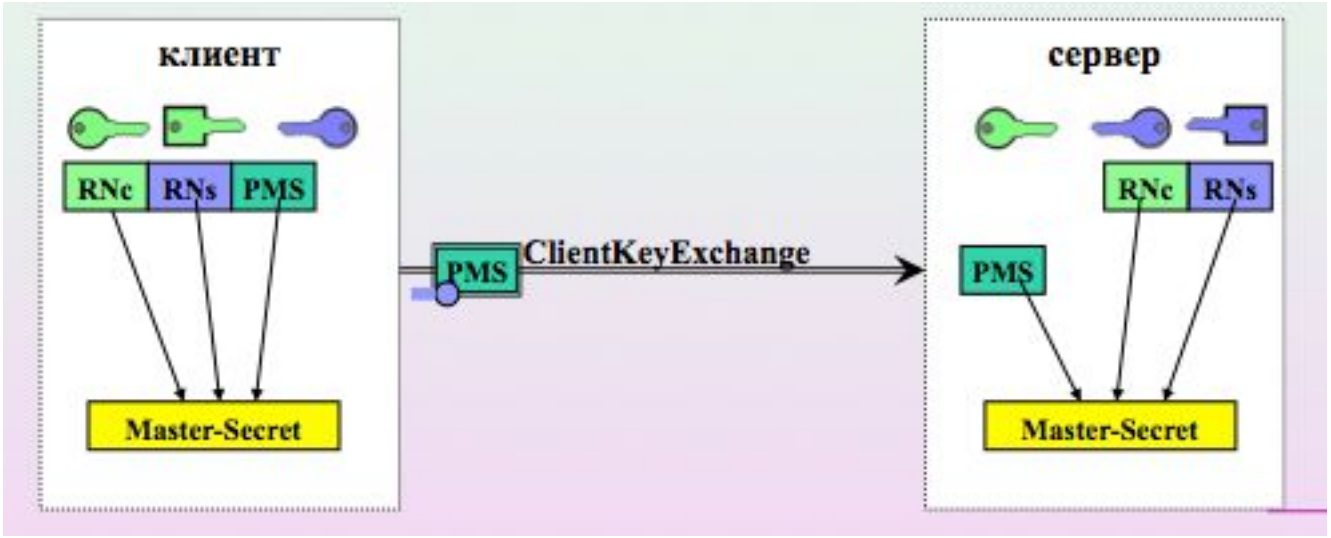
2. Аутентификация клиента (опционально)
   - клиент может предоставить свой сертификат, тогда сервер, используя PKI, проверяет аутентичность клиента – так обеспечивается взаимная аутентификация
   - фаза аутентификации заканчивается сообщ. ServerHelloDone



**ServerDone (4):** Это сообщение указывает на окончание серверной части данной последовательности диалога и побуждает клиента продолжить последовательность протокола. **Примечание:** В этом месте сервер может запросить клиентский сертификат для выполнения взаимной аутентификации.
**Примечание:** Если во время переговоров об установлении TLS/SSL сервер запросил сертификат клиента, то клиент должен отправить свой сертификат (в том же формате, который определен для сервера, с тем исключением, что RFC 6066 позволяет любому клиенту посылать URL сертификат вместо полного сертификата) непосредственно за сообщением **ServerDone** и до сообщения **ClientKeyExchange**.

# Установление соединения: третья фаза

3. Генерация ключа сессии
   - клиент генерирует Pre-Master-Secret и пересылает его серверу в сообщении ClientKeyExchange
   - клиент и сервер на основе RNc, RNs и PMS генерируют ключ для симметричного криптоалгоритма

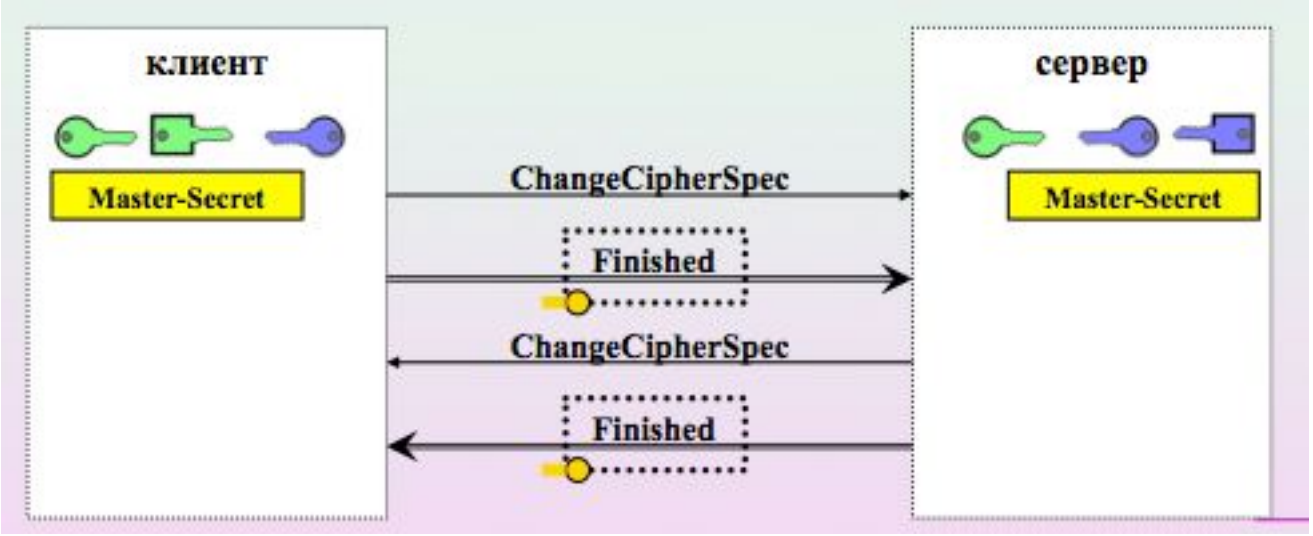


**ClientKeyExchange (5):** Клиент вычисляет так называемый ключ pre-master key, используя случайные числа (или отметки текущего времени) сервера и клиента. Он шифрует этот ключ с помощью открытого ключа сервера, полученного из предоставленного сертификата X.509. Только сервер может расшифровать данное сообщение своим закрытым ключом. Обе стороны независимо друг от друга вычисляют общий секретный ключ master key из ключа pre-master, используя определённый в стандарте алгоритм. Любые сеансовые ключи, которые могут потребоваться, будут производными от этого ключа master key.

# Установление соединения: четвертая фаза

4. Завершение квитирования

- клиент посылает сообщение о переходе в режим шифрования ChangeCipherSpec и посылает зашифрованное сообщение о завершении квитирования с хэшем всех сообщений
- сервер посылает ChangeCipherSpec и зашифрованное сообщение о завершении квитирования с хэшем всех сообщений



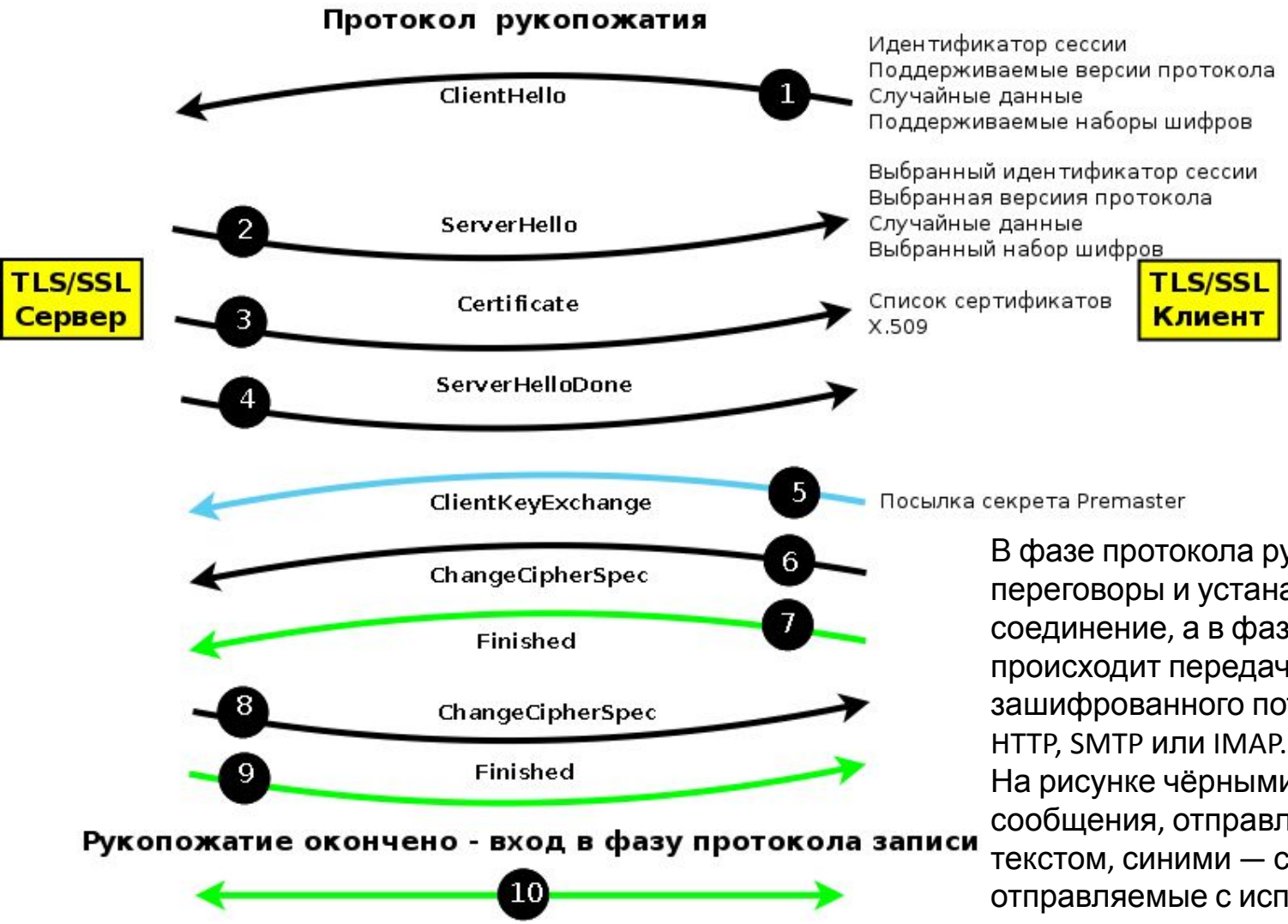
**ChangeCipherSpec — клиент (6):** Это сообщение указывает, что весь последующий трафик, исходящий от данного клиента, будет зашифрован с помощью выбранного (в результате переговоров) алгоритма шифрования объёмных данных и будет содержать MAC, сформированный по выбранному алгоритму. Номинально это сообщение всегда будет шифроваться текущим шифром, который в стадии установления соединения будет NULL, и, следовательно, данное сообщение отправляется в открытом виде. Оно часто объединяется с сообщением Client Key Exchange.

**Finished — клиент (7):** Это сообщение содержит все сообщения, отправленные и полученные во время протокола рукопожатия, за исключением сообщения Finished. Оно шифруется с помощью алгоритма шифрования объемных данных и хэшируется с помощью алгоритма MAC, о которых договорились стороны. Если сервер может расшифровать и верифицировать это сообщение (содержащее все предыдущие сообщения), используя независимо вычисленный им сеансовый ключ, значит диалог был успешным. Если же нет, на этом месте сервер прерывает сессию и отправляет сообщение Alert с некоторой (возможно, неконкретной) информацией об ошибке.

**ChangeCipherSpec — сервер (8):** Это сообщение указывает, что весь последующий трафик, исходящий от данного сервера, будет зашифрован с помощью выбранного (в результате переговоров) алгоритма шифрования объёмных данных и будет содержать MAC, сформированный по выбранному алгоритму. Номинально это сообщение всегда будет шифроваться текущим шифром, который в стадии установления соединения будет NULL, и, следовательно, данное сообщение отправляется в открытом виде. Получение данного сообщения неявно говорит клиенту о том, что сервер получил и смог обработать сообщение Finished этого клиента.

**Finished — сервер (9):** Это сообщение содержит все сообщения, отправленные и полученные во время протокола рукопожатия, за исключением сообщения Finished. Оно шифруется с помощью алгоритма шифрования объемных данных и включает MAC, о которых договорились стороны. Если клиент может расшифровать это сообщение, используя независимо вычисленный им сеансовый ключ, значит диалог был успешным. Если же нет, клиент прерывает соединение и выдаёт сообщение Alert и подходящий (хотя и не всегда конкретный) код ошибки.

**Record Protocol (протокол записи):** Последующие сообщения между сервером и клиентом шифруются с помощью алгоритма шифрования объемных данных и включают MAC, о которых договорились стороны.




В фазе протокола рукопожатия проводятся переговоры и устанавливается соединение, а в фазе протокола записи происходит передача (инкапсуляция) зашифрованного потока данных, таких как HTTP, SMTP или IMAP.

На рисунке чёрными стрелками показаны сообщения, отправляемые открытым текстом, синими — сообщения, отправляемые с использованием открытого ключа, предоставленного сервером (с помощью шифра обмена ключами), зелёными — сообщения, отправляемые с использованием того алгоритма шифрования объёмных данных и того MAC, о которых стороны договорились в процессе переговоров.

# Public Key Infrastructure

- Задачи: определение политики выпуска цифровых сертификатов, выдача их и аннулирование, хранение информации, необходимой для последующей проверки правильности сертификатов.

- Центр сертификации (Certification Authority) является основной структурой, формирующей цифровые сертификаты подчиненных центров сертификации и конечных пользователей. Центр сертификации сам генерирует собственный секретный ключ и сертификат, содержащий открытый ключ данного центра.

❖ Удостоверяет аутентичность открытого ключа пользователя своей электронно-цифровой подписью.

❖ Формирует список отозванных сертификатов.

❖ Ведет базы всех изготовленных сертификатов и списков отозванных сертификатов.

# Public Key Infrastructure

- Прежде чем веб-сервер начнет взаимодействовать с клиентом, на веб-сервере долженбыть установлен сертификат подписанный центром сертификатов (CA). Браузер клиента должен знать этот центр сертификатов (доверять ему). Это значит, что сертификат этого центра сертификатов уже установлен в браузере и срок действия сертификата не истек. Если сертификат не установлен, то при доступе клиента к веб-серверу, у него высветитсясообщение предлагающее принять или отклонить сертификат веб-сервера, так как он подписан неизвестным центром сертификатов.

# Public Key Infrastructure

- Сертификат содержит следующую информацию:
- Версия
- Серийный номер
- Идентификатор алгоритма поиска
- Имя издателя
- Период действия (не ранее/не позднее)
- Субъект сертификата
- Информация об открытом ключе субъекта
- Уникальный идентификатор издателя
- Уникальный идентификатор субъекта
- Дополнения
- Подпись

# Пользователь Боб и веб-сервер Элис собираются взаимодействовать через защищенный канал. Пользователь использует браузер для доступа к серверу, который поддерживает SSL

Для получения сертификата Элис необходимо:

1. Подготовить запрос на получение сертификата для отправки его CA:
   - o Сгенерировать пару открытый/закрытый ключ (с помощью утилиты генерации ключей, которая обычно есть в веб-сервере).
   - o Сгенерировать запрос на получение сертификата, который включает в себя идентифицирующую информацию о компании Элис(Distinguished Name) и открытый ключ.
   - o Отправить запрос на получение сертификата выбранному CA(например, запрос может быть отправлен почтой).
2. В ответ на запрос CA отправляет Элис цифровой сертификат, в котором содержится открытый ключ Элис, подписанный CA.
3. После получения ответа от CA, Элис помещает цифровой сертификат в базу сертификатов веб-сервера. После того как Элис установила цифровой сертификат подписанный CA, она может начинать работу с клиентами.

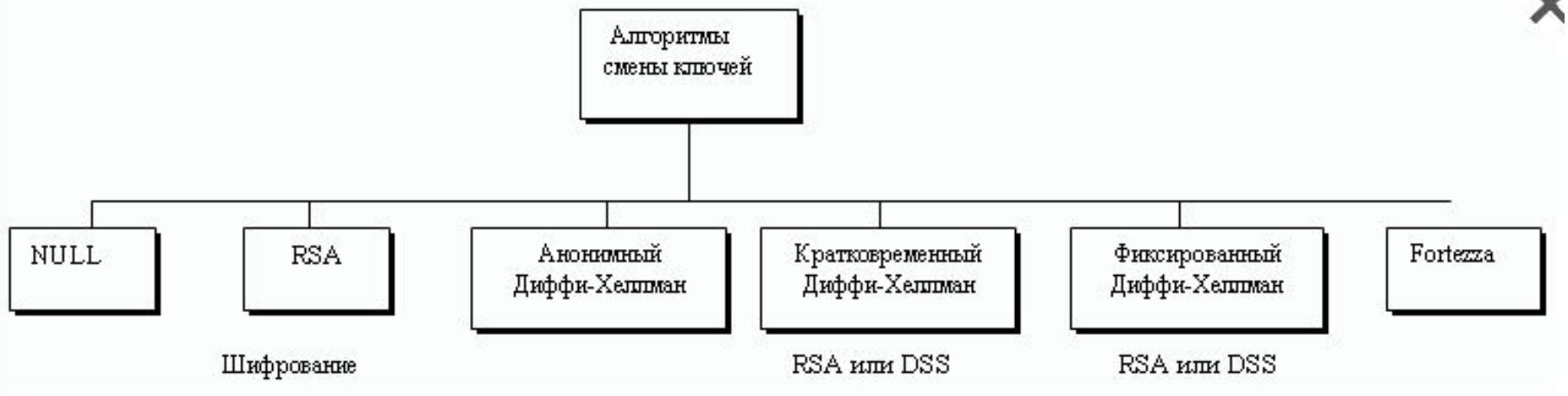
Когда Боб обращается к веб-серверу Элис, выполняются такие действия:

4. Боб обращается с веб-серверу Элис. При этом броузер Боба автоматически скачает цифровой сертификат с веб-сервера Элис.
5. Браузер Боба определит доверяет ли он CA, который подписал цифровой сертификат Элис:
- Сертификат CA дожен быть уже установлен в браузере Боба.
- Если сертификат установлен и срок его действия не истек, то дальнейший процесс будет прозрачным для Боба.
- Браузер дешифрует цифровой сертификат Элис с помощью открытого ключа CA, который получен из цифрового сертификата CA.
- Теперь у браузера Боба есть открытый ключ Элис.

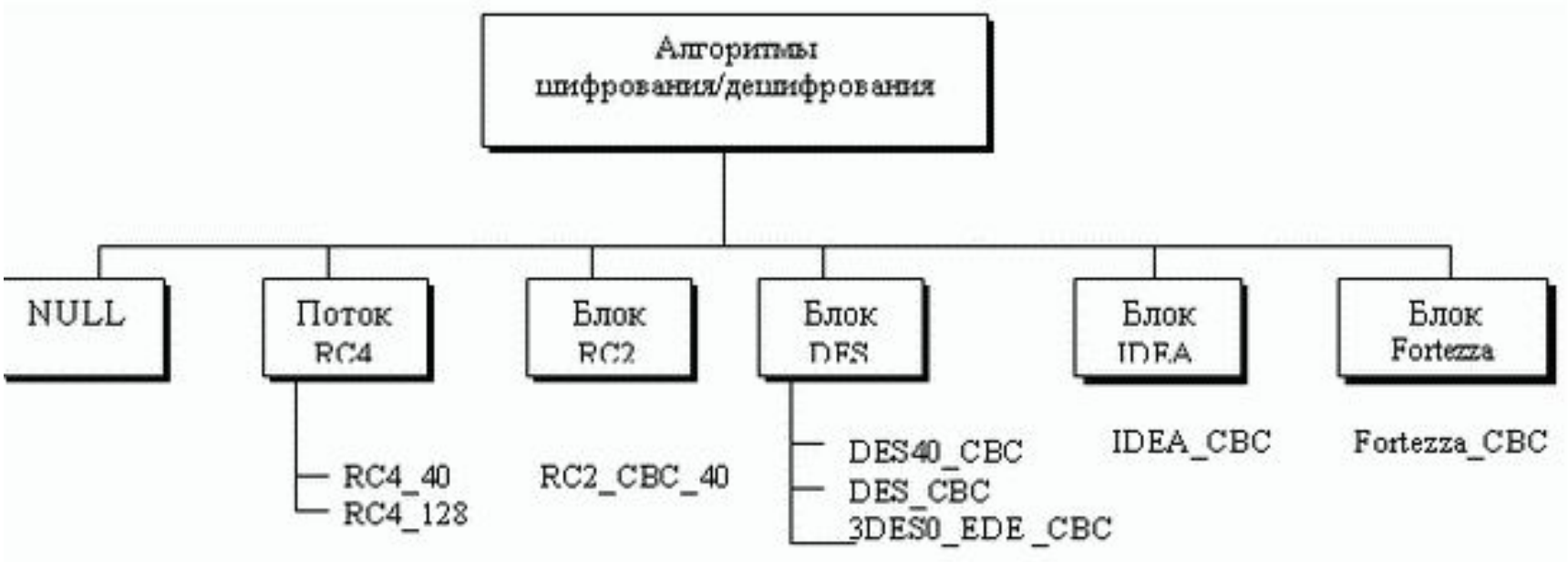
6. Боб отправляет Элис сообщение в котором содержится сессионный ключ и шифрует его с помощью открытого ключа Элис:
- Сессионный ключ используется для шифрования трафика между Бобом и Элис.
- Этот ключ симметричный и будет использоваться для шифрования и дешифрования данных передаваемых между Бобом и Элис.
- Так как сообщение можно расшифровать только закрытым ключем Элис, то сессионный ключ безопасно доставляется на веб-сервер Элис.

7. Как только веб-сервер получает сообщение в котором хранится сессионный ключ, он его дешифрует с помощью закрытого ключа Элис.
- Все дальнейшие сообщения шифруются с помощью сессионного ключа (может передаваться информация об адресе, телефоне, кредитной карточке Боба).
- В зависимости от длительности соединения с веб-сервером,может использоваться несколько сессионных ключей, которые будут меняться

# Алгоритмы смены ключей



# Алгоритмы шифрования/дешифрования

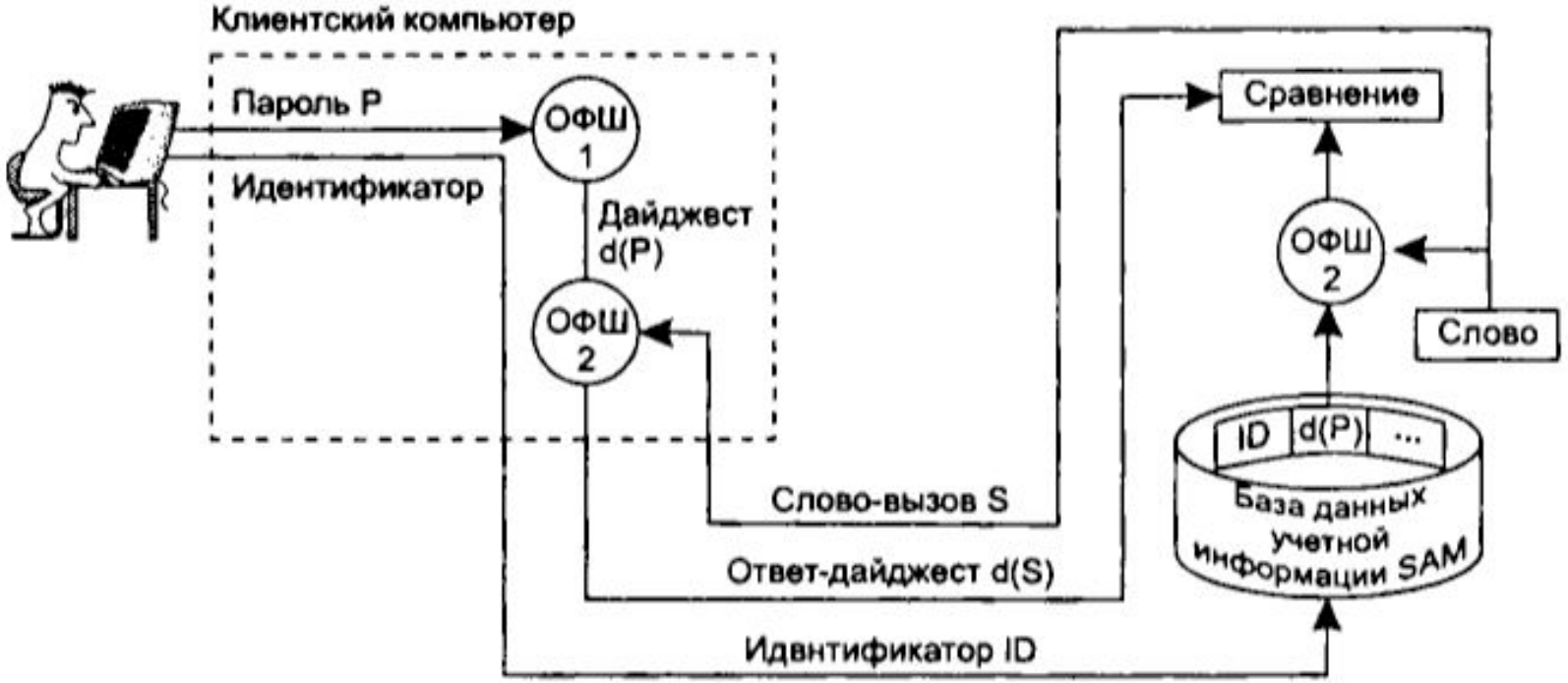


# Алгоритмы хэширования

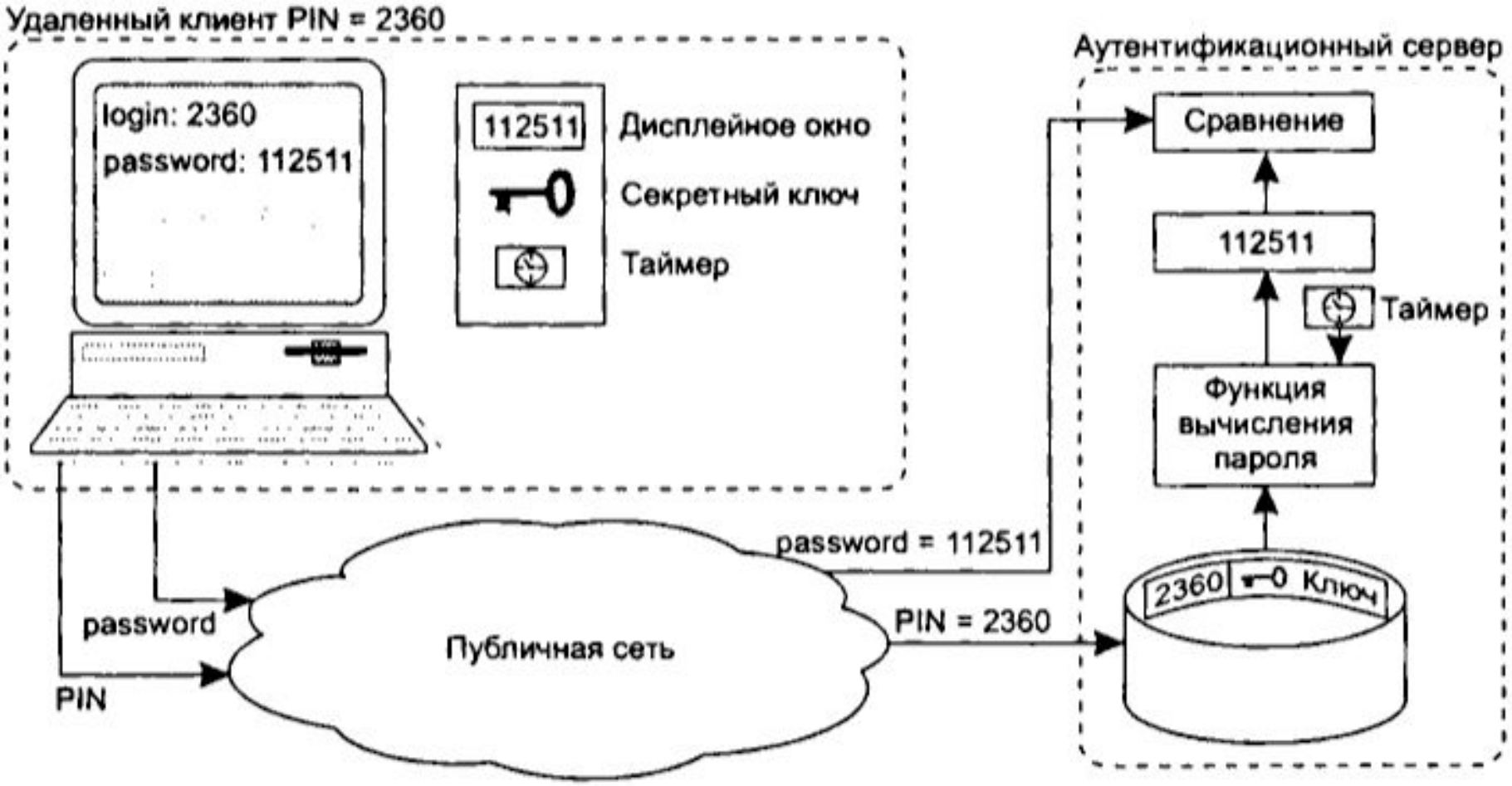
# Список набора шифров SSL

| *Набор шифров* | Смена ключей | Шифрование | Хэш |
|---|---|---|---|
| SSL-NULL-WITH-NULL- NULL | NULL | NULL | NULL |
| SSL_RSA_WITH_NULL_MD5 | RSA | NULL | MD5 |
| SSL_RSA_WITH_NULL_SHA | RSA | NULL | SHA-1 |
| SSL_RSA_WITH_RC4_128_MD5 | RSA | RC4 | MD5 |
| SSL_RSA_WITH_RC4_128_SHA | RSA | RC4 | SHA-1 SHA-1 |
| SSL_RSA_WITH_IDEA_CBC_SHA | RSA | IDEA | SHA-1 |
| SSL_RSA_WITH_DES_CBC_SHA | RSA | DES | SHA-1 |
| SSL RSA WITH 3DES EDE CBC SHA | RSA | 3DES | |
| SSL_ DH_anon_ WITH_ RC4_128_ MD5 | DH anon | RC4 | MD5 |
| SSL_ DH_ anon WITH_ DES_ CBC_ SHA | DH anon | DES | SHA-1 |
| SSL _DH_ anon_ WITH_ 3DES_ EDE_ CBC SHA | DH anon | 3DES | SHA-1 |
| SSL_ DHE_ RSA_ WITH_ DES_ CBC_ SHA | DHE RSA | DES | SHA-1 |
| SSL_ DHE_RSA_ WITH_3DES_ EDE CBC SHA | DHE RSA | 3DES | SHA-1 |
| SSL_DHE_DSS_ WITH_ DES_ CBC_ SHA | DHE DSS | DES | SHA-1 |
| SSL_ DHE_ DSS_ WITH_ 3DES_ EDE_ CBC_ SHA | DHE DSS | 3DES | SHA-1 |

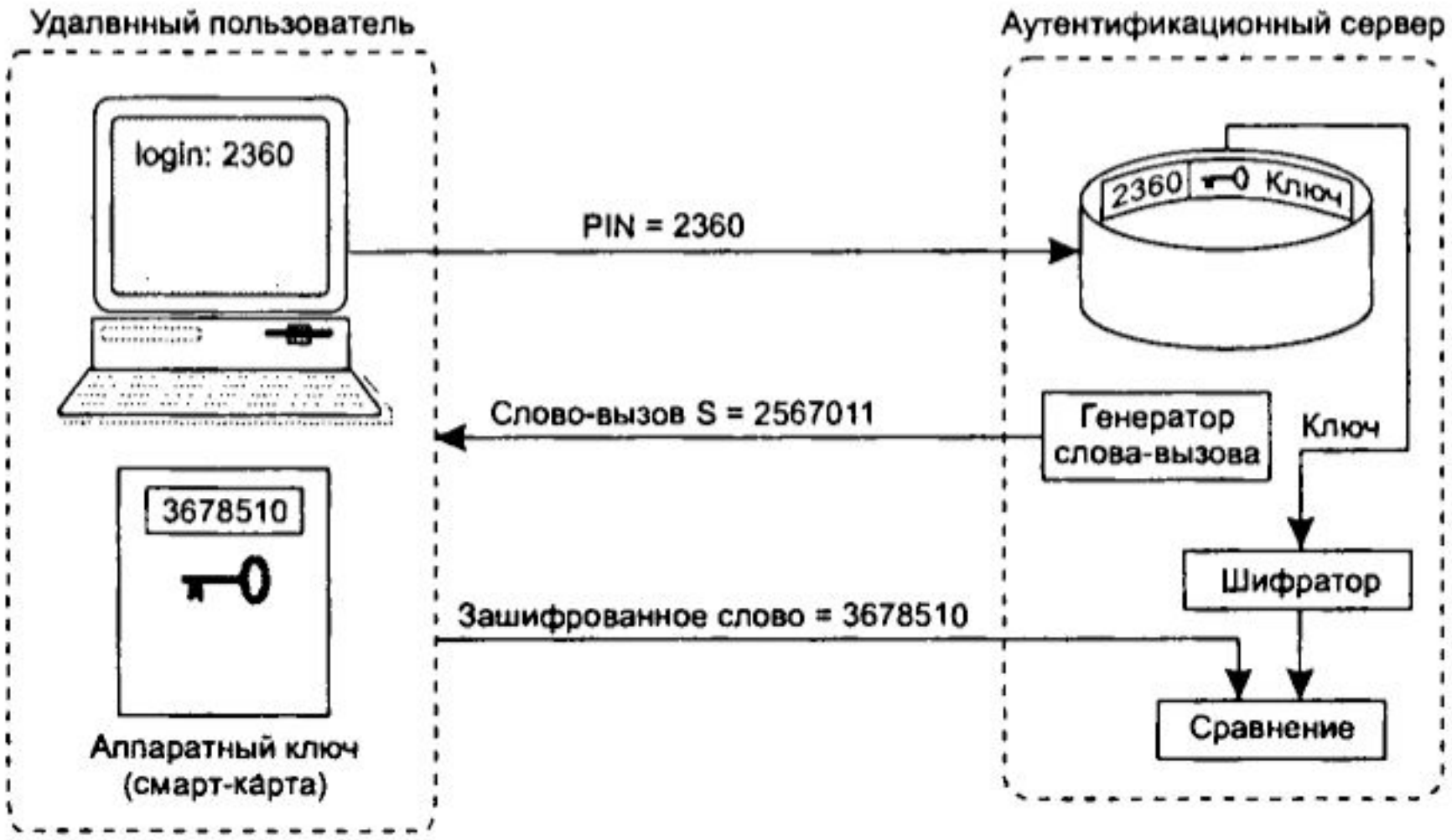
# Схема сетевой аутентификации на основе многоразового пароля



# Аутентификация, основанная на временной синхронизации

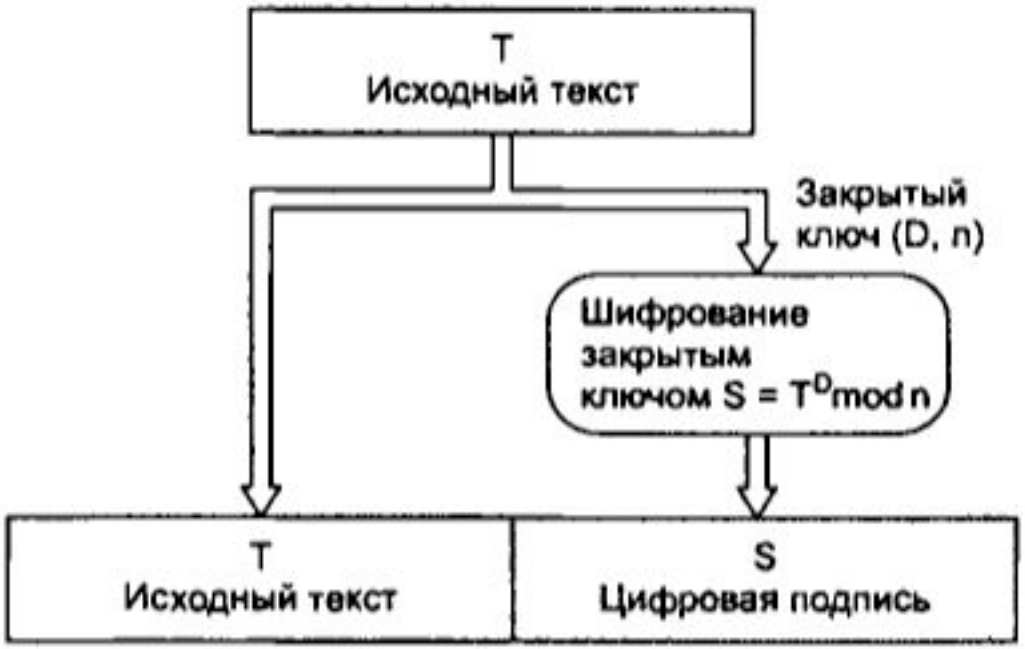


# Аутентификация по схеме “запрос- ответ”



# Аутентификация пользователя на основе сертификатов

# Схема формирования цифровой подписи по алгоритму RSA



# Обеспечение конфиденциальности документа с

# Схема получения аутентикода