## 1. Основные задачи защиты информации. Определения и примеры ?

Защита информации представляет собой принятие правовых, организационных и технических мер, направленных на:

1) обеспечение защиты информации от неправомерного доступа, уничтожения, модифицирования, блокирования, копирования, предоставления, распространения, а также от иных неправомерных действий в отношении такой информации;

2) соблюдение конфиденциальности информации ограниченного доступа;

3) реализацию права на доступ к информации.

Причем конфиденциальность информации – это такое состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на него право.

Основные задачи защиты информации в организациях (предприятиях, учреждениях):

1) Предотвращение разглашения сведений ограниченного доступа (обеспечение режима конфиденциальности (секретности) информации).

2) Предотвращение перехвата информации в процессе ее обработки техническими средствами (защита объектов информатизации от утечки информации по техническим каналам).

3) Предотвращение перехвата акустической (речевой) информации из выделенных (защищаемых) помещений (защита выделенных помещений от утечки речевой информации по техническим каналам).

4) Предотвращение негласного визуального наблюдения и съемки (документирования) объектов информатизации, выделенных помещений, средств отображения информации и т.д.

5) Предотвращение перехвата информации, передаваемой по каналам связи (защита информации, передаваемой по каналам связи).

6) Предотвращение несанкционированного доступа нарушителей к информации, обрабатываемой средствами вычислительной техники (СВТ) и/или автоматизированными системами (АС), в результате которого возможно неправомерное ознакомление с информацией или ее копирование, или ее уничтожение, блокирование или модификация (защита информации от несанкционированного доступа).

7) Предотвращение несанкционированного снятия копий с носителей информации (защита носителей информации от несанкционированного копирования).

8) Предотвращение специальных программных воздействий на информацию или программное обеспечение технических средств обработки информации (ТСОИ), вызывающих или уничтожение, блокирование, модификацию, копирование компьютерной информации, или нейтрализацию средств защиты информации, или нарушение функционирования ТСОИ (защита информации от вредоносных программ и компьютерных вирусов).

9) Предотвращение программно-технических воздействий на информацию, программное обеспечение ТСОИ или непосредственно ТСОИ, вызывающих или уничтожение, блокирование, модификацию, копирование компьютерной информации, или нейтрализацию средств защиты информации, или нарушение функционирования ТСОИ (защита информации от программно-технических воздействий).

10) Предотвращение перехвата информации, обрабатываемой техническими средствами, электронными устройствами перехвата информации (закладочными устройствами), скрытно внедренными в технические средства обработки информации (выявление электронных устройств перехвата, внедренных в технические средства).

11) Предотвращение перехвата акустической (речевой) информации электронными устройствами перехвата информации (закладочными устройствами), скрытно внедренными в выделенные (защищаемые) помещения (выявление электронных устройств перехвата, внедренных в выделенные помещения).

12) Предотвращение преднамеренного силового электромагнитного воздействия на носители информации и ТСОИ, вызывающего разрушение носителей информации и ТСОИ или сбои в их работе (защита ТСОИ и носителей информации от силового электромагнитного воздействия).

13) Предотвращение несанкционированного доступа на объекты информатизации и в выделенные (защищаемые) помещения (физическая защита объектов информатизации и выделенных помещений).

14) Предотвращение хищения и утраты носителей информации (физическая защита носителей информации).

С практической точки зрения ряд задач, близких по содержанию, целесообразно объединить в группы (направления защиты информации):

**защита информации от разглашения** - защита информации, направленная на предотвращение неконтролируемого распространения (предоставления) защищаемой информации;

**защита информации от утечки по техническим каналам** - защита информации, направленная на предотвращение перехвата информации, обрабатываемой техническими средствами или обсуждаемой в выделенном помещении, техническими средствами разведки (ТСР) или специальными техническими средствами, предназначенными для негласного получения информации (СТС);

**защита информации от несанкционированного доступа** - защита информации, направленная на предотвращение доступа к информации, осуществляемого с нарушением установленных прав и (или) правил доступа к информации с применением штатных средств, предоставляемых СВТ или АС, или средств, аналогичных им по своим функциональному предназначению и техническим характеристикам;

**защита информации от несанкционированных и непреднамеренных воздействий** - защита информации, направленная на предотвращение несанкционированных и непреднамеренных воздействий на защищаемую информацию, приводящих к: модифицированию (искажению, подмены), уничтожению, копированию информации, блокированию доступа к информации, нейтрализации средств защиты информации, сбою функционирования технических или программных средств, повреждению или выводу из строя носителя информации и т.д.

**защита информации, передаваемой по каналам связи** - защита информации, направленная на предотвращение перехвата информации, передаваемой по каналам связи;

**физическая защита объектов** информатизации (выделенных помещений) и носителей информации - защита объектов информатизации (выделенных помещений и т.д.), направленная на предотвращение проникновения на них посторонних лиц, а также носителей информации от их хищения и утраты; контроль

защищенности информации - комплекс мероприятий, направленных на оценку соответствие уровня защиты информации требованиям нормативно-правовых актов и нормативно-методических документов по защите информации. Можно выделить два основных вида контроля защищенности информации: аттестация объектов информатизации (выделенных помещений) по требованиям безопасности информации и аудит безопасности информации (аудит информационной безопасности).

## 2. Основные механизмы защиты информации. Определения и примеры ?

Одним из условий безопасной работы в информационной системе является соблюдение пользователем ряда правил, которые проверены на практике и показали свою высокую эффективность. Их несколько:

1. Использование программных продуктов, полученных законным официальным путем. Вероятность наличия вируса в пиратской копии во много раз выше, чем в официально полученном программном обеспечении.
2. Дублирование информации. Прежде всего, необходимо сохранять дистрибутивные носители программного обеспечения. При этом запись на носители, допускающие выполнение этой операции, должна быть, по возможности, заблокирована. Следует особо позаботиться о сохранении рабочей информации. Предпочтительнее регулярно создавать копии рабочих файлов на съемных машинных носителях информации с защитой от записи. Копируется либо весь файл, либо только вносимые изменения. Последний вариант применим, например, при работе с базами данных.
3. Регулярное обновление системного ПО. Операционную систему необходимо регулярно обновлять и устанавливать все исправления безопасности от Microsoft и других производителей, чтобы устранить существующие уязвимости программного обеспечения.
4. Ограничение доступа пользователей к настройкам операционной системы и системным данным. Для обеспечения стабильной работы системы довольно часто требуется ограничивать возможности пользователей, что можно сделать либо с помощью встроенных средств Windows, либо с помощью специализированных программ, предназначенных для управления доступом к компьютеру.В корпоративных сетях возможно применение групповых политик в сети домена Windows.
5. Для максимально эффективного использования сетевых ресурсов необходимо вводить ограничения доступа авторизованных пользователей к внутренним и внешним сетевым ресурсам и блокировать доступ неавторизованных пользователей.
6. Регулярное использование антивирусных средств. Перед началом работы целесообразно выполнять программы-сканеры и программы-ревизоры. Антивирусные базы должны регулярно обновляться. Кроме того, необходимо проводить антивирусный контроль сетевого трафика.
7. Защита от сетевых вторжений обеспечивается применением программно-аппаратных средств, в том числе: использованием межсетевых экранов, систем обнаружения/предотвращения вторжений IDS/IPS (Intrusion Detection/Prevention System), реализацией технологий VPN (Virtual Private Network).

8. Применение средств аутентификации и криптографии – использование паролей (простых/сложных/неповторяющихся) и методов шифрования. Не рекомендуется использовать один и тот же пароль на разных ресурсах и разглашать сведения о паролях. При написании пароля на сайтах следует быть особенно внимательным, чтобы не допустить ввода своего пароля на мошенническом сайте-двойнике.

9. Особую осторожность следует проявлять при использовании новых (не известных) съемных носителей информации и новых файлов. Новые съёмные носители обязательно должны быть проверены на отсутствие загрузочных и файловых вирусов, а полученные файлы – на наличие файловых вирусов. При работе в распределенных системах или в системах коллективного пользования новые сменные носители информации и вводимые в систему файлы целесообразно проверять на специально выделенных для этой цели компьютерах, не подключенных к локальной сети. Только после всесторонней антивирусной проверки дисков и файлов они могут передаваться пользователям системы.

10. При работе с полученными (например, посредством электронной почты) документами и таблицами целесообразно запретить выполнение макрокоманд средствами, встроенными в текстовые и табличные редакторы (MS Word, MS Excel), до завершения полной проверки этих файлов.

11. Если не предполагается осуществлять запись информации на внешние носители, то необходимо заблокировать выполнение этой операции, например, программно отключив USB-порты.

12. При работе с общими ресурсами в открытых сетях (например, Интернет) использовать только проверенные сетевые ресурсы, не имеющие вредоносного контента. Не следует доверять всей поступающей на компьютер информации – электронным письмам, ссылкам на Web-сайты, сообщениям на Интернет-пейджеры. Категорически не рекомендуется открывать файлы и ссылки, приходящие из неизвестного источника.

## 3. Принципы проектирования защищенных систем ?

Концепция построения защищенных систем:

1. **Принцип интегральности**. Поскольку безопасность является интегральной характеристикой системы, средства защиты должны составлять единый комплекс и должны быть интегрированы в систему обработки информации на уровне базовых средств (ОС, СУБД, сетевые протоколы и т. д.).

2. **Принцип инвариантности**. Используемые при построении защищенной системы модели безопасности и методы защиты должны быть ориентированы исключительно на информационные процессы и не зависеть от особенностей архитектуры используемых системных и прикладных средств.

3. **Принцип унификации**. Средства защиты должны обладать достаточной степенью динамичности и гибкости для использования их при реализации различных моделей защиты и политик безопасности, а также для применения в различных системах обработки информации, независимо от ее назначения и характера обрабатываемой информации. Средства защиты должны быть рассчитаны на минимальное вмешательство со стороны администратора и быть прозрачными для рядового пользователя.

4. **Принцип адекватности**. При проектировании архитектуры защищенной системы и выборе средств защиты необходимо учитывать опыт эксплуатации существующих систем, чтобы устранить предпосылки, послужившие причиной успеха известных случаев нарушения безопасности. Только тогда средства защиты будут действительно адекватны реально действующим угрозам.

5. **Принцип корректности**. Использованные при построении архитектуры защищенной системы принципы, модели безопасности и методы защиты должны быть реализованы корректно, т. е. механизмы их осуществления должны в точности воспроизводить заложенные в них возможности управления информационными потоками. Это означает что, с одной стороны, все использованные теоретические модели и методы должны найти свое отражение на практике, а с другой, что средства защиты должны соответствовать продекларированным и математически доказанным свойствам этих моделей и методов.

Для решения проблем предлагается подчинить процесс реализации моделей безопасности и методов защиты следующим принципам:

1. Модели безопасности, используемые для построения реальных систем, должны обладать свойствами полноты и непротиворечивости — для всех комбинаций запрашиваемых операций и типов взаимодействующих сущностей правила модели должны давать удовлетворительный с точки зрения практического применения результат и не создавать коллизий. Это позволит избежать ситуаций, которые требуют введения исключений из общих правил контроля доступа.

2. Разработка средств защиты должна осуществляться с применением механизмов, позволяющих установить соответствие между проектом системы защиты и реализуемыми программными средствами. В качестве подобного механизма могут выступать формальные высокоуровневые спецификации. Применение подобных механизмов позволит избежать ошибок на стадии проектирования, упорядочить процесс кодирования и применять формальные методы верификации.

3. Поскольку ошибок программирования избежать все равно невозможно, при построении защищенных систем и реализации средств защиты необходимо минимизировать объем программного кода, ошибки в котором являются критичными с точки зрения безопасности. Следствием этого принципа является необходимость минимизации объема функций кода, входящего в состав так называемой TCB (Trusted Computing Base).

4. Поскольку в любой защищенной системе присутствует код, ошибки в котором дискредитируют всю систему защиты (например, подсистемы контроля доступа или шифрования), для таких компонентов необходимо применять соответствующие технологии надежного программирования и верификации.

## 4. Основные стандарты защиты информации. Оранжевая книга ?

Требования безопасности к программным средствам криптографической защиты информации в РБ:

СТБ 34.101.27-2011 "Информационные технологии и безопасность. Требования безопасности к программным средствам криптографической защиты информации".

Стандарт устанавливает общие требования безопасности к программным средствам, которые используются для криптографической защиты информации ограниченного распространения (за исключением государственных секретов).

Стандарт предназначен для использования:

– заказчиками (потребителями) при задании требований безопасности к программным средствам криптографической защиты информации;

– разработчиками при создании программных средств криптографической защиты информации;

– экспертами при оценке надежности программных средств криптографической защиты информации.

**Международные стандарты:**

- **BS 7799-1:2005** — Британский стандарт BS 7799 первая часть. BS 7799 Part 1 — Code of Practice for Information Security Management (Практические правила управления информационной безопасностью) описывает 127 механизмов контроля, необходимых для построения системы управления информационной безопасностью (СУИБ) организации, определённых на основе лучших примеров мирового опыта (best practices) в данной области. Этот документ служит практическим руководством по созданию СУИБ
- **BS 7799-2:2005** — Британский стандарт BS 7799 вторая часть стандарта. BS 7799 Part 2 — Information Security management — specification for information security management systems (Спецификация системы управления информационной безопасностью) определяет спецификацию СУИБ. Вторая часть стандарта используется в качестве критериев при проведении официальной процедуры сертификации СУИБ организации.
- **BS 7799-3:2006** — Британский стандарт BS 7799 третья часть стандарта. Новый стандарт в области управления рисками информационной безопасности
- **ISO/IEC 17799:2005** — «Информационные технологии — Технологии безопасности — Практические правила менеджмента информационной безопасности». Международный стандарт, базирующийся на BS 7799-1:2005.
- **ISO/IEC 27000** — Словарь и определения.
- **ISO/IEC 27001** — «Информационные технологии — Методы обеспечения безопасности — Системы управления информационной безопасностью — Требования». Международный стандарт, базирующийся на BS 7799-2:2005.
- **ISO/IEC 27002** — Сейчас: ISO/IEC 17799:2005. «Информационные технологии — Технологии безопасности — Практические правила менеджмента информационной безопасности». Дата выхода — 2007 год.
- **ISO/IEC 27005** — Сейчас: BS 7799-3:2006 — Руководство по менеджменту рисков ИБ.
- **German Information Security Agency. IT Baseline Protection Manual** — Standard security safeguards (Руководство по базовому уровню защиты информационных технологий).

Оранжевая книга:

Критерии определения безопасности компьютерных систем (англ. Trusted Computer System Evaluation Criteria) — стандарт Министерства обороны США, устанавливающий основные условия для оценки эффективности средств компьютерной безопасности, содержащихся в компьютерной системе. Критерии используются для определения, классификации и выбора компьютерных систем, предназначенных для обработки, хранения и поиска важной или секретной информации.

Критерии, часто упоминающиеся как Оранжевая книга, занимают центральное место среди публикаций «Радужной серии» Министерства обороны США.

Изначально выпущенные Центром национальной компьютерной безопасности — подразделением Агентства национальной безопасности в 1983 году и потом обновлённые в 1985.

Аналогом Оранжевой книги является международный стандарт ISO/IEC 15408, опубликованный в 2005 году. Это более универсальный и совершенный стандарт, но вопреки распространенному заблуждению, он не заменял собой Оранжевую книгу в силу разной юрисдикции документов — Оранжевая книга используется исключительно Министерством Обороны США

## 5. Классификация атак и типов злоумышленника ?

**1. По характеру воздействия**

- пассивное
- активное

Пассивное воздействие на распределенную вычислительную систему - воздействие, которое не оказывает непосредственного влияния на работу системы, но может нарушать ее политику безопасности.
Пассивное удаленное воздействие практически невозможно обнаружить.
Пример: прослушивание канала связи в сети.
Активное воздействие на распределенную вычислительную систему - воздействие, оказывающее непосредственное влияние на работу системы (изменение конфигурации РВС, нарушение работоспособности и т. д.) и нарушающее принятую в ней политику безопасности.

Практически все типы удаленных атак являются активными воздействиями. Особенностью активного воздействия по сравнению с пассивным является принципиальная возможность его обнаружения, так как в результате его осуществления в системе происходят определенные изменения. В отличие от активного, при пассивном воздействии не остается никаких следов.

**2. По цели воздействия**

- нарушение конфиденциальности информации
- нарушение целостности информации
- нарушение работоспособности (доступности) системы

При перехвате информации нарушается её конфиденциальность.

Пример: прослушивание канала в сети.

При искажении информации нарушается её целостность.

Пример: внедрение ложного объекта в РВС.

При нарушении работоспособности не происходит несанкционированного доступа, т.е. сохраняется целостность и конфиденциальность информации, однако доступ к ней легальных пользователей также невозможен.
Пример: отказ в обслуживании (DoS).

**3. По условию начала осуществления воздействия**

- Атака по запросу от атакуемого объекта
- Атака по наступлению ожидаемого события на атакуемом объекте
- Безусловная атака

В случае запроса атакующий ожидает передачи от потенциальной цели атаки запроса определенного типа, который и будет условием начала осуществления воздействия.
Инициатором осуществления начала атаки является атакуемый объект.
Пример: DNS- и ARP-запросы в стеке TCP/IP.

В случае наступления события, атакующий осуществляет постоянное наблюдение за состоянием операционной системы удаленной цели атаки и при возникновении определенного события в этой системе начинает воздействие. Инициатором осуществления начала атаки является атакуемый объект.
Пример: прерывание сеанса работы пользователя с сервером в сетевых ОС без выдачи команды LOGOUT.

В случае безусловной атаки начало её осуществления безусловно по отношению к цели атаки, то есть атака осуществляется немедленно и безотносительно к состоянию системы и атакуемого объекта. Следовательно, в этом случае атакующий является инициатором начала осуществления атаки.

**4. По наличию обратной связи с атакуемым объектом**

- с обратной связью
- без обратной связи (однонаправленная атака)

Атака с обратной связью - атака, во время которой атакующий получает ответ от атакуемого объекта на часть своих действий. Эти ответы нужны, чтобы иметь возможность продолжить атаку и/или осуществлять её более эффективно, реагируя на изменения, происходящие на атакуемой системе.
Атака без обратной связи - атака, происходящая без реакции на поведение атакуемой системы.
Пример: отказ в обслуживании (DoS).

**5. По расположению атакующего относительно атакуемого объекта**

- внутрисегментное
- межсегментное

Внутрисегментная атака - атака, при которой субъект и объект атаки находятся внутри одного сегмента сети, где сегмент - есть физическое объединение станций с помощью коммуникационных устройств не выше канального уровня.
Межсегментная атака - атака, при которой субъект и объект атаки находятся в разных сегментах сети.

**6. По количеству атакующих**

- распределённая
- нераспределённая

Распределённая атака - атака, производимая двумя или более атакующими на одну и ту же вычислительную систему, объединёнными единым замыслом и во времени.
Нераспределённая атака проводится одним атакующим.

**7. По уровню эталонной модели ISO/OSI, на котором осуществляется воздействие**

- физический
- канальный
- сетевой
- транспортный
- сеансовый
- представительный
- прикладной

## 6. Определение и классификация типов аутентификации ?

**Идентификация** – сопоставление пользователя и его данных (в частности, его идентификатора). Идентификатор – имя пользователя.

**Аутентификация** – Сопоставление пользователя (уже идентифицированного) и его атрибутов безопасности (на основе секрета: пароль, биометрические данные и т.д.)

Для корректной аутентификации пользователя необходимо, чтобы пользователь предъявил *аутентификационную информацию* – некую уникальную информацию, которой должен обладать только он и никто иной.

Существует три основных типа аутентификационной информации:

1. Проверяемый пользователь *знает* некую уникальную информацию. Пример: парольная аутентификация.
2. Пользователь *имеет* некий предмет с уникальными характеристиками или содержимым. Примеры: смарт-карта, USB-токен и т.д.
3. Аутентификационная информация *является неотъемлемой частью* пользователя. Пример: отпечаток пальца и другие виды биометрической аутентификации (*биометрической* называется аутентификация пользователя по его биометрическим признакам).

**Парольная аутентификация**

В настоящее время парольная аутентификация является наиболее распространенной, прежде всего, благодаря своему единственному достоинству – простоте использования.

Однако, парольная аутентификация имеет множество недостатков:

1. В отличие от случайно формируемых криптографических ключей (которые, например, может содержать уникальный предмет, используемый для аутентификации), пароли пользователя бывает возможно подобрать из-за достаточно небрежного отношения большинства пользователей к формированию пароля. Часто встречаются случаи выбора пользователями легко предугадываемых паролей, например:
   - пароль эквивалентен идентификатору (имени) пользователя (или имени пользователя, записанному в обратном порядке, или легко формируется из имени пользователя и т.д.);
   - паролем является слово или фраза какого-либо языка; такие пароли могут быть подобраны за ограниченное время путем «словарной атаки» - перебора всех слов согласно словарю, содержащему все слова и общеупотребительные фразы используемого языка;
   - достаточно часто пользователи применяют короткие пароли, которые взламываются методом «грубой силы», т.е. простым перебором всех возможных вариантов.
2. Существуют и свободно доступны различные утилиты подбора паролей, в том числе, специализированные для конкретных широко распространенных программных средств. Например, на сайте www.lostpassword.com описана утилита подбора пароля для документа Microsoft Word 2000 (Word Password Recovery Key), предназначенная для восстановления доступа к документу, если его владелец забыл пароль. Несмотря на данное полезное назначение, ничто не мешает использовать эту и подобные ей утилиты для взлома чужих паролей

3. Пароль может быть получен путем применения насилия к его владельцу.
4. Пароль может быть подсмотрен или перехвачен при вводе.

**Аутентификация с помощью уникального предмета**

В большинстве случаев аутентификация с помощью уникального предмета обеспечивает более серьезную защиту, чем парольная аутентификация. Предметы, используемые для аутентификации, можно условно разделить на следующие две группы:

1. «Пассивные» предметы, которые содержат аутентификационную информацию (например, некий случайно генерируемый пароль) и передают ее в модуль аутентификации по требованию. При этом, аутентификационная информация может храниться в предмете как в открытом (примеры: магнитные карты, смарт-карты с открытой памятью, электронные таблетки Touch Memory), так и в защищенном виде (смарт-карты с защищенной памятью, USB-токены). В последнем случае требуется ввод PIN-кода для доступа к хранящимся данным, что автоматически превращает предмет в средство двухфакторной аутентификации.
2. «Активные» предметы, которые обладают достаточными вычислительными ресурсами и способны активно участвовать в процессе аутентификации (примеры: микропроцессорные смарт-карты и USB-токены). Эта возможность особенно интересна при удаленной аутентификации пользователя, поскольку на основе таких предметов можно обеспечить *строгую* аутентификацию. Под этим термином скрывается такой вид аутентификации, при котором секретная информация, позволяющая проверить подлинность пользователя, не передается в открытом виде.

Аутентификация с помощью уникальных предметов обладает и рядом недостатков:

1. Предмет может быть похищен или отнят у пользователя.
2. В большинстве случаев требуется специальное оборудование для работы с предметами.
3. Теоретически возможно изготовление копии или эмулятора предмета.

**Биометрическая аутентификация**

Биометрическая аутентификация основана на уникальности ряда характеристик человека. Наиболее часто для аутентификации используются следующие характеристики:

1. Отпечатки пальцев.
2. Узор радужной оболочки глаза и структура сетчатки глаза.
3. Черты лица.
4. Форма кисти руки.
5. Параметры голоса.
6. Схема кровеносных сосудов лица.
7. Форма и способ подписи.

В процессе биометрической аутентификации эталонный и предъявленный пользователем образцы сравнивают с некоторой погрешностью, которая определяется и устанавливается заранее. Погрешность подбирается для установления оптимального соотношения двух основных характеристик используемого средства биометрической аутентификации:

1. FAR (False Accept Rate) – коэффициент ложного принятия (т.е. некто успешно прошел аутентификацию под именем легального пользователя).
2. FRR (False Reject Rate) – коэффициент ложного отказа (т.е. легальный пользователь системы не прошел аутентификацию).

Обе величины измеряются в процентах и должны быть минимальны. Следует отметить, что величины являются обратнозависимыми, поэтому аутентифицирующий модуль при использовании биометрической аутентификации настраивается индивидуально – в зависимости от используемой биометрической характеристики и требований к качеству защиты ищется некая «золотая середина» между данными коэффициентами. Серьезное средство биометрической аутентификации должно позволять настроить коэффициент FAR до величин порядка 0,01 – 0,001 % при коэффициенте FRR до 3 – 5%.

В зависимости от используемой биометрической характеристики, средства биометрической аутентификации имеют различные достоинства и недостатки. Например, использование отпечатков пальцев наиболее привычно и удобно для пользователей, но, теоретически, возможно создание «искусственного пальца», успешно проходящего аутентификацию.

Общий же недостаток биометрической аутентификации – необходимость в оборудовании для считывания биометрических характеристик, которое может быть достаточно дорогостоящим.

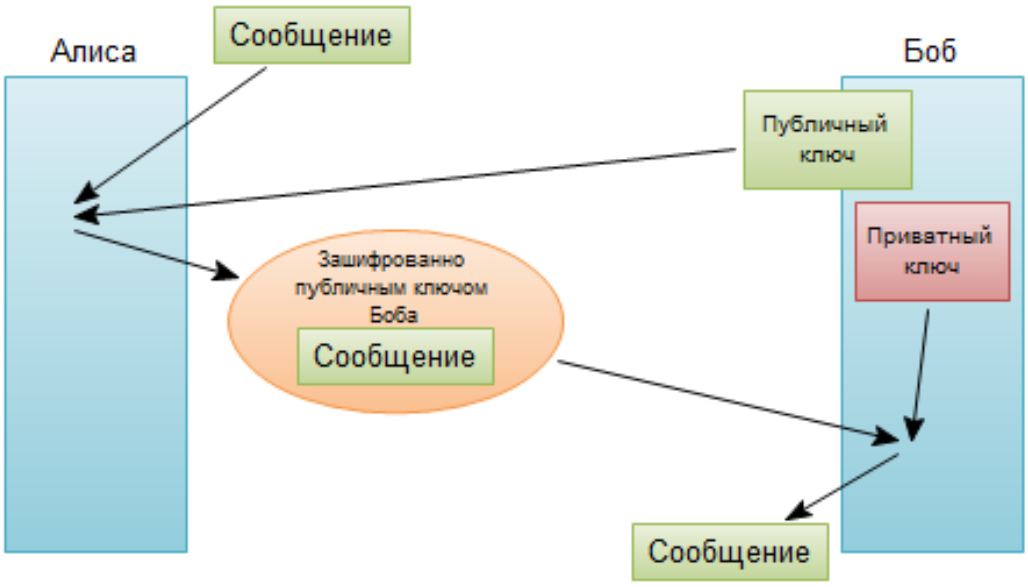
## 7. Аутентификация с паролем и асимметричным шифрованием. Блок-схема и пример ?

Аутентификация с паролем в предыдущем вопросе (см. выше).

**Асимметричное шифрование**

асимметричный алгоритм предполагает под собой наличие двух ключей — публичного и приватного. То есть сообщение шифруется публичным ключом, а расшифровывается приватным и ни как иначе. Собственно именно эту концепцию сформулировал Диффи-Хеллман.

В общем суть данного алгоритма заключается в том, что принимающая сторона перед приемкой сообщения генерирует пару ключей на основе алгоритма модульной арифметики (принцип такой же как и в алгоритме Диффи-Хеллмана), собственно приватный и публичный ключ. Отправитель перед отправкой получает публичный ключ и шифрует сообщение данным ключом, после чего данное сообщение можно расшифровать только приватным ключом, который хранится в секрете у принимающей стороны.



Представьте, что Алиса проектирует замок и ключ. Она бдительно охраняет ключ, но при этом изготавливает тысячи дубликатов замков и рассылает их по почтовым отделениям по всему миру. Если Боб хочет послать сообщение, он кладет его в коробку, идет на местный почтамт, просит «замок Алисы» и запирает им коробку. Теперь уже ему не удастся открыть коробку, но когда коробку получит Алиса, она сможет открыть ее своим единственным ключом.

Замок и защелкивание его, чтобы он закрылся, эквивалентны общему ключу для зашифровывания, поскольку все имеют доступ к замкам и все могут воспользоваться замком, чтобы закрыть сообщение в коробке. Ключ от замка эквивалентен секретному ключу для расшифровывания, потому что он имеется только у Алисы, только она сможет открыть замок, и только она сможет получить доступ к находящемуся в коробке сообщению.

Есть несколько алгоритмов реализующих асимметричное шифрование. Самый известный из них — RSA.

## 8. Атаки на аутентификацию. Примеры и требования к реализации ?

7 наиболее распространенных атак.

### 1. Анализ сетевого трафика

Данный вид атаки направлен в первую очередь на получение пароля и идентификатора пользователя путем "прослушивания сети". Реализуется это с помощью sniffer – специальная программа-анализатор, которая перехватывает все пакеты, идущие по сети. И если протокол, например, FTP или TELNET, передает аутентификационную информацию в открытом виде, то злоумышленник легко получает доступ к учетной записи пользователя.

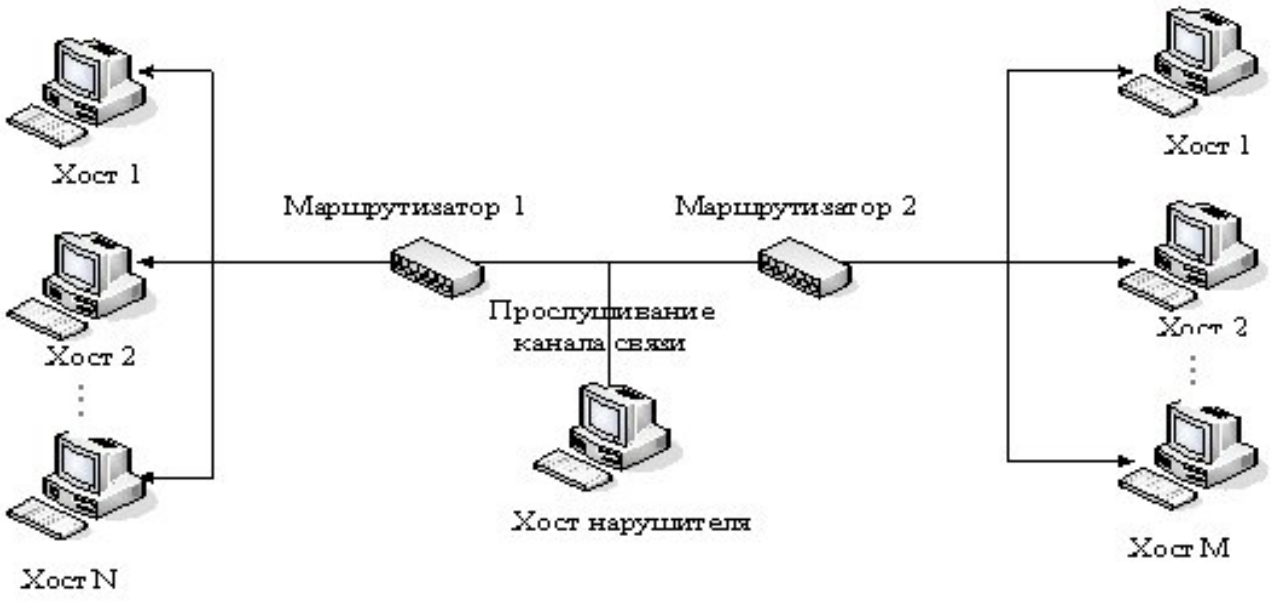



**Рис. 4.3.** Схема реализации угрозы “Анализ сетевого трафика”

### 2. Сканирование сети

Суть данной атаки состоит в сборе информации о топологии сети, об открытых портах, используемых протоколах и т.п. Как правило, реализация данной угрозы предшествует дальнейшим действиям злоумышленника с использованием полученных в результате сканирования данных.

**3. Угроза выявления пароля**

Целью атаки является преодоление парольной защиты и получении НСД к чужой информации. Методов для кражи пароля очень много: простой перебор всех возможных значений пароля, перебор с помощью специальных программ (атака словаря), перехват пароля с помощью программы-анализатора сетевого трафика.

**4. Подмена доверенного объекта сети и передача по каналам связи сообщений от его имени с присвоением его прав доступа.**

Доверенный объект – это элемент сети, легально подключенный к серверу.Такая угроза эффективно реализуется в системах, где применяются нестойкие алгоритмы идентификации и аутентификации хостов, пользователей и т.д.

Могут быть выделены две разновидности процесса реализации указанной угрозы: с установлением и без установления виртуального соединения.

Процесс реализации с установлением виртуального соединения состоит в присвоении прав доверенного субъектавзаимодействия, что позволяет нарушителю вести сеанс работы с объектом сети от имени доверенного субъекта. Реализация угрозы данного типа требует преодоления системы идентификации и аутентификации сообщений (например, атака rsh-службы UNIX-хоста).

Процесс реализации угрозы без установления виртуального соединения может иметь место в сетях, осуществляющих идентификацию передаваемых сообщений только по сетевому адресу отправителя. Сущность заключается в передаче служебных сообщений от имени сетевых управляющих устройств (например, от имени маршрутизаторов) об изменении маршрутно-адресных данных.

В результате реализации угрозы нарушитель получает права доступа, установленные его пользователем для доверенного абонента, к техническому средству ИСПД– цели угроз.

**5. Навязывание ложного маршрута сети**

Данная атака стала возможной из-за недостатков протоколов маршрутизации (RIP, OSPF, LSP) и управления сетью (ICMP, SNMP), таких как слабая аутентификация маршрутизаторов. Суть атаки состоит в том, что злоумышленник, используя уязвимости протоколов, вносит несанкционированные изменения в маршрутно-адресные таблицы.

**6. Внедрение ложного объекта сети**

Когда изначально объекты сети не знают информацию друг о друге, то для построения адресных таблиц и последующего взаимодействия, используется механизм запрос (как правило, широковещательный) - ответ с искомой информацией. При этом если нарушитель перехватил такой запрос, то он может выдать ложный ответ, изменить таблицу маршрутизации всей сети, и выдать себя за легального субъекта сети. В дальнейшем все пакеты, направленные к легальному субъекту, будут проходить через злоумышленника.

**7. Отказ в обслуживании**

Этот тип атак является одним из самых распространенных в настоящее время. Целью такой атаки является отказ в обслуживании, то есть нарушение доступности информации для законных субъектов информационного обмена.

При защите персональных данных должно быть обеспечено:

1. предотвращение несанкционированного доступа к ПД и передачи их лицам, не имеющим соответствующих прав;
2. своевременное обнаружение фактов НСД(несанкционированный доступ) к ПД;
3. предотвращение воздействия на технические средства обработки ПД, которое может нарушить их функционирование;
4. возможность немедленного восстановления ПД в случае их модификации или уничтожения в результате НСД.
5. постоянный контроль уровня защищенности персональных данных.

## 9. Аутентификация OpenID ?

OpenID - это открытая децентрализованная система, которая позволяет пользователю использовать одну учетную запись для аутентификации в различных системах. При этом эти системы не обязательно должны быть связаны друг с другом. Начала свое развитие в 2005 году, одним из создателей LiveJournal.

Основная цель использования OpenID – это упрощение регистрации пользователя. Упрощение достигается в децентрализованной системе единого входа, которая позволяет использовать один логин и пароль на большом количестве сайтов. На сайтах, поддерживающих OpenID, пользователям не приходится каждый раз регистрироваться и помнить данные для каждого сайта. Вместо этого им достаточно быть зарегистрированными на сайте «провайдера идентификации» OpenID (предоставляющего идентификатор). Так как технология OpenID децентрализованная, то любой сайт может использовать программное обеспечение OpenID в качестве средства входа. Удобство OpenID также заключается в том, что при авторизации по OpenID, посещаемому сайту сразу могут передаваться личные данные пользователя (email, Имя, пол, дата рождения и др) и ему не придется каждый раз вводить данную информацию — достаточно это сделать один раз на сайте OpenID провайдера.

Существует 3 важных понятия в OpenID аутентификации:

1. **User** - пользователь, который делает запрос на аутентификацию
2. **Provider** - система, которая хранит данные пользователя, и обрабатывает запросы OpenID
3. **RelyingParty** - зависимая сторона, которая хочет проверить подлинность пользователя.

Процесс аутентификации по OpenID описывается следующими шагами:

1. Пользователь пытается получить доступ к закрытому ресурсу зависимой стороны (Relying Party);
2. Зависимая сторона запрашивает у пользователя логин;
3. Пользователь отправляет свой URL OpenID;
4. Зависимая сторона перенаправляет пользователя к провайдеру OpenID для проверки подлинности;
5. Пользователь вводит свои данные для входа;
6. Провайдер OpenID аутентифицирует пользователя;

7. В случае успешной аутентификации, провайдер перенаправляет пользователя обратно к зависимой стороне, включая идентификационную информацию;

8. Зависимая сторона проверяет токен от провайдера и допускает пользователя.

## 10.Определение и основные принципы токенизации ?

- **Токен**. Случайные данные, используемые в качестве некоего суррогата для других данных. В токене нет математических зависимостей между случайными и оригинальными данными, поэтому оригинальные данные не могут быть определены по значению токена. Ассоциация между оригинальными и случайными данными ведется только в базе данных и сопоставление оригинальных данных и суррогата может быть проведено только в ней.
- **Токенизация.** Процесс создания токенов.

Её принцип заключается в подмене реальных конфиденциальных данных некими значениями – токенами. Токенизация в чем-то схожа с шифрованием – обе технологии занимаются маскированием реальных данных, но подход к этому процессу в корне отличается. В случае шифрования, процесс сокрытия данных обратим при наличии правильного ключа. Любой человек, получив ключ шифрования, сможет восстановить исходные данные.

В случае с токенизацией, процесс не является обратимым, ведь вместо данных при токенизации используется токен, не несущий в себе никакой конфиденциальной информации, и лишь логически связанный с реальными данными, которые хранятся в хорошо защищенной базе данных. При этом токен может иметь тот же формат (размер и структура), что и оригинальные данные, что позволяет минимизировать необходимость внесения изменений в работающие с ним приложения. Но при этом похищать токен бессмысленно, так как он не поможет получить никакие реальные конфиденциальные данные.

Главное преимущество токенизации в том, что токены могут полноценно использоваться внутри среды их применения, но совершенно бесполезны вне ее. Токенизация идеально подходит для защиты конфиденциальных данных, таких как номера кредитных карт, номера социального страхования, а также любых других данных, которые злоумышленники регулярно похищают с целью использования или продажи на “черном рынке”. Если злоумышленникам не удастся взломать сам сервер токенизации, чтобы получить связанные с токенами реальные данные, то похищенные токены не дадут им ровным счетом никаких возможностей их использовать.

## 11.Методы токенизации с примерами ?

смотреть 10 вопрос.

Например, одним из самых распространенных примеров использования токенизации является ее применение для платеже с использованием платежных карт. Использование токена вместо значения номера платежной карты позволяет провести отслеживание транзакции и ее выполнение без риска компрометации реальных данных карты. Доступ к реальному номеру понадобится только в процессинговом центре. Ну а в том случае, если процессинговый центр также использует токенизацию, то возможно проведение транзакции вообще без использования реальных данных.

## 12.Сравнение токенизации и шифрования ?

смотреть 10 вопрос.

Токенизация позволяет обеспечить защиту данных при потенциально низкой стоимости. Добавление шифрования в системы – особенно в уже действующие – значительно увеличивает нагрузку на них. Внесение изменений в приложения для внедрения шифрования может значительно повысить расходы, снизить производительность и увеличить требования к программистам и администраторам систем. Кроме того, в случае использования разнородных систем, возникает необходимость шифровании, расшифровании и повторном шифровании данных в различных местах, что создает дополнительные уязвимости, которые могут быть использованы злоумышленниками. Причем, чем больше ключей используется в системе, тем больше возможностей для ее атаки. Работа с ключами особенно опасна, учитывая распространенность вредоносного ПО для перехвата данных непосредственно в оперативной памяти, которое позволяет получить доступ к ключам даже не имея административных привилегий на зараженной машине.

Помимо минимизации требований к внесению изменений в используемые приложения, токенизация снижает риски для конфиденциальных данных. При правильной реализации, приложения смогут использовать токены во всей системе и обращаться к защищенным конфиденциальным данным только в случае крайней необходимости. Приложения могут хранить, использовать и совершать транзакции, оперируя только токеном и не подвергая реальные данные риску. Хотя некоторые операции, конечно, требуют обращения к реальным данным, токенизация позволяет минимизировать их использование.

## 13.Стандарт PCI DSS ?

Стандарт PCI DSS был разработан Советом по стандартам безопасности данных индустрии платежных карт PCI SSC (Payment Card Industry Security Standards Council) и регламентирует определенный перечень требований к обеспечению безопасности данных платежных карт (ДПК), которые затрагивают как техническую, так и организационную сторону организаций.

В первую очередь стандарт определяет требования к организациям, в информационной инфраструктуре которых хранятся, обрабатываются или передаются данные платёжных карт, а также к организациям, которые могут влиять на безопасность этих данных. Начиная с середины 2012 года все организации, вовлечённые в процесс хранения, обработки и передачи ДПК должны соответствовать требованиям, определяемым PCI DSS

Если же несколько углубиться, стандарт требует прохождения порядка 440 проверочных процедур, которые должны дать положительный результат при проверке на соответствие требованиям. Ниже приведены, в общих чертах, 12 разделов требований:

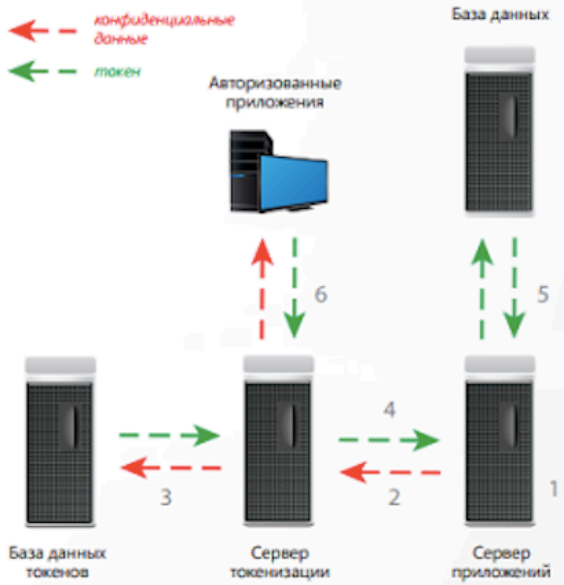
| Требования стандарта PCI DSS | |
|---|---|
| 1. Защита вычислительной сети | 7. Управление доступом к данным о держателях карт |
| 2. Конфигурация компонентов информационной инфраструктуры | 8. Механизмы аутентификации |
| 3. Защита хранимых данных о держателях карт | 9. Физическая защита информационной инфраструктуры |
| 4. Защита передаваемых данных о держателях карт | 10. Протоколирование событий и действий |
| 5. Антивирусная защита информационной инфраструктуры | 11. Контроль защищённости информационной инфраструктуры |
| 6. Разработка и поддержка информационных систем | 12. Управление информационной безопасностью |

## 14.Схема работы сервиса токенизации. Узкие места ?

см 10 вопрос.

### Базовая архитектурная модель

1. *Приложение собирает или генерирует части конфиденциальных данных.*
2. *Данные немедленно передаются на сервер токенизации – они не хранятся локально.*
3. *Сервер токенизации генерирует для данных случайные или полуслучайные токены. Конфиденциальные данные и токены хранятся в защищенной базе данных (обычно в зашифрованном виде).*
4. *Сервер токенизации возвращает токен приложению.*
5. *Приложение сохраняет токен, а не конфиденциальные данные. Токен используется для большинства операций приложения.*

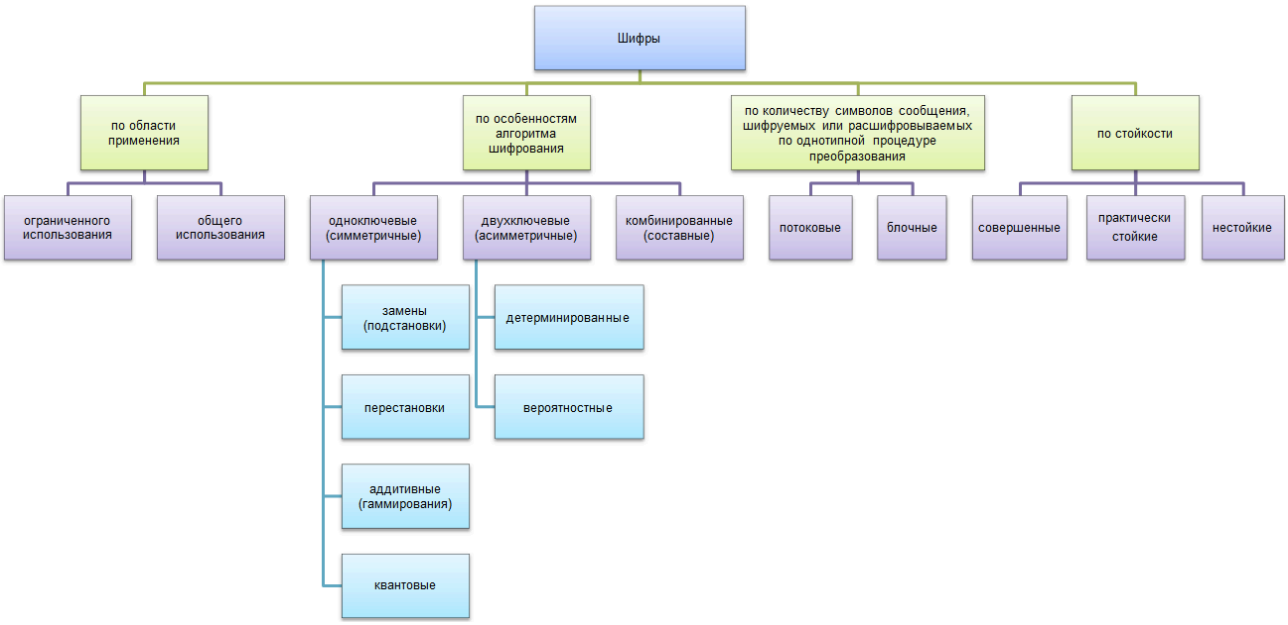



6. *При необходимости получить реальные конфиденциальные данные, авторизованное приложение или пользователь могут запросить их. Конфиденциальные данные не хранятся в локальных базах данных и в большинстве случаев доступ к ним значительно ограничен, что значительно снижает риск утечки.*

Концептуально, токенизация – это очень простой процесс. Фактически это просто замена одного значения (конфиденциальных данных) другим значением (токеном) – которой не несет в себе никакой важной информации. Но токен небесполезен – он имеет большой значение внутри среды в которой он должен работать(или даже во множестве сред), но вне ее теряет всякий смысл.

Прокси-агенты: программные агенты, которые перехватывают обращения к базе данных (например, путем замены компонентов ODRC или JDBC). В этой модели использования, процесс или приложение, отправляющее конфиденциальные данные могут, в принципе, не знать о процессе токенизации. Приложение просто отсылает данные, как делает это обычно, а прокси-агент перехватывает запрос. Агент заменяет конфиденциальные данные токеном, а далее пересылает их серверу токенизации или поддерживаемому серверу приложений. Эта модель минимизирует необходимость во внесении изменений в приложения, так как вам нужно заменить соединение между приложением и базой данных, а новое программное обеспечение будет самостоятельно управлять токенизацией. Но в этой модели есть **потенциальные узкие места** и проблема отказоустойчивости, так как все решения работают последовательно с используемыми системами обработки транзакций

## 15.Классификация видов шифрования. Примеры и определения ?




Примеры в конце ответа…

**I. По области применения** различают шифры ограниченного и общего использования.

Стойкость **шифров ограниченного использования** основывается на сохранении в секрете алгоритма криптографического преобразования в силу его уязвимости, малого количества ключей или отсутствия таковых (секретные кодовые системы).

Стойкость **шифров общего использования** основывается на секретности ключа и сложности его подбора потенциальным противником.

**II. По особенностям алгоритма шифрования** шифры общего использования можно разделить на следующие виды.

В **одноключевых** системах для шифрования и дешифрования используется один и тот же ключ.

В шифрах перестановки все буквы открытого текста остаются в шифрограмме, но меняют свои позиции. В шифрах замены наоборот, позиции букв в шифрограмме остаются теми же, что и у открытого текста, но символы открытого текста заменяются символами другого алфавита.

В аддитивных шифрах буквы алфавита заменяются числами, к которым затем добавляются числа секретной случайной (псевдослучайной) числовой последовательности (гаммы), после чего берется остаток от деления по модулю (операция mod). Если исходное сообщение и гамма представляются в битовом виде, то при шифровании и расшифровании применяется логическая операция «Исключающее ИЛИ» (XOR, сложение по модулю 2).

Квантовая криптография вносит в процесс шифрования естественную неопределенность квантового мира. Процесс отправки и приёма информации выполняется посредством объектов квантовой механики (например, при помощи электронов в электрическом токе или фотонов в линиях волоконно-оптической связи). Самым ценным свойством этого вида шифрования является то, что при

посылке сообщения отправляющая и принимающая сторона с достаточно большой вероятностью могут установить факт перехвата противником зашифрованного сообщения.

В **двухключевых** системах для шифрования и дешифрования используется два совершено разных ключа. В детерминированных шифрах при шифровании одного и того же сообщения одним и тем же ключом всегда будет получаться один и тот же шифртекст. В вероятностных шифрах в процедуре шифрования используется дополнительная случайная величина (число) - в результате при шифровании одного и того же исходного сообщения одним и тем же ключом могут получиться разные шифртексты, которые при расшифровке дадут один и тот же результат (исходное сообщение).
Комбинированные (составные) методы предполагают использование для шифрования сообщения сразу нескольких методов (например, сначала замена символов, а затем их перестановка).

**III. По количеству символов сообщения (или его кодовой замены), шифруемых или расшифровываемых по однотипной процедуре преобразования** различают:

- **потоковые шифры** – процедура преобразование применяется к отдельному символу сообщения;

- **блочные шифры** – процедура преобразование применяется к набору (блоку) символов сообщения.

Отличить потоковый шифр от блочного можно по следующему признаку - если в результате разбиения исходного сообщения на отдельные элементарные символы и применения к ним однотипной процедуры преобразования получаемая шифрограмма эквивалентна той, которая получается при применении преобразования «как будто ко всему исходному сообщению», то шифр потоковый, иначе блочный.

**IV. По стойкости шифры** делятся на три группы:

- **совершенные (абсолютно стойкие, теоретически стойкие)** – шифры, заведомо неподдающиеся вскрытию (при правильном использовании). Дешифрование секретного сообщения приводит к нескольким осмысленным равновероятным открытым сообщениям;

- **практически (вычислительно, достаточно) стойкие** – шифры, вскрытие которых за приемлемое время невозможно на современном или перспективном уровне вычислительной техники. Практическая стойкость таких систем базируется на теории сложности и оценивается исключительно на какой-то определенный момент времени с двух позиций:

  - вычислительная сложность полного перебора;
  - известные на данный момент слабости (уязвимости) и их влияние на вычислительную сложность;
  - нестойкие шифры.

**Пару примеров:**

- Симметричные (с секретным, единым ключом, одноключевые, single-key).
  - Потоковые (шифрование потока данных):
    - с одноразовым или бесконечным ключом (infinite-key cipher);
    - с конечным ключом (система Вернама - Vernam);
    - на основе генератора псевдослучайных чисел (ПСЧ).
  - Блочные (шифрование данных поблочно):

- Шифры перестановки (permutation, P-блоки);
- Шифры замены (подстановки, substitution, S-блоки):
- моноалфавитные (код Цезаря);
- полиалфавитные (шифр Видженера, цилиндр Джефферсона, диск Уэтстоуна, Enigma);

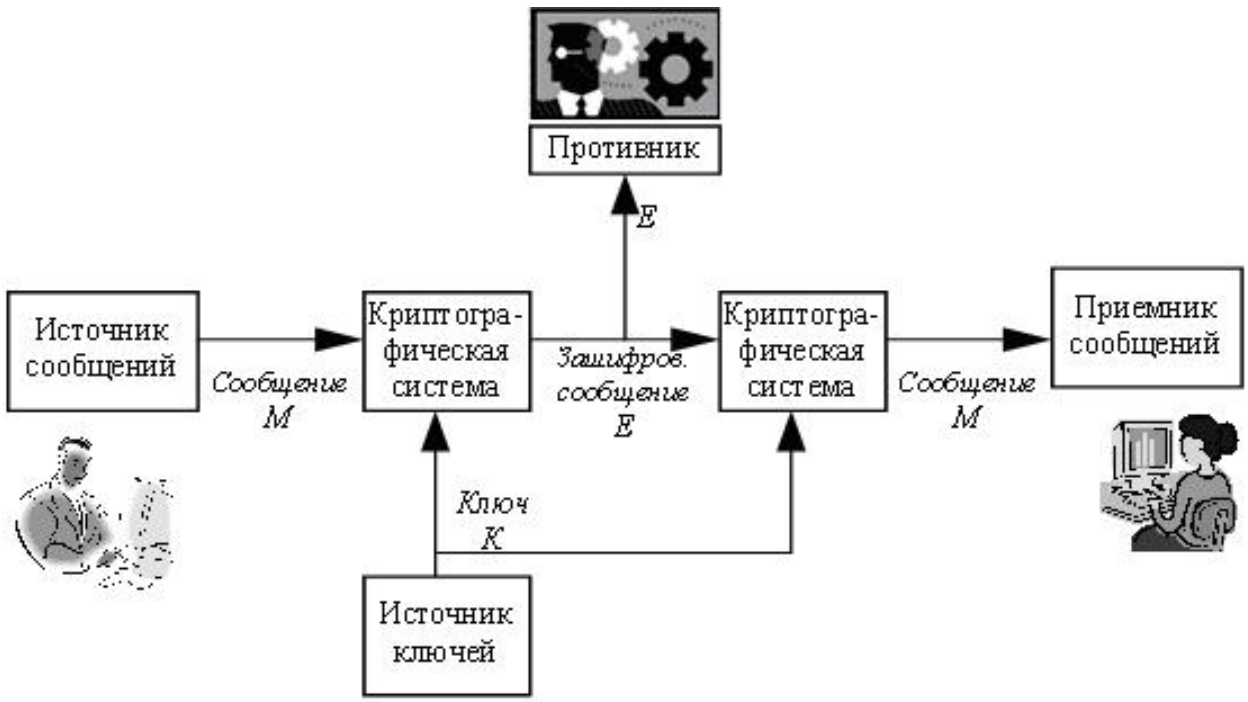
  - Составные:
    - Lucipher (фирма IBM, США);
    - DES (Data Encryption Standard, США);
    - FEAL-1 (Fast Enciphering Algoritm, Япония);
    - IDEA/IPES (International Data Encryption Algorithm/
    - Improved Proposed Encryption Standard, фирма Ascom-Tech AG, Швейцария);
    - B-Crypt (фирма British Telecom, Великобритания);
    - ГОСТ 28147-89 (СССР); * Skipjack (США).

- Асимметричные (с открытым ключом, public-key):
  - Диффи-Хеллман DH (Diffie, Hellman);
  - Райвест-Шамир-Адлеман RSA (Rivest, Shamir, Adleman);
  - Эль-Гамаль ElGamal.

## 16. Системы симметрического шифрования. Режимы шифрования. Блочные и поточные шифры ?

есть инфа в 15 вопросе +

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



зашифровать, так и расшифровать сообщение. Схематичная структура такой системы представлена на рис:

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

Классическими примерами таких алгоритмов являются **симметричные криптографические алгоритмы**, перечисленные ниже:

- Простая перестановка
- Одиночная перестановка по ключу
- Двойная перестановка
- Перестановка "Магический квадрат"

**Простая перестановка**

Простая перестановка без ключа — один из самых простых методов шифрования. Сообщение записывается в таблицу по столбцам. После того, как открытый текст записан колонками, для образования шифртекста он считывается по строкам. Для использования этого шифра отправителю и получателю нужно договориться об общем ключе в виде размера таблицы. Объединение букв в группы не входит в ключ шифра и используется лишь для удобства записи несмыслового текста.

**Одиночная перестановка по ключу**

Более практический метод шифрования, называемый одиночной перестановкой по ключу, очень похож на предыдущий. Он отличается лишь тем, что колонки таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы.

**Двойная перестановка**

Для дополнительной скрытности можно повторно шифровать сообщение, которое уже было зашифровано. Этот способ известен под названием двойная перестановка. Для этого размер второй таблицы подбирают так, чтобы длины её строк и столбцов отличались от длин в первой таблице. Лучше всего, если они будут взаимно простыми. Кроме того, в первой таблице можно переставлять столбцы, а во второй строки. Наконец, можно заполнять таблицу зигзагом, змейкой, по спирали или каким-то другим способом. Такие способы заполнения таблицы если и не усиливают стойкость шифра, то делают процесс шифрования гораздо более занимательным.

**Перестановка «Магический квадрат»**

Магическими квадратами называются квадратные таблицы со вписанными в их клетки последовательными натуральными числами от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число. Подобные квадраты широко применялись для вписывания шифруемого текста по приведенной в них нумерации. Если потом выписать содержимое таблицы по строкам, то получалась шифровка перестановкой букв. На первый взгляд кажется, будто магических квадратов очень мало. Тем не менее, их число очень быстро возрастает с увеличением размера квадрата. Так, существует лишь один магический квадрат размером 3 x 3, если не принимать во внимание его повороты. Магических квадратов 4 x 4 насчитывается уже 880, а число магических квадратов размером 5 x 5 около 250000. Поэтому магические квадраты больших размеров могли быть хорошей основой для надежной системы шифрования того времени, потому что ручной перебор всех вариантов ключа для этого шифра был немыслим.

## 17.Системы асимметрического шифрования ?

Асимметричное шифрование или шифрование с открытым ключом – это шифрование которое использует свойства функций с секретом, разработанных Диффи и Хеллманом.
Эти системы характеризуются наличием у каждого абонента двух ключей: открытого и закрытого (секретного). При этом открытый ключ передается всем участникам секретных переговоров. Таким образом, решаются две проблемы: нет нужды в секретной доставке ключа (так как при помощи открытого ключа нельзя расшифровать сообщения, для этого же открытого ключа зашифрованные, и, следовательно, перехватывать открытый ключ нет смысла); отсутствует также квадратичная зависимость числа ключей от числа пользователей - для n пользователей требуется 2n ключей.

Первым шифром, разработанным на принципах асимметричного шифрования, является шифр RSA.

Другим шифром, использующим асимметричное шифрование, является DSS. Стандарт DSS (Digital Signature Standard) одобрен правительством США. Длина используемого ключа варьируется в пределах от 512 до 1024 бит. DSS предназначен для создания цифровой подписи (см. далее раздел о цифровой подписи), но не для закрытия информации. В стандарте DSS найдены некоторые слабые места защиты, вследствие чего он не так широко распространен.

## 18.Схема ЭЦП ?

**Электро́нная цифровая по́дпись** (ЭЦП) — реквизит электронного документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи (целостность), принадлежность подписи владельцу сертификата ключа подписи (авторство), а в случае успешной проверки подтвердить факт подписания электронного документа (неотказуемость).

Использование ЭП предполагается для осуществления следующих важных направлений в электронной экономике:

- Полный контроль целостности передаваемого электронного платежного документа: в случае любого случайного или преднамеренного изменения документа цифровая подпись станет недействительной, потому как вычисляется она по специальному алгоритму на основании исходного состояния документа и соответствует лишь ему.
- Эффективная защита от изменений (подделки) документа. ЭП даёт гарантию, что при осуществлении контроля целостности будут выявлены всякого рода подделки. Как следствие, подделывание документов становится нецелесообразным в большинстве случаев.
- Фиксирование невозможности отказа от авторства данного документа. Это аспект вытекает из того, что вновь создать правильную электронную подпись можно лишь в случае обладания так называемым закрытым ключом, который, в свою очередь, должен быть известен только владельцу этого самого ключа (автору документа). В этом случае владелец не сможет сформировать отказ от своей подписи, а значит — от документа.
- Формирование доказательств подтверждения авторства документа: исходя из того, что создать корректную электронную подпись можно, как указывалось выше, лишь зная Закрытый ключ, а он по определению должен быть известен только владельцу-автору документа, то владелец ключей может однозначно доказать своё авторство подписи под документом. Более того, в документе могут быть подписаны только отдельные поля документа, такие как «автор», «внесённые изменения», «метка времени» и т. д. То есть, может быть доказательно подтверждено авторство не на весь документ.

Существует несколько схем построения цифровой подписи:

- На основе алгоритмов симметричного шифрования. Данная схема предусматривает наличие в системе третьего лица — арбитра, пользующегося доверием обеих сторон. Авторизацией документа является сам факт зашифрования его секретным ключом и передача его арбитру.[6]
- На основе алгоритмов асимметричного шифрования. На данный момент такие схемы ЭП наиболее распространены и находят широкое применение. Кроме этого, существуют другие разновидности цифровых подписей (групповая подпись, неоспоримая подпись, доверенная подпись), которые являются модификациями описанных выше схем.[6] Их появление обусловлено разнообразием задач, решаемых с помощью ЭП.

Асимметричные схемы ЭП относятся к криптосистемам с открытым ключом. В отличие от асимметричных алгоритмов шифрования, в которых шифрование производится с помощью открытого ключа, а расшифровка — с помощью закрытого, в асимметричных схемах цифровой подписи подписание производится с применением закрытого ключа, а проверка подписи — с применением открытого. Общепризнанная схема цифровой подписи охватывает три процесса

- Генерация ключевой пары. При помощи алгоритма генерации ключа равновероятным образом из набора возможных закрытых ключей выбирается закрытый ключ, вычисляется соответствующий ему открытый ключ.
- Формирование подписи. Для заданного электронного документа с помощью закрытого ключа вычисляется подпись.
- Проверка (верификация) подписи. Для данных документа и подписи с помощью открытого ключа определяется действительность подписи.

**Использование хэш-функций**

Поскольку подписываемые документы — переменного (и как правило достаточно большого) объёма, в схемах ЭП зачастую подпись ставится не на сам документ, а на его хэш. Для вычисления хэша используются криптографические хэш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функции не являются частью алгоритма ЭП, поэтому в схеме может быть использована любая надёжная хэш-функция.

## 19.Хэш функции. Определение, примеры, уязвимости, применение ?

*Хэш-функцией* называется алгоритм, конвертирующий строку произвольной длины (сообщение) в битовую строку фиксированной длины, называемой *хэш-кодом*, проверочной суммой или цифровым отпечатком.

Криптографической хеш-функцией называется всякая хеш-функция, являющаяся криптостойкой, то есть удовлетворяющая ряду требований специфичных для криптографических приложений. В криптографии хэш-функции применяются для решения следующих задач:
— построения систем контроля целостности данных при их передаче или хранении,
— аутентификация источника данных.

Основным требованием к хэш-функциям является равномерность распределения их значений при случайном выборе значений аргумента. Для криптографических хеш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось. Это называется лавинным эффектом.

К ключевым функциям хэширования предъявляются следующие требования:
— невозможность фабрикации,
— невозможность модификации.

Первое требование означает высокую сложность подбора сообщения с правильным значением свертки.

Второе — высокую сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки.

К бесключевым функциям предъявляют требования:
— однонаправленность,
— устойчивость к коллизиям,
— устойчивость к нахождению второго прообраза.

Под однонаправленностью понимают высокую сложность нахождения сообщения по заданному значению свертки. Следует заметить что на данный момент нет используемых хэш-функций с доказанной однонаправленностью.

Под устойчивостью к коллизиям понимают сложность нахождения пары сообщений с одинаковыми значениями свертки. Обычно именно нахождение способа построения коллизий криптоаналитиками служит первым сигналом устаревания алгоритма и необходимости его скорой замены.

Под устойчивостью к нахождению второго прообраза понимают сложность нахождения второго сообщения с тем же значением свертки для заданного сообщения с известным значением свертки.

Примеры:
Алгоритмы **CRC16/32** — контрольная сумма (не криптографическое преобразование).

Алгоритмы **MD2/4/5/6**. Являются творением Рона Райвеста, одного из авторов алгоритма RSA.
Алгоритм MD5 имел некогда большую популярность, но первые предпосылки взлома появились еще в конце девяностых, и сейчас его популярность стремительно падает.
Алгоритм MD6 — очень интересный с конструктивной точки зрения алгоритм. Он выдвигался на конкурс SHA-3, но, к сожалению, авторы не успели довести его до кондиции, и в списке кандидатов, прошедших во второй раунд этот алгоритм отсутствует.

Алгоритмы линейки **SHA** Широко распространенные сейчас алгоритмы. Идет активный переход от SHA-1 к стандартам версии SHA-2. SHA-2 — собирательное название алгоритмов SHA224, SHA256, SHA384 и SHA512. SHA224 и SHA384 являются по сути аналогами SHA256 и SHA512 соответственно, только после расчета свертки часть информации в ней отбрасывается. Использовать их стоит лишь для обеспечения совместимости с оборудованием старых моделей.

## 20.Пример транспорта сессионного ключа на основе RSA ?

(спиздил у инфы)

(Мб это  поможет)

В криптосистеме с открытым ключом, в отличие от симметричной, используются два ключа: открытый и закрытый (закрытый хранится в секрете).
**Открытый ключ** используется для проверки ЭЦП и для шифрования сообщений.
**Закрытый ключ** – для генерации ЭЦП и для расшифрования сообщений.
**ЭЦП** – (Электронная цифровая подпись) – атрибут электронного документа, который получается в результате некоторого криптографического преобразования данных. **ЭЦП** позволяет проверить целостность документов, конфиденциальность документов, а так же идентифицировать владельца документа. Аналог обычной подписи.

## Криптосистема с открытым ключом

| Ключи | Создание ЭЦП | Шифрование | Проверка ЭЦП | Расшифрование |
|---|---|---|---|---|
|  | Закрытый ключ | Открытый ключ | Открытый ключ | Закрытый ключ |

Стоит отметить, в основе криптографических систем с открытым ключом лежат **односторонние функции** – такие функции, которые обладают следующими свойствами:

1. Пусть известно значение xx, тогда вычислить F(x) относительно просто.
2. Пусть известно y=F(x), однако вычислить xx сложно.

Генерация ключей в RSA осуществляется следующим образом:

1.Выбираются два простых числа p и q (такие что p неравно q).

2.Вычисляется модуль $N=p*q$.

3.Вычисляется значение функции Эйлера от модуля N: $\phi(N)=(p-1)(q-1)$.

4. Выбирается число **e**, называемое открытой экспонентой, число **e** должно лежать в интервале $1<e<\phi(N)$, а так же быть взаимно простым со значением функции $\phi(N)$.

5. Вычисляется число d, называемое секретной экспонентой, такое, что $d*e=1(\mathrm{mod}\phi(N))$, то есть является мультипликативно обратное к числу ее по модулю $\phi(N)$.

Пара (e,N)- открытый ключ.

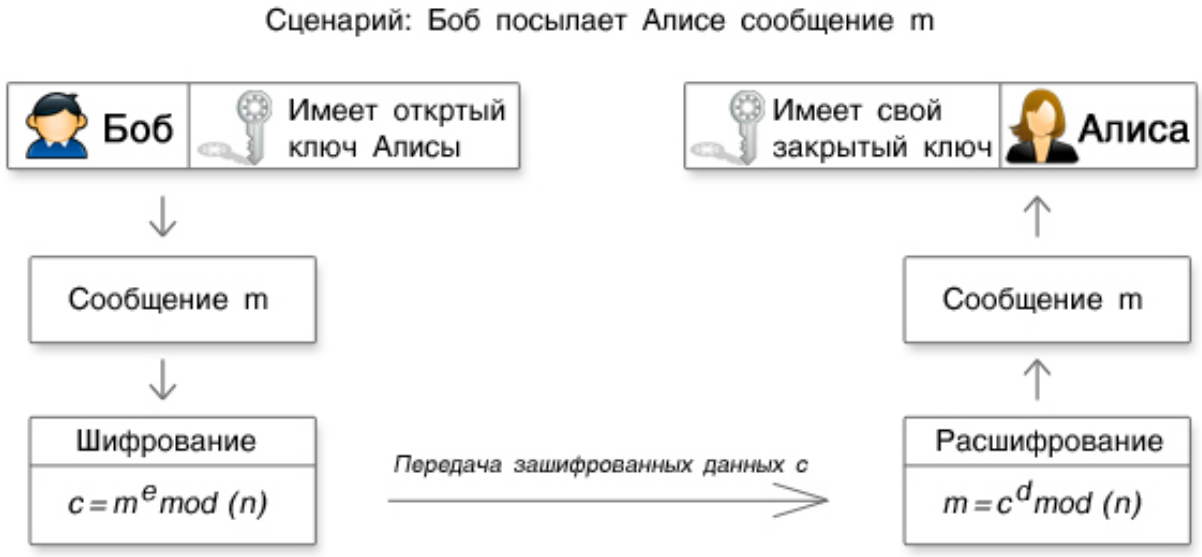
Пара (d,N) - закрытый ключ.

Есть следующий сценарий: Боб и Алиса переписываются в интернете, но хотят использовать шифрование, чтобы поддерживать переписку в секрете :). Алиса заранее сгенерировала закрытый и открытый ключ, а затем отправила открытый ключ Бобу. Боб хочет послать зашифрованное сообщение Алисе:

**Шифрование:** Боб шифрует сообщение m, используя открытый ключ Алисы (e,N): $C=E(M)=M^e \mathrm{mod}(N)$, и отправляет с Алисе.

**Расшифровывание:** Алиса принимает зашифрованное сообщение c. Используя закрытый ключ (d,N), расшифровывает сообщение $M=D(C)=C^d \mathrm{mod}(N)$

Проиллюстрируем то, что описано выше:




## 21.Протокол TLS. Основные фазы ?

TLS — Transport Layer Security, безопасность транспортного уровня.

TLS использует асимметричное шифрование для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности сообщений для сохранения целостности сообщений.

TLS-протокол основан на спецификации протокола SSL версии 3.0, разработанной компанией Netscape.

Данный протокол широко используется в приложениях, работающих с сетью Интернет, таких как веб-браузеры, работа с электронной почтой, обмен мгновенными сообщениями и IP-телефония (VoIP).

TLS даёт возможность клиент-серверным приложениям осуществлять связь в сети таким образом, что нельзя производить прослушивание пакетов и осуществить несанкционированный доступ.

Так как большинство протоколов связи могут быть использованы как с, так и без TLS (или SSL), при установке соединения необходимо явно указать серверу, хочет ли клиент устанавливать TLS. Это может быть достигнуто либо с помощью использования унифицированного номера порта, по которому соединение всегда устанавливается с использованием TLS (как например порт 443 для HTTPS), либо с использованием произвольного порта и специальной команды серверу со стороны клиента на переключение соединения на TLS с использованием специальных механизмов протокола (как например STARTTLS для протоколов электронной почты). Как только клиент и сервер договорились об использовании TLS, им необходимо установить защищённое соединение. Это делается с помощью процедуры подтверждения связи[4][5]. Во время этого процесса клиент и сервер принимают соглашение относительно различных параметров, необходимых для установки безопасного соединения.

**Основные шаги процедуры создания защищённого сеанса связи:**

- клиент подключается к серверу, поддерживающему TLS, и запрашивает защищённое соединение;
- клиент предоставляет список поддерживаемых алгоритмов шифрования и хеш-функций;
- сервер выбирает из списка, предоставленного клиентом, наиболее надёжные алгоритмы среди тех, которые поддерживаются сервером, и сообщает о своём выборе клиенту;
- сервер отправляет клиенту цифровой сертификат для собственной аутентификации. Обычно цифровой сертификат содержит имя сервера, имя удостоверяющего центра сертификации и открытый ключ сервера;
- клиент, до начала передачи данных, проверяет валидность (аутентичность) полученного серверного сертификата, относительно имеющихся у клиента корневых сертификатов удостоверяющих центров (центров сертификации). Клиент также может проверить не отозван ли серверный сертификат, связавшись с сервисом доверенного удостоверяющего центра;
- для шифрования сессии используется сеансовый ключ. Получение общего секретного сеансового ключа клиентом и сервером проводится по протоколу Диффи-Хеллмана. Существует исторический метод передачи сгенерированного клиентом секрета на сервер, при помощи шифрования асимметричной криптосистемой RSA (используется ключ из сертификата сервера). Данный метод не рекомендован, но иногда продолжает встречаться на практике.

На этом заканчивается процедура подтверждения связи. Между клиентом и сервером установлено безопасное соединение, данные, передаваемые по нему, шифруются и расшифровываются с использованием симметричной криптосистемы до тех пор, пока соединение не будет завершено.

При возникновении проблем на некоторых из вышеуказанных шагов подтверждение связи может завершиться с ошибкой, а безопасное соединение не будет установлено.

## 22.Протокол TLS. Уязвимости ?

смотреть 21 вопрос +

TLS имеет множество мер безопасности:

- Защита от понижения версии протокола к предыдущей (менее защищённой) версии или менее надёжному алгоритму шифрования;
- Нумерация последовательных записей приложения и использование порядкового номера в коде аутентификации сообщения (MAC);
- Использование ключа в идентификаторе сообщения (только владелец ключа может сгенерировать код аутентификации сообщения). Алгоритм вычисления кода аутентификации (HMAC), используемый во многих сессиях TLS, определён в RFC 2104;
- Сообщение, которым заканчивается подтверждение связи («Finished»), используется для подтверждения аутентичности ранее переданных сообщений и, таким образом, выбранных параметров TLS-соединения.

**Уязвимость протокола TLS 1.0**, которая считалась теоретической, была продемонстрирована на конференции Ekoparty в сентябре 2011 года.

Демонстрация включала в себя дешифрование cookies, использованных для аутентификации пользователя[6].

**Уязвимость в фазе возобновления соединения**, обнаруженная в августе 2009 года, позволяла криптоаналитику, способному взломать https-соединение, добавлять собственные запросы в сообщения, отправленные от клиента к серверу[7]. Так как криптоаналитик не может дешифровать переписку сервера и клиента, этот тип атаки отличается от стандартной атаки, типа человек посередине. В случае, если пользователь не обращает внимания на индикацию браузера о том, что сессия является безопасной (обычно значок замка), уязвимость может быть использована для атаки типа человек посередине[8]. Для устранения этой уязвимости было предложено как на стороне клиента, так и на стороне сервера добавлять информацию о предыдущем соединении и осуществлять проверку при возобновлении соединения. Это было представлено в стандарте RFC 5746, а также реализовано в последних версиях OpenSSL[9] и других библиотеках[10][11].

Также существуют варианты атак, основанные непосредственно на программной реализации протокола, а не на его алгоритме[12].

## 23.Протокол TLS. Фаза рукопожатия ?

см 21 вопрос +

Протокол Рукопожатия
Криптографические параметры сессии создаются протоколом Рукопожатия, который выполняется выше протокола Записи. Когда клиент и сервер начинают взаимодействовать, они согласовывают версию протокола, выбирают криптографические алгоритмы, могут аутентифицировать друг друга, используя технологию с открытым ключом. Для создания разделяемого секрета также используется технология с открытым ключом.

Протокол Рукопожатия состоит из следующих шагов:

1. Обмен сообщениями Hello для согласования алгоритмов, обмена случайными значениями и проверки возобновляемости сессии.
2. Обмен необходимыми криптографическими параметрами, которые позволяют клиенту и серверу согласовать премастер-секрет.
3. Обмен сертификатами и криптографической информацией, которая позволяет клиенту и серверу аутентифицировать друг друга.
4. Предоставление параметров безопасности на уровень Записи.
5. Возможность клиенту и серверу проверить, что они вычислили одни и те же параметры безопасности и что Рукопожатие произошло без вмешательства злоумышленника.

Протокол разработан для минимизации риска атак "встреча посередине", но защита от атак, при которых злоумышленник может блокировать доступ к порту, не предусмотрена.

Клиент посылает сообщение ClientHello, на которое сервер должен ответить сообщением ServerHello или фатальной ошибкой и разрывом соединения. ClientHello и ServerHello используются для определения максимального уровня безопасности между клиентом и сервером. Client Hello и Server Hello устанавливают следующие атрибуты: Protocol Version, Session ID, Cipher

Suite и Compression Method. Дополнительно создаются и передаются два случайных значения: ClientHello.random и ServerHello.random.

Аутентификация и обмен общим секретом осуществляются в четырех сообщениях: сертификат сервера, обмен ключа сервера, сертификат клиента и обмен ключа клиента. Общий секрет должен быть достаточно большим; текущие методы распределения ключа обмениваются секретами, длина которых находится в диапазоне от 48 до 126 байт.

После сообщений Hello сервер посылает сертификат, с помощью которого клиент выполняет аутентификацию сервера. Дополнительно может быть послано сообщение обмена ключа сервера, если сервер не имеет сертификата или его сертификат может использоваться только для проверки подписи. Если сервер аутентифицирован, он может запросить сертификат клиента, если того требует установленная политика безопасности на стороне сервера. После этого сервер посылает сообщение Server Hello Done, указывающее на то, что фаза Hello-сообщений рукопожатия завершена. Затем сервер ждет ответа клиента. Если сервер послал сообщение запроса сертификата, клиент должен послать сообщение Certificate. После этого посылается сообщение обмена ключа клиента. Содержимое этого сообщения зависит от выбранного алгоритма выработки общего секрета. Если клиент посылал свой сертификат, то он посылает сообщение, содержащее цифровую подпись для проверки всех сообщений Рукопожатия.

В данной точке клиентом посылается сообщение об изменении состояния, и клиент копирует ожидаемое состояние в текущее состояние. После этого клиент посылает заключительное сообщение с использованием новых алгоритмом, ключей и секретов. В ответ сервер посылает свое сообщение об изменении состояния, преобразует ожидаемое состояние в текущее состояние и посылает заключительное сообщение с использованием новых алгоритмов и ключей. После этого рукопожатие считается выполненным, и клиент и сервер могут начинать обмен данными прикладного уровня.

## 24.Протокол TLS. Фаза обмена ключами ?

см вопрос 21 +

В текущей версии протокола доступны следующие алгоритмы:

- Для обмена ключами и проверки их подлинности применяются комбинации алгоритмов: RSA (асимметричный шифр), Diffie-Hellman (безопасный обмен ключами), DSA (алгоритм цифровой подписи), ECDSA;

## 25.Транспорт ключа. Протокол Диффи-Хеллмана ?

Основные сведения
Первая публикация данного алгоритма появилась в 70-х годах XX века в статье Диффи и Хеллмана, в которой вводились основные понятия криптографии с открытым ключом. Алгоритм Диффи-Хеллмана не применяется для шифрования сообщений или формирования электронной подписи. Его назначение – в

распределении ключей. Он позволяет двум или более пользователям обменяться без посредников ключом, который может быть использован затем для симметричного шифрования. Это была первая криптосистема, которая позволяла защищать информацию без использования секретных ключей, передаваемых по защищенным каналам. Схема открытого распределения ключей, предложенная Диффи и Хеллманом, произвела настоящую революцию в мире шифрования, так как снимала основную проблему классической криптографии – проблему распределения ключей.

Алгоритм основан на трудности вычислений дискретных логарифмов. Попробуем разобраться, что это такое. В этом алгоритме, как и во многих других алгоритмах с открытым ключом, вычисления производятся по модулю некоторого большого простого числа P. Вначале специальным образом подбирается некоторое натуральное число A, меньшее P. Если мы хотим зашифровать значение X, то вычисляем

$Y = A^X \bmod P$.

Причем, имея X, вычислить Y легко. Обратная задача вычисления X из Y является достаточно сложной. Экспонента Xкак раз и называется дискретным логарифмом Y. Таким образом, зная о сложности вычисления дискретного логарифма, число Y можно открыто передавать по любому каналу связи, так как при большом модуле P исходное значение Xподобрать будет практически невозможно. На этом математическом факте основан алгоритм Диффи-Хеллмана для формирования ключа.

Формирование общего ключа
Пусть два пользователя, которых условно назовем пользователь 1 и пользователь 2, желают сформировать общий ключ для алгоритма симметричного шифрования. Вначале они должны выбрать большое простое число P и некоторое специальное число A, 1 < A < P–1, такое, что все числа из интервала [1, 2, ..., P–1] могут быть представлены как различные степени A mod P. Эти числа должны быть известны всем абонентам системы и могут выбираться открыто. Это будут так называемые общие параметры.

Затем первый пользователь выбирает число $X_1$ ($X_1$<P), которое желательно формировать с помощью датчика случайных чисел. Это будет закрытый ключ первого пользователя, и он должен держаться в секрете. На основе закрытого ключа пользователь 1 вычисляет число

$$Y_1 = A^{X_1} \, mod \, P$$

которое он посылает второму абоненту.

Аналогично поступает и второй пользователь, генерируя $X_2$ и вычисляя

$$Y_2 = A^{X_2} \, mod \, P$$

Это значение пользователь 2 отправляет первому пользователю.

После этого у пользователей должна быть информация, указанная в следующей таблице:

| | Общие параметры | Открытый ключ | Закрытый ключ |
|---|---|---|---|
| Пользователь 1 | P, A | $Y_1$ | $X_1$ |
| Пользователь 2 | | $Y_2$ | $X_2$ |

Из чисел $Y_1$ и $Y_2$, а также своих закрытых ключей каждый из абонентов может сформировать общий секретный ключ $Z$ для сеанса симметричного шифрования. Вот как это должен сделать первый пользователь:

$$Z = (Y_2)^{X_1} \, mod \, P$$

Никто другой кроме пользователя 1 этого сделать не может, так как число $X_1$ секретно. Второй пользователь может получить то же самое число $Z$, используя свой закрытый ключ и открытый ключ своего абонента следующим образом:

$$Z = (Y_1)^{X_2} \, mod \, P$$

Если весь протокол формирования общего секретного ключа выполнен верно, значения $Z$ у одного и второго абонента должны получиться одинаковыми. Причем, что самое важное, противник, не зная секретных чисел $X_1$ и $X_2$, не сможет вычислить число $Z$. Не зная $X_1$ и $X_2$, злоумышленник может попытаться вычислить $Z$, используя только передаваемые открыто $P$, $A$, $Y_1$ и $Y_2$. Безопасность формирования общего ключа в алгоритме Диффи-Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, очень трудно вычислить дискретные логарифмы. Для больших простых чисел размером сотни и тысячи бит задача считается неразрешимой, так как требует колоссальных затрат вычислительных ресурсов.

Пользователи 1 и 2 могут использовать значение $Z$ в качестве секретного ключа для шифрования и расшифрования данных. Таким же образом любая пара абонентов может вычислить секретный ключ, известный только им.

## 26. Транспорт ключа. Разделенное хранение ключа ?

## 27.Транспорт ключа. Протокол Нидхема-Шредера ?

**Протокол Нидхема — Шрёдера** — общее название для симметричного и асимметричного протоколов аутентификации и обмена ключами. Оба протокола были предложены Майклом Шрёдером (англ.)русск. и Роджером Нидхемом[1]. Вариант, основанный на симметричном шифровании, использует промежуточную доверенную сторону. Этот протокол стал основой для целого класса подобных протоколов. Например, Kerberos является одним из вариантов симметричного протокола Нидхема — Шрёдера. Вариант, основанный на асимметричном шифровании, предназначен для взаимной аутентификации сторон. В оригинальном виде оба варианта протокола являются уязвимыми.

При схеме шифрования с симметричным ключом, предполагается, что секретный ключ известен и серверуаутентификации  (Трент) и обоим субъектам обмена:  (Алиса) и  (Боб). Изначально оба субъекта имеют секретные ключи:  и , известные только им и некоторой доверенной стороне — серверу аутентификации. В ходе выполнения протокола Алиса и Боб получают от сервера новый секретный сессионный ключ для шифрования взаимных сообщений в данном сеансе связи, то есть сообщения от Алисы к Бобу расшифровать может только Боб, сообщения от Боба к Алисе расшифровать может только Алиса. Кроме того субъекты обмена должны быть уверены, что пришедшее сообщение было отправлено именно тем, с кем должен произойти обмен. Боб должен быть уверен, что получил сообщение именно от Алисы и наоборот. Это также обеспечивается протоколом.

реализацию протокола[4]:

1. $$Alice \to A, B, R_A \to Trent$$
2. $$Trent \to \left\{ R_A, B, K, \{K, A\}_{K_B} \right\}_{K_A} \to Alice$$
3. $$Alice \to \{K, A\}_{K_B} \to Bob$$
4. $$Bob \to \{R_B\}_K \to Alice$$
5. $$Alice \to \{R_B - 1\}_K \to Bob$$

Обмен начинается с того, что Алиса генерирует некоторое случайное число (идентификатор), использующееся один раз. Первое сообщение от Алисы к Тренту содержит в себе имена участников предстоящего обмена и генерированное Алисой случайное число:

$$Alice \to A, B, R_A \to Trent$$

Данное сообщение посылается открытым текстом, но может быть зашифровано ключом Алисы :

$$Alice \to \{A, B, R_A\}_{K_A} \to Trent$$

При получении этого сообщения Трент извлекает из базы данных секретные ключи Алисы и Боба:  и , а также вычисляет новый сессионный ключ . Далее Трент посылает Алисе следующее сообщение:

$$Trent \to \left\{R_A, B, K, \{K, A\}_{K_B}\right\}_{K_A} \to Alice$$

Алиса может расшифровать и прочесть сообщение от Трента. Она проверяет наличие своего идентификатора  в сообщении, что подтверждает то, что данное сообщение является откликом на её первое сообщение Тренту. Также она проверяет имя субъекта, с которым собирается обмениваться данными. Эта проверка обязательна, так как если бы не было этого имени, Злоумышленник мог бы заменить имя Боба на своё в первом сообщении, и Алиса, ничего не подозревая, в дальнейшем бы взаимодействовала со Злоумышленником. Часть сообщения Алиса прочитать не может, так как эта часть зашифрована ключом Боба. Алиса пересылает Бобу зашифрованный его ключом фрагмент:

$$Alice \to \{K, A\}_{K_B} \to Bob$$

Расшифровать его может только Боб, так как оно зашифровано его секретным ключом. После расшифровки Боб тоже владеет сессионным ключом . Имя Алисы в сообщении подтверждает факт, что сообщение от неё. Далее при обмене данными будет использоваться сессионный ключ. Чтобы сделать схему симметричной и уменьшить вероятность атаки воспроизведения, Боб генерирует некоторое случайное число  (идентификатор Боба) и посылает Алисе следующее сообщение, зашифрованное сессионным ключом:

$$Bob \to \{R_B\}_K \to Alice$$

Алиса расшифрует его и посылает отклик, который ожидает Боб, также зашифрованный сессионным ключом:

$$Alice \to \{R_B - 1\}_K \to Bob$$

## 28.Определение авторизации. Базовые модели ?

**Авториза́ция** — предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

В информационных технологиях посредством авторизации устанавливаются права доступа к информационным ресурсам и системам обработки данных. Для этого применяются различные виды авторизации, которые можно поделить **на три класса:**

**Дискреционное управление доступом**

В случае дискреционного (избирательного) управления (DAC), доступ к объектам, данным или функциям, предоставляется явно указанным субъектам, пользователям или группам пользователей. Например, пользователю user_1 разрешено читать файл file_1, но запрещено в него писать. Каждый объект имеет привязанного к нему субъекта — владельца, который и устанавливает права доступа к объекту. Также система имеет одного выделенного субъекта — суперпользователя, имеющего право устанавливать права доступа для всех субъектов. А любой субъект может передавать имеющиеся у него права другим субъектам. Такой доступ используется в современных операционных системах, где для авторизации наиболее распространено использование полномочий и списков контроля доступа (ACL).[1]

**Мандатное управление доступом**

Мандатный доступ (MAC) заключается в разделении информации по степени секретности, а пользователей по уровням допуска к этой информации. Главное преимущество мандатного доступа заключается в ограничении прав владельца объекта. Права субъектов на создаваемые ими объекты будут зависеть от их уровня допуска, соответственно они не смогут случайно или преднамеренно делегировать их неавторизированным пользователям. Согласно требованиям ФСТЭК мандатное управление доступом является ключевым отличием систем защиты Государственной Тайны РФ старших классов 1В и 1Б от младших классов защитных систем, основанных на дискреционной модели. Поддержка мандатного управления доступом присутствует в некоторых операционных системах, таких как Ubuntu, SUSE Linux, FreeBSD. Также используется в системах управления базами данных. Иногда применяется вместе с дискреционным контролем доступа.

**Управление доступом на основе ролей**

Развитием политики избирательного доступа является управление доступом на основе ролей (RBAC), где доступ к объектам системы формируется с учётом специфики их применения на основе роли субъектов в каждый момент времени. Роли позволяют определить понятные для пользователей правила разграничения доступа. Роль сочетает свойства избирательного управления доступом, ставя в соответствие субъектам объекты, и мандатного, при изменении ролей изменится и доступ к группе файлов, но этот тип доступа более гибкий, по сравнению с предыдущими, и может их моделировать. Сейчас RBAC широко используется для управления пользовательскими привилегиями в пределах единой системы или приложения. Список таких систем включает в себя Microsoft Active Directory, SELinux, FreeBSD, Solaris, СУБД Oracle, PostgreSQL 8.1, SAP R/3, Lotus Notes и множество других.

**Другие типы управления доступом**

- Контроль доступа на основе контекста (CBAC)
- Контроль доступа на основе решетки (LBAC)

## 29.Основные методы контроля доступа ?

ответ выше в 28 вопросе.

## 30.Контроль доступа на основе генерации ключей вручную ?

Как бы ни была сложна и надежна сама криптосистема, она основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в системе, где количество пользователей составляет десятки и сотни управление ключами, – это серьезная проблема.
Под ключевой информацией понимается совокупность всех действующих в системе ключей. Если не обеспечено достаточно надежное управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации.
Управление ключами – информационный процесс, включающий в себя три элемента:

- генерацию ключей;
- накопление ключей;
- распределение ключей.

**Генерация ключей.** В реальных системах используются специальные аппаратные и программные методы генерации случайных ключей. Как правило используют датчики случайных чисел. Однако степень случайности их генерации должна быть достаточно высокой. Идеальными генераторами являются устройства на основе “натуральных” случайных процессов. Например, генерация ключей на основе белого радиошума. Другим случайным математическим объектом являются десятичные знаки иррациональных чисел, например [?] или e, которые вычисляются с помощью стандартных математических методов.
В системах со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют случайные числа как сложную функцию от текущего времени и (или) числа, введенного пользователем.

## 31.Контроль доступа на основе иерархической модели и функций хэширования ?

## 32.Контроль доступа на основе шифрования с атрибутами ?

**Attribute-based Encryption** (рус. *Шифрование на основе атрибутов, ABE-шифрование*) — разновидность алгоритмов шифрования с открытым ключом, в которых закрытый ключ, применяемый пользователем для расшифрования данных, зависит от некоторых атрибутов пользователя (например, должность, место жительства, тип учётной записи).

Схема была предложена Sahai и Waters в 2005 году. В ней участвуют владелец данных (передающий данные), пользователь (принимающий данные) и третья сторона (доверенный центр), чья роль заключается в том, чтобы генерировать ключи для шифрования и расширования данных владельцами данных и пользователями.

Ключи (в частности, открытый ключ и универсальный ключ) генерируются по установленному полному набору атрибутов. Если к системе присоединится пользователь с новым атрибутом, атрибут будет добавлен к набору, а открытый и универсальный ключи будут сгенерированы заново.

Владелец данных шифрует данные при помощи открытого ключа и некоторого набора атрибутов.

Пользователь может расшифровать данные при помощи собственного закрытого ключа, который он получает от доверенного центра. Проверяется соответствие между атрибутами, которые составляют закрытый ключ пользователя, и атрибутами зашифрованных данных. Если число совпадающих атрибутов превышает установленный порог , закрытый ключ пользователя сможет расшифровать данные. (Пусть, например, атрибуты шифрованных данных {"Кафедра радиотехники", "Преподаватель", "Студент"}, . Чтобы пользователь смог расшифровать данные и получить к ним доступ, в наборе атрибутов, составляющих его закрытый ключ, должны присутствовать по крайней мере два из трёх атрибутов шифрованных данных).

## 33.Протокол OAuth2.0 Определение и основные типы потоков (Flow) ?

ЭТО пример версии 1 (ниже 2 версия, первая нужна для пояснения 2)

## OAuth 1.0

Поясним работу протокола OAuth 1.0 на примере[5]. Допустим, что пользователь (владелец ресурсов) хочет распечатать свои фотографии (ресурсы), загруженные на сайт «photos.example.net» (сервер), при помощи сервиса печати «printer.example.net» (клиент).

1. Клиент посредством протокола HTTPS отправляет серверу запрос, который содержит идентификатор клиента, метку времени, адрес обратного вызова (по нему нужно вернуть токен), используемый тип цифровой подписи и саму подпись.
2. Сервер подтверждает запрос и отвечает клиенту токеном запроса (англ. *Request Token*) и частью разделённого секрета.
3. Клиент передаёт токен владельцу ресурсов (пользователю) и перенаправляет его на сервер для прохождения аутентификации.
4. Сервер, получив от пользователя токен, запрашивает у него его логин и пароль, и в случае успешной аутентификации просит пользователя подтвердить доступ клиента к ресурсам (происходит авторизация), после чего пользователь перенаправляется сервером к клиенту.
5. Клиент передаёт серверу токен запроса посредством протокола TLS и запрашивает доступ к ресурсам.
6. Сервер подтверждает запрос и отвечает клиенту новым токеном доступа (англ. *Access Token*).
7. Используя токен доступа, клиент обращается к серверу за ресурсами.
8. Сервер подтверждает запрос и предоставляет ресурсы.

**OAuth 2.0** **(с вики, а ниже есть и с хабры)**

**OAuth** — открытый протокол авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей (третьей стороне) логин и пароль

Протокол OAuth 2.0 обратно не совместим с протоколом OAuth 1.0[1]. Вместо того, чтобы дополнить OAuth 1.0, было принято решение разработать другой протокол[16]. Поэтому принцип работы OAuth 1.0 и OAuth 2.0 отличается. Так в стандарте OAuth 2.0 описаны следующие потоки (сценарии взаимодействия сторон):

- Поток неявного доступа (англ. *Implicit Grant Flow*)

Является самым коротким потоком протокола: пользователь сначала перенаправляется клиентом на сервер, чтобы разрешить доступ к ресурсам, а после того, как доступ будет получен, перенаправляется сервером обратно к клиенту[16].

- Поток с кодом подтверждения (англ. *Authorization Code Flow*)

Отличие данного потока от потока неявного доступа заключается в дополнительном шаге аутентификации клиента[16].

- Поток с обновляемым токеном (англ. Refreshing an Expired Access Token Flow)

Отличия этого потока от приведённого примера в следующем: на шаге 2 сервер помимо токена доступа, который имеет ограниченное время жизни, выдает токен обновления; на шаге 8 сервер проверяет, является ли токен доступа валидным (в смысле истечения времени жизни), и в зависимости от этого либо предоставляет доступ к ресурсам, либо требует обновления токена доступа (который предоставляется при предъявлении токена обновления).

- Поток с предоставлением клиенту пароля (англ. *Resource Owner Password Credentials Flow*)

В этом потоке владелец ресурсов предоставляет клиенту логин и пароль, он передает их серверу и получает токен для доступа к ресурсам. Несмотря на то, что такой режим работы несколько противоречит концепции создания протокола, он описан в спецификации.

- Поток клиентских полномочий (англ. *Client Credentials Flow*)

В данном режиме работы протокола предоставление сервером токена доступа происходит после передачи клиентом его пользователя и пароля, который был предварительно установлен сервером авторизации (в спецификации не оговорено, каким именно образом). Фактически, клиент сразу проходит как авторизацию, так и аутентификацию.

OAuth поддерживает два метода аутентификации сообщений от клиента: HMAC-SHA1 и RSA-SHA1. Есть возможность передавать сообщения без подписи, тогда в поле типа подписи указывается «plain text». Но в этом случае, согласно спецификации, соединение между клиентом и сервером должно устанавливаться через протокол SSL или TLS[5].

**Что такое OAuth 2.0 с хабры**

OAuth 2.0 — протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе. Протокол избавляет от необходимости доверять приложению логин и пароль, а также позволяет выдавать ограниченный набор прав, а не все сразу.

**Чем отличаются OpenID и OAuth**

OpenID предназначен для **аутентификации** — то есть для того, чтобы понять, что этот конкретный пользователь является тем, кем представляется. Например, с помощью OpenID некий сервис Ололо может понять, что зашедший туда пользователь, это именно Рома Новиков с Mail.Ru. При следующей аутентификации Ололо сможет его опять узнать и понять, что, это тот же Рома, что и в прошлый раз.

OAuth же является протоколом **авторизации**, то есть позволяет выдать права на действия, которые сам Ололо сможет производить в Mail.Ru от лица Ромы. При этом Рома после авторизации может вообще не участвовать в процессе выполнения действий, например, Ололо сможет самостоятельно заливать фотографии на Ромин аккаунт.

**Как работает OAuth 2.0**

Как и первая версия, OAuth 2.0 основан на использовании базовых веб-технологий: HTTP-запросах, редиректах и т. п. Поэтому использование OAuth возможно на любой платформе с доступом к интернету и браузеру: на сайтах, в мобильных и desktop-приложениях, плагинах для браузеров…

Ключевое отличие от OAuth 1.0 — простота. В новой версии нет громоздких схем подписи, сокращено количество запросов, необходимых для авторизации.

Общая схема работы приложения, использующего OAuth, такова:

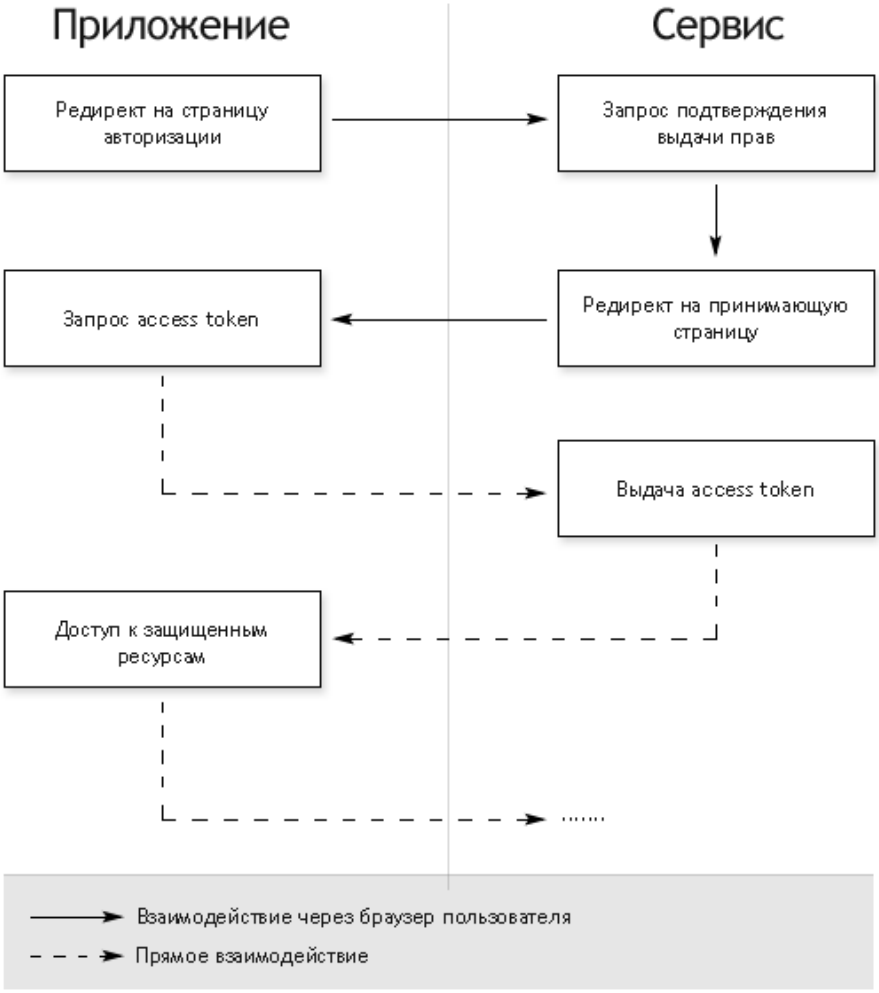
1. получение авторизации
2. обращение к защищенным ресурсам

Результатом авторизации является ***access token*** — некий ключ (обычно просто набор символов), предъявление которого является пропуском к защищенным ресурсам. Обращение к ним в самом простом случае происходит по HTTPS с указанием в заголовках или в качестве одного из параметров полученного *access token*'а.

**В протоколе описано несколько вариантов авторизации, подходящих для различных ситуаций:**

- авторизация для приложений, имеющих серверную часть (чаще всего, это сайты и веб-приложения)
- авторизация для полностью клиентских приложений (мобильные и desktop-приложения)
- авторизация по логину и паролю
- восстановление предыдущей авторизации

**Авторизация для приложений, имеющих серверную часть**




1. Редирект на страницу авторизации
2. На странице авторизации у пользователя запрашивается подтверждение выдачи прав
3. В случае согласия пользователя, браузер редиректится на URL, указанный при открытии страницы авторизации, с добавлением в GET-параметры специального ключа — *authorization code*
4. Сервер приложения выполняет POST-запрос с полученным *authorization code* в качестве параметра. В результате этого запроса возвращается *access token*

Это самый сложный вариант авторизации, но только он позволяет сервису однозначно установить приложение, обращающееся за авторизацией (это происходит при коммуникации между серверами на последнем шаге). Во всех остальных вариантах авторизация происходит полностью на клиенте и по понятным причинам возможна маскировка одного приложения под другое. Это стоит учитывать при внедрении OAuth-аутентификации в API сервисов.

## 34.Протокол OAuth2.0 Authorization code flow ?

в 33 вопросе, выделено желтым (если правильно понял)

## 35.Протокол OAuth2.0  Access token flow ?

в 33 вопросе, выделено желтым (если правильно понял)

## 36.Протокол OAuth2.0 в Facebook ?

Процесс получение данных из социальной сети происходит обычно с использование открытого протокола авторизации OAuth 2.0, который позволяет без необходимости предоставления третьей стороне (нашему сайту) логина и пароля, получать ограниченный доступ к пользовательским ресурсам. Происходит это следующим образом:


1. Сайт перенаправляет пользователя на специальную страницу в социальной сети, указывая в параметрах права на какую информацию необходимы
2. На этой странице пользователь авторизовывается, если необходимо.
3. После авторизации пользователю разъясняется права запрашивемые сайтом, а также спрашивается его разрешение
4. В случае согласия, пользователь перенаправляется на сайт с специальным идентификатором, с помощью которого сайт может получить авторизационный токен
5. С помощью этого токена и делаются запросы к api социальной сети
6. При повторной авторизации этап подтверждения прав пропускается за ненадобностью

В социальной сети необходимо создать приложение, идентификатор и секретный ключ которого будут использоваться при запросе токена в качестве параметров client_id и client_secret.

Использование OAuth 2.0 протокола удобно, но в реальном мире есть проблемы. Все социальные сети реализуют данный протокол по разному. Общая концепция сохраняется, но существуют различия, из-за которых приходится писать довольно много кода.

Используется библиотека, позволяющая производить авторизацию в социальных сетях без сложностей. С помощью нее разработчику будет проще разобраться с данной темой. Использовать библиотеку просто.
Первым делом формируем ссылку и перенаправляем пользователя в социальную сеть.

```php
//auth.php
  //$REDIRECT_URL_VK = 'http://site.tld/
auth_callback.php';
  $auth = new \Social\Auth\AuthVk($APP_ID_VK,
$APP_SECRET_VK, $APP_SCOPE_VK);
  $url = $auth->getAuthorizeUrl($REDIRECT_URL_VK);
  stopAndRedirect($url);
```

После подтверждения прав пользователь перенаправляется на redirect_url, в котором сайт запрашивает токен и производит первый запрос к api социальной сети, получая информацию о пользователе.

```php
//auth_callback.php
  $auth = new \Social\Auth\AuthVk($APP_ID_VK,
$APP_SECRET_VK, $APP_SCOPE_VK);
  $token = $auth->authenticate($_REQUEST,
$REDIRECT_URL_VK);

  //call api with access_token
  $api = new \Social\Api\ApiVk($token);
  $user = $api->getProfile();

  // use or save user object
  // $user->id
  // $user->firstName
  // $user->lastName
  // $user->nickname
  // $user->screenName
  // $user->photoUrl
  // $user->photoBigUrl
  // ...
```

Как видно все просто. Библиотека берет на себя все сложности.

## 37.Уязвимости протокола OAuth2.0 ?

хз нашел это

OAuth 2.0 — развивающийся стандарт. Это значит, что спецификация еще не устоялась и постоянно меняется, иногда довольно заметно.

Безопасность OAuth 2.0 во многом основана на SSL. Это сильно упрощает жизнь разработчикам, но требует дополнительных вычислительных ресурсов и администрирования. Это может быть существенным вопросом в высоко нагруженных проектах.

Как показало исследование, OAuth 2.0 не определяет ни требования по безопасности, ни защиту взаимодействия бэкенда и сторонних приложений. Это породило ряд кастомизированных расширений API для обеспечения единого входа. «К сожалению, безопасность таких самопальных адаптаций/API и рабочие требования обычно воспринимаются как нечто само собой разумеющееся; в документации они зачастую отражены неявным образом либо вовсе непонятны сторонним разработчикам мобильных приложений, — сетуют исследователи. — Более того, общие для сторонних разработчиков правила использования API системы единого входа до сих пор не сформулированы».

Атака, разработанная в стенах университета Гонконга, не требует взаимодействия с пользователем или его устройством. Инициатору атаки нужно лишь создать для своего устройства прокси в использующем SSL домене, который он контролирует. Этот прокси, позволяющий мониторить и изменять входящий и исходящий трафик на устройстве атакующего, должен занимать промежуточное положение между сервером сторонней программы и сервером провайдера услуг аутентификации (IdP). Установив уязвимое приложение, атакующий осуществляет вход по OAuth 2.0 под своей учетной записью и в ходе OAuth-обмена заменяет свой ID идентификатором жертвы (раздобытым в открытых источниках).

«Поскольку бэкенд-сервер стороннего приложения идентифицирует пользователя, используя доказательства, предоставленные клиентским приложением, и дополнительных проверок не производит, атакующий сможет успешно осуществить вход в приложение от имени жертвы и в большинстве случаев получит полный доступ к ее конфиденциальной информации, размещенной на бэкенд-сервере этого приложения, — пишут далее исследователи. — Основной причиной данной уязвимости является общее, но неоправданное доверие к данным аутентификации, которые бэкенд-сервер получает от своего мобильного клиента, а тот, в свою очередь, полагается на полученную от клиента IdP информацию, которая потенциально может быть модифицирована».

Facebook несколько уменьшила вероятность таких атак благодаря поддержке закрепления сертификатов, введенной в мае 2014 года, и не принимает измененные сообщения. Однако эта мера защиты не всем доступна: нынешнее исследование показало, что для обеспечения обратной совместимости Facebook идентифицирует тех ранних пользователей мобильного приложения, которым был присвоен ID общего доступа. Таким образом, если пользователь Facebook зарегистрировался в приложении через OAuth до мая 2014 года, он открыт для MitM-атак с подменой ID.

## 38.Атаки типа XSS. Определение, методы защиты ?

**XSS** (англ. *Cross-Site Scripting* — «межсайтовый скриптинг») — тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника. Является разновидностью атаки «внедрение кода[en]».

Специфика подобных атак заключается в том, что вредоносный код может использовать авторизацию пользователя в веб-системе для получения к ней расширенного доступа или для получения авторизационных данных пользователя. Вредоносный код может быть вставлен в страницу как через уязвимость в веб-сервере, так и через уязвимость на компьютере пользователя[1].

Для термина используют сокращение «XSS», чтобы не было путаницы с каскадными таблицами стилей, использующими сокращение «CSS».

**Средства защиты**

**Защита на стороне сервера**

- Кодирование управляющих HTML-символов, JavaScript, CSS и URL перед отображением в браузере. Для фильтрации входных параметров можно использовать следующие функции: `filter_sanitize_encoded` (для кодирования URL)[27], `htmlentities` (для фильтрации HTML)[28].
- Кодирование входных данных. Например с помощью библиотек OWASP Encoding Project[29], HTML Purifier, htmLawed, Anti-XSS Class.
- Регулярный ручной и автоматизированный анализ безопасности кода и тестирование на проникновение. С использованием таких инструментов, как Nessus, Nikto Web Scanner и OWASP Zed Attack Proxy.
- Указание кодировки на каждой web-странице (например, ISO-8859-1 или UTF-8) до каких-либо пользовательских полей[30].
- Обеспечение безопасности cookies, которая может быть реализована путём ограничения домена и пути для принимаемых cookies, установки параметра HttpOnly[31], использованием SSL[32].
- Использование заголовка Content Security Policy, позволяющего задавать список, в который заносятся желательные источники, с которых можно подгружать различные данные, например, JS, CSS, изображения и пр.

**Защита на стороне клиента**

- Регулярное обновление браузера до новой версии[18].
- Установка расширений для браузера, которые будут проверять поля форм, URL, JavaScript и POST-запросы, и, если встречаются скрипты, применять XSS-фильтры для предотвращения их запуска. Примеры подобных расширений: NoScript для FireFox, NotScripts для Chrome и Opera.

## 39.Атаки типа CSRF. Определение, методы защиты ?

**CSRF** (англ. *Cross Site Request Forgery* — «Межсайтовая подделка запроса», также известен как XSRF) — вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер (например, на сервер платёжной системы), осуществляющий некую вредоносную операцию (например, перевод денег на счёт злоумышленника). Для осуществления данной атаки жертва должна быть аутентифицирована на том сервере, на который отправляется запрос, и этот запрос не должен требовать какого-либо подтверждения со стороны пользователя, которое не может быть проигнорировано или подделано атакующим скриптом.

Основное применение CSRF — вынуждение выполнения каких-либо действий на уязвимом сайте от лица жертвы (изменение пароля, секретного вопроса для восстановления пароля, почты, добавление администратора и т. д.). Также с помощью CSRF возможна эксплуатация отраженных XSS, обнаруженных на другом сервере.

**Защита**

Защищаться должны все запросы изменяющие данные на сервере, а также возвращающие персональные либо иные деликатные данные.

Наиболее простым для понимания способом защиты от данного типа атак является механизм, когда веб-сайты должны требовать подтверждения большинства действий пользователя и проверять поле HTTP_REFERER, если оно указано в запросе. Но этот способ может быть небезопасен, и использовать его не рекомендуется[2].

Другим распространённым способом защиты является механизм, при котором с каждой сессией пользователя ассоциируется дополнительный секретный уникальный ключ, предназначенный для выполнения запросов. Секретный ключ не должен передаваться в открытом виде, например если это POST запрос, то ключ следует передавать в теле запроса, а не в адресе страницы. Пользователь посылает этот ключ среди параметров каждого запроса, и перед выполнением каких-либо действий сервер проверяет этот ключ. Преимуществом данного механизма, по сравнению с проверкой Referer, является гарантированная защита от атак данного типа. Недостатком же являются требование возможности организации пользовательских сессий и требование динамической генерации HTML-кода активных страниц сайта.

Спецификация протокола HTTP/1.1 [3] определяет безопасные методы запросов, такие как GET, HEAD, которые не должны изменять данные на сервере. Для таких запросов, при соответствии сервера спецификации, нет необходимости применять защиту CSRF.

Может возникнуть желание подстраховаться и добавить ключ в каждый запрос, но следует иметь в виду, что спецификация HTTP/1.1 [3] допускает наличие тела для любых запросов, но для некоторых методов запроса (GET, HEAD, DELETE) семантика тела запроса не определена, и должна быть проигнорирована. Поэтому ключ может быть передан только в самом URL, или в HTTP заголовке запроса. Необходимо защитить самого пользователя от неблагоразумного распространения ключа, в составе URL, например на форуме, где он может оказаться доступным злоумышленнику. Поэтому запросы, с ключом в URL, не следует использовать в качестве адреса для перехода, то есть исключить переход по такому адресу клиентским скриптом, перенаправлением сервера, действием формы, гиперссылкой на странице и т.п. с целью сокрытия ключа, входящего в URL. Их можно использовать лишь как внутренние запросы скриптом с использованием XMLHttpRequest или обёрткой, например AJAX.

Существенен факт того, что CSRF токен может быть предназначен не для конкретного запроса или формы, а для всех запросов пользователя вообще. Поэтому достаточно утечки CSRF токена, с URL выполняющим простое действие или не выполняющего действие вовсе, как защиты от подделки запроса лишается любое действие, а не только то с которым связан ставший известным URL.

Существует более жёсткий вариант предыдущего механизма, в котором с каждым действием ассоциируется уникальный одноразовый ключ. Такой способ более сложен в реализации и требователен к ресурсам. Способ используется некоторыми сайтами и порталами, такими как Livejournal, Rambler и др. В настоящий момент (2016 г.) нет сведений о преимуществе более жёсткого варианта, по сравнению с вариантом, где используется единственный для каждой сессии секретный ключ[4].

## 40.Атаки типа Clickjacking. Определение, методы защиты ?

**Кликджекинг** (англ. *Clickjacking*) — механизм обмана пользователей интернета, при котором злоумышленник может получить доступ к конфиденциальной информации или даже получить доступ к компьютеру пользователя, заманив его на внешне безобидную страницу или внедрив вредоносный код на безопасную страницу. Принцип основан на том, что поверх видимой страницы располагается невидимый слой, в который и загружается нужная злоумышленнику страница, при этом элемент управления (кнопка, ссылка), необходимый для осуществления требуемого действия, совмещается с видимой ссылкой или кнопкой, нажатие на которую ожидается от пользователя. Возможны различные применения технологии — от подписки на ресурс в социальной сети до кражи конфиденциальной информации и совершения покупок в интернет-магазинах за чужой счёт..

Кликджекингу подверглись в своё время Twitter, Facebook , PayPal, YouTube и многие другие сайты. Сейчас, конечно, они уже защищены.

Вот как выглядел «угон клика» пользователя, который зарегистрирован на Facebook:

1. На вредоносной странице пользователю подсовывается безобидная ссылка (скажем, что-то скачать, «разбогатеть сейчас», посмотреть ролик или просто перейти по ссылке на интересный ресурс).
2. Поверх этой заманчивой ссылки помещен прозрачный iframe со страницей facebook.com, так что кнопка «Like» находится чётко над ней.
3. Кликая на ссылку, посетитель на самом деле нажимает на эту кнопку.

Плохая защита:

> Самый старый метод защиты – это код JavaScript, не позволяющий отобразить веб-страницу внутри фрейма («framebusting», также его называют «framekilling» и «framebreaking»).

Другой способ защиты (там ещё вроде что-то есть… но этот самый нормальный):

Заголовок X-Frame-Options
Все современные браузеры поддерживают заголовок `X-Frame-Options`.

Он разрешает или запрещает отображение страницы, если она открыта во фрейме.

Браузеры игнорируют заголовок, если он определен в META теге. Таким образом, `<meta http-equiv="X-Frame-Options"...>` будет проигнорирован.

**У заголовка может быть три значения:**

**SAMEORIGIN**
Рендеринг документа, при открытии во фрейме, производится только в том случае, когда верхний (top) документ – с того же домена.

**DENY**
Рендеринг документа внутри фрейма запрещён.

**ALLOW-FROM domain**
Разрешает рендеринг, если внешний документ с данного домена (не поддерживается в Safari, Firefox).

К примеру, Twitter использует `X-Frame-Options: SAMEORIGIN`

Атака особенно опасна, поскольку, проектируя интерфейс сайта, обычно никто и не задумывается о том, что клик от имени юзера может сделать хакер. Точки уязвимости могут быть в совершенно непредсказуемых местах.

1) Рекомендуется использовать `X-Frame-Options` на страницах, заведомо не предназначеных для запуска во фрейме и на важнейших страницах (финансовые транзакции).

2) Используйте перекрывающий `<div>`, если это допустимо вашим проектом и вы хотите разрешить безопасный показ документа во фреймах с любых доменов.

## 41.Атаки типа SQL injection. Определение, методы защиты ?

Атака типа SQL Injection — это атака, при которой производится вставка вредоносного кода в строки, передающиеся затем в экземпляр SQL Server для синтаксического анализа и выполнения. Любая процедура, создающая инструкции SQL, должна рассматриваться на предмет уязвимости к вставке небезопасного кода, поскольку SQL Server выполняет все получаемые синтаксически правильные запросы. Даже параметризованные данные могут стать предметом манипуляций опытного злоумышленника.

Основная форма атаки SQL Injection состоит в прямой вставке кода в пользовательские входные переменные, которые объединяются с командами SQL и выполняются. Менее явная атака внедряет небезопасный код в строки, предназначенные для хранения в таблице или в виде метаданных. Когда впоследствии сохраненные строки объединяются с динамической командой SQL, происходит выполнение небезопасного кода.

Атака осуществляется посредством преждевременного завершения текстовой строки и присоединения к ней новой команды. Поскольку к вставленной команде перед выполнением могут быть добавлены дополнительные строки, злоумышленник заканчивает внедряемую строку меткой комментария «--». Весь последующий текст во время выполнения не учитывается.

**Для защиты** от данного типа атак необходимо тщательно фильтровать входные параметры, значения которых будут использованы для построения SQL-запроса:

**1) Фильтрация строковых параметров**

Чтобы внедрение кода (закрытие строки, начинающейся с кавычки, другой кавычкой до её завершения текущей закрывающей кавычкой для разделения запроса на две части) было невозможно, для некоторых СУБД, в том числе, для MySQL, требуется брать в кавычки все строковые параметры. В само́м параметре заменяют кавычки на \", апостроф на \', обратную косую черту на \\ (это называется «экранировать спецсимволы»).

### 2) Фильтрация целочисленных параметров

В данном случае поле id имеет числовой тип, и его чаще всего не берут в кавычки. Поэтому «закавычивание» и замена спецсимволов на escape-последовательности не проходит. В таком случае помогает проверка типа; если переменная id не является числом, запрос вообще не должен выполняться

### 3) Усечение входных параметров

Для внесения изменений в логику выполнения SQL-запроса требуется внедрение достаточно длинных строк. Так, минимальная длина внедряемой строки в вышеприведённых примерах составляет 8 символов («1 OR 1=1»). Если максимальная длина корректного значения параметра невелика, то одним из методов защиты может быть максимальное усечение значений входных параметров.

Например, если известно, что поле id в вышеприведённых примерах может принимать значения не более 9999, можно «отрезать лишние» символы, оставив не более четырёх

### 4) Использование параметризованных запросов

Многие серверы баз данных поддерживают возможность отправки параметризованных запросов (подготовленные выражения). При этом параметры внешнего происхождения отправляются на сервер отдельно от самого запроса либо автоматически экранируются клиентской библиотекой.