### История развития безопасности в информационных системах.

**(из ЭРУД):** За несколько последних десятилетий требования к информационной безопасности существенно изменились. До начала широкого использования автоматизированных систем обработки данных безопасность информации достигалась исключительно физическими и административными мерами. С появлением компьютеров стала очевидной необходимость использования автоматических средств защиты файлов данных и программной среды.

Следующий этап развития автоматических средств защиты связан с появлением распределенных систем обработки данных и компьютерных сетей, в которых средства сетевой безопасности используются в первую очередь для защиты передаваемых по сетям данных.

**(из интернетов):** С развитием средств информационных коммуникаций, а следовательно и возможности нанесения ущерба информации, которая хранится и передается с их помощью, возникла информационная безопасность (ИБ). Основной задачей ИБ до 1816 (мб 1916??) года была защита разного рода информации, которая имеет для субъекта (организации или конкретного человека) особое значение. Внедрение и использование возможностей радиосвязи выявило необходимость обеспечения защищенности радиосвязи от помех с помощью применения помехоустойчивого кодирования и декодирования сигнала. Позже появились радиолокационные и гидроакустические средства (1935 год), ИБ которых обеспечивалась через повышение защищенности радиолокационных средств от воздействия радиоэлектронных помех. Начиная с 1946 года, с широким использованием в практической деятельности электронно-вычислительных машин (ЭВМ), ИБ достигалась, в основном, с помощью ограничения физического доступа к оборудованию, которое содержало или обрабатывало защищаемую информацию.

С 1965 года развивались локальные сети, информационная безопасность которых в основном достигалась путём администрирования и управления доступом к сетевым ресурсам. С началом использования мобильных коммуникационных устройств угрозы информационной безопасности стали гораздо серьёзнее и сложнее. Потребовалась разработка новых методов безопасности, так как для передачи и хранения информации широко использовались беспроводные сети передачи данных. Появились хакеры – сообщества людей, целью которых было нанесение ущерба ИБ разного объема (от отдельных пользователей до целых стран). С тех пор обеспечение информационной безопасности становится важнейшей и обязательной составляющей безопасности страны. С развитием глобальных сетей для решения задачи информационной безопасности должны решаться через создание макросистемы информационной безопасности всего человечества. Передача, обработка, хранение информации сегодня происходит исключительно с помощью информационных систем. Глобальные сети позволяют решать огромный спектр задач области связи (например, через электронную почту, мобильные телефоны), развлечений (MP3, цифровое телевидение, игры), транспорта (двигатели, навигация), торговли (кредитные карты, интернет-магазины ), медицины (оборудование, архивы медицинских материалов) и т.д.

### Понятие информации, защиты информации, информационной системы, безопасности автоматизированных систем обработки информации.

Эруд:

**Защита информации(ЗИ)** – это деятельность по предотвращению утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию

Интернет:

**Информация** - в широком смысле абстрактное понятие, имеющее множество значений, в зависимости от контекста. В узком смысле этого слова — сведения (сообщения, данные) независимо от формы их представления. В настоящее время не существует единого определения термина информация. С точки зрения различных областей знания, данное понятие описывается своим специфическим набором признаков. Информация — совокупность данных, зафиксированных на материальном носителе, сохранённых и распространённых во времени и пространстве.

**Информация — осознанные сведения об окружающем мире, которые являются объектом хранения, преобразования, передачи и использования.**

Сведения — это знания, выраженные в сигналах, сообщениях, известиях, уведомлениях и т. д. Каждого человека в мире окружает море информации различных видов. Несмотря на то, что единого строгого определения информации не существует, имеется возможность описать этот термин через характерные свойства: дуализм(двойственность), достоверность, полнота, адекватность, доступность и актуальность.

Основные **виды информации** по её форме представления, способам её кодирования и хранения, что имеет наибольшее значение для информатики, это:

* **графическая или изобразительная** — первый вид, для которого был реализован способ хранения информации об окружающем мире в виде наскальных рисунков, а позднее в виде картин, фотографий, схем, чертежей на бумаге, холсте, мраморе и др. материалах, изображающих картины реального мира;
* **звуковая (акустическая)** — мир вокруг нас полон звуков и задача их хранения и тиражирования была решена с изобретением звукозаписывающих устройств в 1877 г. (см., например, историю звукозаписи на сайте — <http://radiomuseum.ur.ru/index9.html>); её разновидностью является музыкальная информация — для этого вида был изобретен способ кодирования с использованием специальных символов, что делает возможным хранение её аналогично графической информации;
* **текстовая** — способ кодирования речи человека специальными символами — буквами, причем разные народы имеют разные языки и используют различные наборы букв для отображения речи; особенно большое значение этот способ приобрел после изобретения бумаги и книгопечатания;
* **числовая** — количественная мера объектов и их свойств в окружающем мире; особенно большое значение приобрела с развитием торговли, экономики и денежного обмена; аналогично текстовой информации для её отображения используется метод кодирования специальными символами — цифрами, причем системы кодирования (счисления) могут быть разными;
* **видеоинформация** — способ сохранения «живых» картин окружающего мира, появившийся с изобретением кино.

Существуют также виды информации, для которых до сих пор не изобретено способов их кодирования и хранения — это тактильная информация, передаваемая ощущениями, органолептическая, передаваемая запахами и вкусами и др.

точки зрения информатики наиболее важными представляются следующие общие качественные свойства: достоверность, полнота, точность, актуальность, полезность, ценность, своевременность, понятность, доступность, краткость и пр.

1. ***Объективность информации***. Информация в любом своём проявлении объективна, она отображает объективную действительность. Например фраза "На улице тёплая погода" означает, что человек её произнесший считает погоду на улице тёплой, т.е. информацией в данном случае будет являться то, что определённый человек произнёс фразу следующего содержания.
2. ***Достоверность информации***. Информация достоверна, если она отражает истинное положение дел. Достоверная информация помогает принять нам правильное решение. Недостоверной информация может быть по следующим причинам:
   * преднамеренное искажение (дезинформация) или непреднамеренное искажение субъективного свойства;
   * искажение в результате воздействия помех («испорченный телефон») и недостаточно точных средств ее фиксации.
3. ***Полнота информации***. Информацию можно назвать полной, если ее достаточно для понимания и принятия решений. Неполная информация может привести к ошибочному выводу или решению.
4. ***Точность информации*** определяется степенью ее близости к реальному состоянию объекта, процесса, явления и т. п.
5. ***Актуальность информации*** — важность для настоящего времени, злободневность, насущность. Только вовремя полученная информация может быть полезна.
6. ***Полезность (ценность) информации***. Полезность может быть оценена применительно к нуждам конкретных ее потребителей и оценивается по тем задачам, которые можно решить с ее помощью.

Защита информации АСОИ:

Эруд:

**Под *информационной безопасностью*** понимают защищенность информации от незаконного ознакомления, преобразования и уничтожения, а также защищенность информационных ресурсов от воздействий, направленных на нарушение их работоспособности.

Компоненты Автоматизированных систем обработки информации (далее – АСОИ) можно разбить на следующие группы:

* Аппаратные средства;
* Программное обеспечение;
* Данные;
* Персонал.

*Информационая безопасность* компьютерных систем достигается обеспечением *конфиденциальности, целостности и достоверности* обрабатываемых данных, а также доступности и целостности информационных компонентов и ресурсов системы.

*Конфиденциальность данных* – этот статус, предоставленный данным и определяющий требуемую степень их защиты. К конфиденциальным данным можно отнести, например, следующие: личная информация пользователей; учетные записи; данные о кредитных картах; бухгалтерские сведения.

Под *целостностью информации* понимается свойство информации сохранять свою структуру и/или содержание в процессе передачи и хранения. Целостность информации обеспечивается в том случае, если данные не в системе не отличаются в семантическом отношении от исходных документов, то есть если не произошло их случайного или преднамеренного разрушения.

*Достоверность информации* – свойство информации, выражающееся в строгой принадлежности субъекту, который является ее источником, либо тому субъекту, от которого эта информация принята.

Основным назначением АСОИ является переработка (сбор, хранение, обработка и выдача) информации, поэтому проблема обеспечения информационной безопасности является для АСОИ центральной. Обеспечение безопасности АСОИ предполагает организацию противодействия любому несанкционированному вторжению в процесс функционирования АСОИ, а также попыткам модификации, хищения, выведения из строя или разрушения ее компонентов, т.е. защиту всех компонентов АСОИ – аппаратных средств, программного обеспечения, данных и персонала.

**Существуют два подхода к проблеме обеспечения безопасности АСОИ: "фрагментарный" и комплексный.**

*"Фрагментарный" подход* направлен на противодействие четко определенным угрозам в заданных условиях. В качестве примеров реализации такого подхода можно указать отдельные средства управления доступом, автономные средства шифрования, специализированные антивирусные программы и т.п.

Достоинством такого подхода является высокая избирательность к конкретной угрозе. Существенным недостатком данного подхода является отсутствие единой защищенной среды обработки информации. Фрагментарные меры защиты информации обеспечивают защиту конкретных объектов АСОИ только от конкретной угрозы. Даже небольшое видоизменение угрозы ведет к потере эффективности защиты.

*Комплексный подход* ориентирован на создание защищенной среды обработки информации в АСОИ, объединяющей в единый комплекс разнородные меры противодействия угрозам. Организация защищенной среды обработки информации позволяет гарантировать определенный уровень безопасности АСОИ, что является несомненным достоинством комплексного подхода. К недостаткам этого подхода относятся: ограничения на свободу действий пользователей АСОИ, большая чувствительность к ошибкам установки и настройки средств защиты, сложность управления.

Комплексный подход применяют для защиты АСОИ крупных организаций или небольших АСОИ, выполняющих ответственные задачи или обрабатывающих особо важную информацию. Нарушение безопасности информации в АСОИ крупных организаций может нанести огромный материальный ущерб как самим организациям, так и их клиентам. Поэтому такие организации вынуждены уделять особое внимание гарантиям безопасности и реализовывать комплексную защиту. Комплексного подхода придерживаются большинство государственных и крупных коммерческих предприятий и учреждений. Этот подход нашел свое отражение в различных стандартах.

### Цель защиты информации.

*Цель Защиты информации (ЗИ)* – это желаемый результат ЗИ. Целью ЗТ может быть предотвращение ущерба собственнику, владельцу, пользователю информации в результате возможной утечки информации и/или несанкционированного и непреднамеренного воздействия на информацию. (эумкд)

**Цели:**

* предотвращение утечки, хищения, искажения, подделки инфор­мации;
* предотвращение угроз безопасности личности, общества, госу­дарства;
* предотвращение несанкционированных действий по уничтоже­нию, модификации, копированию, блокированию информации, предотвращение других форм незаконного вмешательства в информационные ресурсы и информационные системы, обеспече­ние правового режима как объекта собственности;
* защита конституционных прав граждан по сохранению личной тайны, конфиденциальности персональных данных, имеющих­ся в информационных системах;
* сохранение государственной тайны, конфиденциальности до­кументированной информации в соответствии с законодатель­ством;
* обеспечение прав субъектов в информационных процессах и при разработке, производстве и применении информационных систем, технологии и средств их обеспечения.

(<https://studopedia.org/8-110221.html>)

### Базовые свойства информации.

С точки зрения информационной безопасности выделяют следующие свойства информации: конфиденциальность, целостность и доступность.

*Конфиденциальность информации* – это ее свойство быть известной только допущенным и прошедшим проверку (*авторизованным*) субъектам системы. Для остальных субъектов системы эта информация должна быть неизвестной.

Проверка субъекта при допуске его к информации может осуществляться путем проверки знания им некого секретного ключа, пароля, идентификации его по фиксированных характеристикам и т.п.



*Целостность информации* – ее свойство быть неизменной в семантическом смысле при функционировании системы в условиях случайных или преднамеренных искажений или разрушающих воздействий.

*Доступность информации* – ее свойство быть доступной для авторизованных законных субъектов системы, готовность служб к обслуживанию запросов.

### Понятие угрозы безопасности информации.

**Угроза информационной безопасности** — совокупность условий и факторов, создающих опасность нарушения [информационной безопасности](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

Под угрозой (в общем) понимается потенциально возможное событие, действие (воздействие), процесс или явление, которые могут привести к нанесению ущерба чьим-либо интересам.

Угрозы информационной безопасности могут быть классифицированы по различным признакам:

* По аспекту информационной безопасности, на который направлены угрозы:
  + *Угрозы* [*конфиденциальности*](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) (неправомерный доступ к информации). Угроза нарушения конфиденциальности заключается в том, что информация становится известной тому, кто не располагает полномочиями доступа к ней. Она имеет место, когда получен доступ к некоторой информации ограниченного доступа, хранящейся в вычислительной системе или передаваемой от одной системы к другой. В связи с угрозой нарушения конфиденциальности, используется термин «утечка». Подобные угрозы могут возникать вследствие «человеческого фактора» (например, случайное делегировании тому или иному пользователю привилегий другого пользователя), сбоев работе программных и аппаратных средств. К информации ограниченного доступа относится государственная тайна и конфиденциальная информация (коммерческая тайна, персональные данные, профессиональные виды тайна: врачебная, адвокатская, банковская, служебная, нотариальная, тайна страхования, следствия и судопроизводства, переписки, телефонных переговоров, почтовых отправлений, телеграфных или иных сообщений (тайна связи), сведения о сущности изобретения, полезной модели или промышленного образца до официальной публикации (ноу-хау) и др.).
  + *Угрозы* [*целостности*](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) (неправомерное изменение данных). Угрозы нарушения целостности – это угрозы, связанные с вероятностью модификации той или иной информации, хранящейся в информационной системе. Нарушение целостности может быть вызвано различными факторами – от умышленных действий персонала до выхода из строя оборудования.
  + *Угрозы* [*доступности*](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) (осуществление действий, делающих невозможным или затрудняющих доступ к ресурсам информационной системы). Нарушение доступности представляет собой создание таких условий, при которых доступ к услуге или информации будет либо заблокирован, либо возможен за время, которое не обеспечит выполнение тех или иных бизнес-целей.
* По расположению источника угроз:
  + *Внутренние* (источники угроз располагаются внутри системы);
  + *Внешние* (источники угроз находятся вне системы).
* По размерам наносимого ущерба:
  + *Общие* (нанесение ущерба объекту безопасности в целом, причинение значительного ущерба);
  + *Локальные* (причинение вреда отдельным частям объекта безопасности);
  + *Частные* (причинение вреда отдельным свойствам элементов объекта безопасности).
* По степени воздействия на информационную систему:
  + *Пассивные* (структура и содержание системы не изменяются);
  + *Активные* (структура и содержание системы подвергается изменениям).
* По природе возникновения:
  + *Естественные* (объективные) — вызванные воздействием на информационную среду объективных физических процессов или стихийных природных явлений, не зависящих от воли человека;
  + *Искусственные* (субъективные) — вызванные воздействием на информационную сферу человека. Среди искусственных угроз в свою очередь выделяют:
    - *Непреднамеренные* (случайные) угрозы — ошибки программного обеспечения, персонала, сбои в работе систем, отказы вычислительной и коммуникационной техники;
    - *Преднамеренные* (умышленные) угрозы — неправомерный доступ к информации, разработка специального программного обеспечения, используемого для осуществления неправомерного доступа, разработка и распространение вирусных программ и т.д. Преднамеренные угрозы обусловлены действиями людей. Основные проблемы информационной безопасности связаны прежде всего с умышленными угрозами, так как они являются главной причиной преступлений и правонарушений.

### Виды угроз и возможные пути их проявления. (Беб)

(ЭРУД)

АСОИ = Автоматизированных систем обработки информации

По цели воздействия различают **три основных типа угроз** безопасности АСОИ:

1. угрозы нарушения конфиденциальности информации;
2. угрозы нарушения целостности информации;
3. угрозы нарушения работоспособности системы (отказы в обслуживании).

Классификация возможных угроз информационной безопасности АСОИ может быть приведена по ряду базовых **признаков**:

1. По природе возникновения различают:
   1. естественные угрозы;
   2. искусственные угрозы.
2. По степени преднамеренности проявления различают:
   1. угрозы, вызванные ошибками или халатностью персонала;
   2. угрозы преднамеренного действия.
3. По непосредственному источнику угроз. Источниками угроз могут быть:
   1. природная среда;
   2. человек;
   3. санкционированные программно-аппаратные средства;
   4. несанкционированные программно-аппаратные средства.
4. По положению источника угроз. Источник угроз может быть расположен:
   1. вне контролируемой зоны АСОИ;
   2. в пределах контролируемой зоны АСОИ;
   3. непосредственно в АСОИ.
5. По степени зависимости от активности АС. Угрозы проявляются:
   1. независимо от активности АСОИ;
   2. только в процессе обработки данных.
6. По степени воздействия на АСОИ различают:
   1. пассивные угрозы;
   2. активные угрозы.
7. По этап доступа пользователей или программ к ресурсам АСОИ различают:
   1. угрозы, проявляющиеся на этапе доступа к ресурсам АСОИ;
   2. угрозы, проявляющиеся после разрешения доступа к ресурсам АСОИ.
8. По способу доступа к ресурсам АСОИ различают:
   1. угрозы, с использованием стандартного пути доступа к ресурсам АСОИ;
   2. угрозы с использованием скрытого нестандартного пути доступа к ресурсам АСОИ.
9. По текущему месту расположения информации, хранимой и обрабатываемой в АСОИ, различают:
   1. угрозы доступа к информации на внешних запоминающих устройствах;
   2. угрозы доступа к информации в оперативной памяти;
   3. угрозы доступа к информации, циркулирующей в линиях связи;
   4. угрозы доступа к информации, отображаемой на терминале или печатаемой на принтере.

(Интернеты)

Под **утечкой информации** будем понимать безконтрольный выход конфиденциальной информации за пределы организации или круга лиц, которым она была доверена.

Канал утечки - путь от источника конфиденциальной информации к злоумышленнику, посредством которого последний может получить доступ к ней.

Каналы утечки:

1. Электромагнитные
   1. радиоканалы
   2. низкочастотный канал
   3. сетевой (наводки)
   4. заземление
   5. линейные (канал связи)
2. Акустический
3. Визуальный
4. Информационный
   1. канал коммутируемых линий связи
   2. канал выделенной линии связи
   3. канал локальной сети
   4. канал терминальных и периферийных устройств.
   5. канал магнитных носителей информации

Основные причины утечки информации:

* несоблюдение персоналом норм и требований
* ошибки проектирования и реализации систем связи
* ведение противником технической и агентурной разведки

### Основные направления инженерно-технической защиты информации.

<http://www.techstages.ru/setons-1183-1.html>

### Задачи и принципы инженерно-технической защиты информации.

**(из интернетов):** Инженерно-техническая защита информации включает комплекс организационных и технических мер по обеспечению безопасности информации техническими средствами. Она решает следующие **задачи**:

**1**. Предотвращение проникновения злоумышленника к источникам информации с целью ее уничтожения, хищения или изменения.

**2**. Защита носителей информации от уничтожения в результате воздействия стихийных сил и прежде всего, пожара и воды (пены) при его тушении.

**3.** Предотвращение утечки информации по различным техническим каналам.

Способы и средства решения первых двух задач не отличаются от способов и средств защиты любых материальных ценностей, третья задача решается исключительно способами и средствами инженерно-технической защиты информации. Для обеспечения эффективной инженерно-технической защиты информации необходимо определить:

**·** что защищать техническими средствами в конкретной организации, здании, помещении;

**·** каким угрозам подвергается защищаемая информация со стороны злоумышленников и их технических средств;

**·** какие способы и средства целесообразно применять для обеспечения безопасности информации с учетом как величины угрозы, так и затрат на ее предотвращение;

**·** как организовать и реализовать техническую защиту информации в организации.

Основы инженерно-технической защиты должны содержать как теоретические знания, так и методические рекомендации, обеспечивающие решение этих задач. В основу защиты должны быть положены следующие **принципы**:

**1.** Непрерывность защиты информации, характеризующая постоянную готовность системы защиты к отражению угроз безопасности информации в любое время;

**2.** Активность, предусматривающая прогнозирование действий злоумышленника, разработку и реализацию опережающих мер по защите;

**3.** Скрытность, исключающая ознакомление посторонних лиц со средствами и технологией защиты информации;

**4.** Целеустремленность, предполагающая сосредоточение усилий по предотвращению угроз наиболее ценной информации;

**5.** Комплексное использование различных способов и средств защиты информации, позволяющая компенсировать недостатки одних достоинствами других.

Эти принципы хотя и не содержат конкретных рекомендаций, однако определяют общие требования к способам и средствам защиты информации. Следующая группа принципов характеризует основные профессиональные подходы к организации защиты информации, обеспечивает рациональный уровень ее защиты и позволяет сократить затраты. Эта группа включает следующие принципы:

**6.** Соответствие уровня защиты ценности информации (принцип адекватности). Затраты на защиту не должны превышать цену защищаемой информации. В противном случае защита нерентабельна.

**7.** Гибкость защиты. Проявляется в возможности изменении степени защищенности в соответствии с изменившимися требованиями к безопасности информации.

**8.** Многозональность защиты, предусматривающая размещение источников информации в зонах с контролируемым уровнем ее безопасности; Каждая КЗ характеризуется уровнем безопасности находящейся в ней информации. Безопасность информации в зоне зависит от:

- расстояния от источника информации (сигнала) до злоумышленника или его средства добывания информации;

- количества и уровня защиты рубежей на пути движения злоумышленника или распространения иного носителя информации (например, поля);

- эффективности способов и средств управления допуском людей и автотранспорта в зону;

- мер по защите информации внутри зоны.

**9.** Многорубежность защиты информации на пути движения злоумышленника или распространения носителя

Рассмотренные выше принципы относятся к защите в целом. При построении системы защиты целесообразно учитывать также следующие принципы:

**10.** Минимизация дополнительных задач и требований к сотрудникам организации, вызванных мерами по защите информации;

**11.** Надежность в работе технических средств системы, исключающая как не реагирование на угрозы безопасности (пропуски угроз) информации, так и ложные реакции при их отсутствии;

**12.** Ограниченный и контролируемый доступ к элементам системы обеспечения безопасности информации;

**13.** Адаптируемость (приспособляемость) системы к изменениям окружающей среды;

### Обоснование метода ветвей и границ для решения задач частично целочисленного программирования. Описание алгоритма. Обсуждение правил выбора задач из списка.

Задача линейного программирования формулируется как:

Есть А, есть b, есть c. Нужно найти такое x, что Ax = b и при этом c^T x -> max, а также x\_i - целое для всех заранее оговорённых i.

Вспомним, что имеется метод под названием симплекс-метод, который находит такой x, но без условия целочисленности x. Так же симплекс-методу можно задавать ограничения d\_i < x\_i < d^i для каждого i.

Описание алгоритма

Сформулируем алгоритм для нахождения частично целочисленного решения:

1. У нас будет список задач, где в начале лежит исходная задача. Задачи в списке отличаются только ограничениями d; матрица A и вектора b и c остаются теми же.
2. Возьмём какую-то задачу из списка, пока она есть (далее будут описаны оптимизации выбора задачи из списка)
3. Запустим на ней симплекс с записанными в ней ограничениями и получим какое-то решение x, возможно не целочисленное
4. Проверим все x\_i, для которых задано условие целочисленности.
   1. Если находится такое x\_i, для которого не выполняется условие целочисленности, то мы разбиваем текущую задачу на 2 подзадачи, где условия отличаются только d\_i и d^i:
      1. В одной из задач d\_i, d^i = d\_i, floor(x\_i)
      2. В другой d\_i, d^i = ceil(x\_i), d^i
   2. Иначе же, мы нашли решение, которое удовлетворяет условиям, но возможно оно не максимизирует функцию ценности. Запомним его и пойдём дальше.
5. Переходим на шаг 2, пока есть задачи в списке
6. Выбираем максимальное решение из запомненных

Обоснование:

1. Алгоритм не отсекает целочисленные решения, при постановке более строгих условий на d
2. Постановка более строгих условий d для задачи не может увеличить функцию ценности.

А из 1 следует, что мы рассмотрим вообще все целочисленные решения, а из 2 следует, что во время 4b мы отрежем неперспективные решения, которые гарантированно хуже текущего.

Правила выбора из списка

Если мы нарисуем дерево ветвей и границ, то нетрудно показать, что значение функции ценности в сыновьях вершины будет не больше функции ценности в родителе. Тогда давайте на этапе 4а при разбиении задачи сохранять у сыновей s = c^T x родителя, а также давайте сохранять глобальный максимальный скор из всех рассмотренных задач. В таком случае, если глобальный скор больше, чем s, то нам нет смысла пытаться решать задачу эту задачу, то есть если s < global\_max\_score, то мы пропускаем эту задачу.

Также заметим, что чем быстрее мы увеличим global\_max\_score, тем больше неперспективных вершин мы отбросим (и меньше задач решим). Для этого предлагается брать из списка задачу с максимальным s.

### Основные понятия, относящиеся к алгоритмам симметричного шифрования.

Эруд: лекция 3.

(<https://iitp-pages.github.io/mzi_erud/Theory/content/lk3/lk3.html>)

**Основные понятия и определения**

Cимметричный шифр — способ шифрования, в котором для шифрования и расшифрования применяется один и тот же криптографический ключ.

До изобретения схемы асимметричного шифрования единственным существовавшим способом являлось симметричное шифрование.

Ключ алгоритма должен сохраняться в секрете обеими сторонами. Ключ алгоритма выбирается сторонами до начала обмена сообщениями.

Рассмотрим общую схему симметричного шифрования.



**Симметричные алгоритмы** можно классифицировать на:

1. *Моно- и многоалфавитные подстановки.*
2. *Перестановки*
3. *Гаммирование.*
4. *Блочные шифры.*

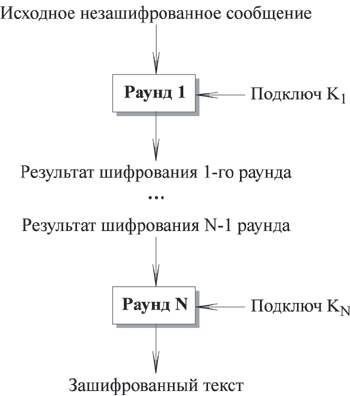
Алгоритмы симметричного шифрования различаются способом, которым обрабатывается исходный текст. Возможно шифрование блоками или шифрование потоком.

Все действия, производимые над данными блочным криптоалгоритмом, основаны на том факте, что преобразуемый блок может быть представлен в виде целого неотрицательного числа из диапазона, соответствующего его разрядности. Так, например, 32-битный блок данных можно интерпретировать как число из диапазона 0..4'294'967'295. Кроме того, блок, разрядность которого обычно является "степенью двойки", можно трактовать как несколько независимых неотрицательных чисел из меньшего диапазона (рассмотренный выше 32-битный блок можно также представить в виде 2 независимых чисел из диапазона 0..65535 или в виде 4 независимых чисел из диапазона 0..255).

Над этими числами блочным криптоалгоритмом и производятся по определенной схеме следующие действия:

* Биективные математические функции (cложение, исключающее ИЛИ, умножение по модулю 2N+1, умножение по модулю 2N);
* Битовые сдвиги (арифметические и циклические сдвиги влево и вправо);
* Табличные подстановки.

Описанные выше операции циклически повторяются в алгоритме, образуя так называемые *раунды*. Входом каждого *раунда* является выход предыдущего *раунда* и ключ, который получен по определенному алгоритму из ключа шифрования K. Ключ *раунда* называется *подключом*. Каждый алгоритм шифрования может быть представлен следующим образом:



Характерным признаком блочных алгоритмов является многократное и косвенное использование материала ключа. Это диктуется в первую очередь требованием невозможности обратного декодирования в отношении ключа при известных исходном и зашифрованном текстах. Для решения этой задачи в приведенных выше преобразованиях чаще всего используется не само значение ключа или его части, а некоторая, иногда необратимая (небиективная) функция от материала ключа. Более того, в подобных преобразованиях один и тот же блок или элемент ключа используется многократно. Это позволяет при выполнении условия обратимости функции относительно величины X сделать функцию необратимой относительно ключа Key.

Поскольку операция зашифровки или расшифровки отдельного блока в процессе кодирования пакета информации выполняется многократно (иногда до сотен тысяч раз), а значение ключа и, следовательно, функций *Vi*(Key) остается неизменным, то иногда становится целесообразно заранее однократно вычислить данные значения и хранить их в оперативной памяти совместно с ключом. Поскольку эти значения зависят только от ключа, то они в криптографии называются материалом ключа. Необходимо отметить, что данная операция никоим образом не изменяет ни длину ключа, ни криптостойкость алгоритма в целом. Здесь происходит лишь оптимизация скорости вычислений путем кеширования (англ. caching) промежуточных результатов. Описанные действия встречаются практически во многих блочных криптоалгоритмах и носят название расширение ключа (англ. key scheduling).

**Области применения**

Стандартный алгоритм шифрования должен быть применим во многих приложениях:

* Шифрование данных.
* Создание случайных чисел.
* Хэширование.

**Платформы**

Стандартный алгоритм шифрования должен быть реализован на различных платформах:

* Специализированная аппаратура.
* Большие процессоры.
* Процессоры среднего размера.
* Малые процессоры.

**Используемые критерии при разработке алгоритмов**

Считается, что алгоритм симметричного шифрования должен:

- Иметь размер блока 64 или 128 бит.

- Иметь масштабируемый ключ до 256 бит.

- Использовать простые операции, которые эффективны на микропро­цессорах, Не должно использоваться сдвигов переменной длины, побитных перестановок или условных переходов.

- Должна быть возможность реализации алгоритма на 8-битном про­цессоре с минимальными требованиями к памяти.

- Использовать заранее вычисленные подключи.

- Состоять из переменного числа итераций.

- По возможности не иметь слабых ключей.

- Задействовать подключи, которые являются односторонним хэшем ключа.

- Не иметь линейных структур, которые уменьшают комплексность и не обеспечивают исчерпывающий поиск.

- Использовать простую для понимания разработку.

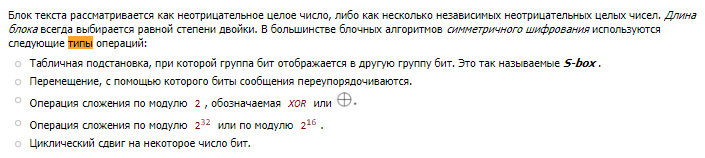
Там дальше идут штуки всякие: DES, сеть фейштеля, криптоанализ, blowfish хз насколько это надо

### Определение стойкости алгоритма, типы операций, используемые в алгоритмах симметричного шифрования.

**Криптографическая стойкость** (или криптостойкость) — способность криптографического алгоритма противостоять криптоанализу. Стойким считается алгоритм, атака на который требует от атакующего наличия столь значительных вычислительных ресурсов или огромных затрат времени на расшифровку перехваченных сообщений, что к моменту их расшифровки защищённая информация потеряет свою актуальность. В большом количестве случаев криптостойкость не может быть математически доказана; можно только доказать уязвимость криптографического алгоритма либо (в случае криптосистем с открытым ключом) свести задачу взлома алгоритма к некоторой задаче, которая считается вычислительно сложной (доказать, что взлом не легче решения этой задачи).

***Стойкость алгоритмов, где шифрование идет блоками:***

Блочный алгоритм преобразовывает n-битный блок незашифрованного текста в n-битный блок зашифрованного текста. Число блоков длины n равно 2 n. Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста. При маленькой длине блока такая подстановка плохо скрывает статистические особенности незашифрованного текста. Если блок имеет длину 64 бита, то он уже хорошо скрывает статистические особенности исходного текста. Но в данном случае преобразование текста не может быть произвольным в силу того, что ключом будет являться само преобразование, что исключает эффективную как программную, так и аппаратную реализации.



[https://ru.wikipedia.org/wiki/Симметричные\_криптосистемы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)

ХЗ ТУТ, надо еще поискать

### Сеть Фейштеля.

Наиболее широкое распространение получили сети Фейштеля, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны.

Сеть Фейштеля имеет следующую структуру. Входной блок делится на несколько равной длины подблоков, называемых ветвями. В случае, если блок имеет длину 64 бита, используются две ветви по 32 бита каждая. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево. Такое преобразование выполняется несколько циклов или раундов. В случае двух ветвей каждый раунд имеет структуру, показанную на рисунке ниже.

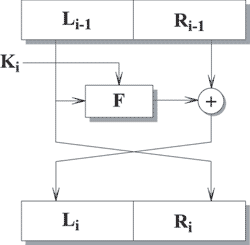


Рисунок 3.2 – I-ый раунд сети Фейштеля

Каждый раунд состоит из вычисления функции *F* для одной ветви и побитового выполнения операции XOR результата *F* с другой ветвью. После этого ветви меняются местами. Считается, что оптимальное число раундов – от 8 до 32. Важно то, что увеличение количества раундов значительно увеличивает криптостойкость алгоритма. Возможно, эта особенность и повлияла на столь активное распространение сети Фейштеля, так как для большей криптостойкости достаточно просто увеличить количество раундов, не изменяя сам алгоритм. В последнее время количество раундов не фиксируется, а лишь указываются допустимые пределы.

Сеть Фейштеля является обратимой даже в том случае, если функция *F* не является таковой, так как для расшифрования не требуется вычислять *F-1*.Для расшифрования используется тот же алгоритм, но на вход подается зашифрованный текст, и ключи используются в обратном порядке.

Достоинства сети Фейштеля:

* процедуры шифрования и расшифрования совпадают, с тем исключением, что ключевая информация при расшифровании используется в обратном порядке;
* хорошая изученность алгоритмов на основе сетей Фейштеля;
* для построения устройств шифрования можно использовать те же блоки в цепях шифрования и расшифрования.

Недостатком является то, что на каждой итерации изменяется только половина блока обрабатываемого текста, что приводит к необходимости увеличивать число итераций для достижения требуемой стойкости.

В настоящее время все чаще используются различные разновидности сети Фейштеля для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что наи­более популярными до сих пор остаются процессоры с 32-разрядными словами, следовательно, оперировать 32-разрядными словами эффективнее, чем с 64-разрядными.

Основной характеристикой алгоритма, построенного на основе сети Фейштеля, является функция *F*. Различные варианты касаются также на­чального и конечного преобразований. Подобные преобразования, назы­ваемые забеливанием (whitening), осуществляются для того, чтобы вы­полнить начальную рандомизацию входного текста.

### Основные понятия криптоанализа, линейный и дифференциальный криптоанализ. (Артём Г.)

(из ЭРУД)

Понятие дифференциального криптоанализа было введено Эли Бихамом (Biham) и Ади Шамиром (Shamir) в 1990 году. Конечная задача дифференциального криптоанализа - используя свойства алгоритма, в основном свойства S-box, определить подключ раунда. Конкретный способ дифференциального криптоанализа зависит от рассматриваемого алгоритма шифрования.

Если в основе алгоритма лежит сеть Фейштеля, то можно считать, что блок m состоит из двух половин - m0 и m1. Дифференциальный криптоанализ рассматривает отличия, которые происходят в каждой половине при шифровании. (Для алгоритма DES "отличия" определяются с помощью операции XOR, для других алгоритмов возможен иной способ). Выбирается пара незашифрованных текстов с фиксированным отличием. Затем анализируются отличия, получившиеся после шифрования одним раундом алгоритма, и определяются вероятности различных ключей. Если для многих пар входных значений, имеющих одно и то же отличие Х, при использовании одного и того же подключа одинаковыми (Y) оказываются и отличия соответствующих выходных значений, то можно говорить, что Х влечет Y с определенной вероятностью. Если эта вероятность близка к единице, то можно считать, что подключ раунда найден с данной вероятностью. Так как раунды алгоритма независимы, вероятности определения подключа каждого раунда следует перемножать. Как мы помним, считается, что результат шифрования данной пары известен. Результаты дифференциального криптоанализа используются как при разработке конкретных S-box, так и при определении оптимального числа раундов.

Другим способом криптоанализа является линейный криптоанализ, который использует линейные приближения преобразований, выполняемых алгоритмом шифрования. Данный метод позволяет найти ключ, имея достаточно большое число пар (незашифрованный текст, зашифрованный текст). Рассмотрим основные принципы, на которых базируется линейный криптоанализ. Обозначим

P[1], … , P[n] - незашифрованный блок сообщения.

C[1], … , C[n] - зашифрованный блок сообщения.

K[1], … , K[m] - ключ.

A[i, j, …, k] = A[i] ⊕ A[j] ⊕ … ⊕ A[k]

Целью линейного криптоанализа является поиск линейного уравнения вида

P[α1, α 2, …, α a] ⊕ C[β1, β2, …, βb ] = K[γ1, …, γ c]

Выполняющееся с вероятностью р <> 0.5. α i, βi и γi - фиксированные позиции в блоках сообщения и ключе. Чем больше р отклоняется от 0.5, тем более подходящим считается уравнение.

Это уравнение означает, что если выполнить операцию XOR над некоторыми битами незашифрованного сообщения и над некоторыми битами зашифрованного сообщения, получится бит, представляющий собой XOR некоторых битов ключа. Это называется линейным приближением, которое может быть верным с вероятностью р.

Уравнения составляются следующим образом. Вычисляются значения левой части для большого числа пар соответствующих фрагментов незашифрованного и зашифрованного блоков. Если результат оказывается равен нулю более чем в половине случаев, то полагают, что K[γ1, …, γс] = 0. Если в большинстве случаев получается 1, полагают, что K[γ1, …, γс] = 1. Таким образом получают систему уравнений, решением которой является ключ.

### Описание алгоритмов DES и тройного DES. (Беб)

*DES* является классической *сетью Фейстеля* с двумя ветвями. Данные шифруются **64-битными блоками**, используя **56-битный ключ**. Алгоритм преобразует за несколько *раундов* 64-битный вход в 64-битный выход. Длина ключа равна 56 битам. Процесс шифрования состоит из четырех этапов.

* На **первом** из них выполняется начальная перестановка (*IP*) 64-битного исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей.
* **Второй** этап состоит из 16 *раундов* одной и той же функции, которая использует операции сдвига и подстановки.
* На **третьем** этапе левая и правая половины выхода последней (16-й) итерации меняются местами.
* Наконец,на **четвертом** этапе выполняется перестановка IP-1 результата, полученного на третьем этапе. Перестановка IP-1 инверсна начальной перестановке.



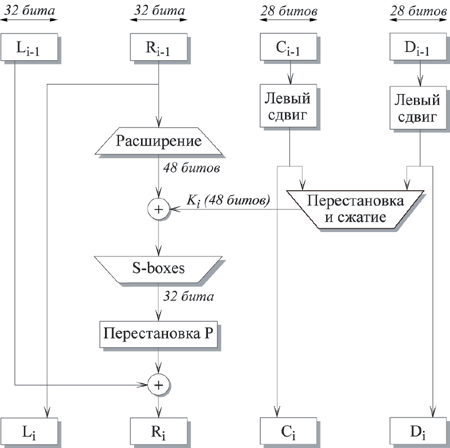
**Шифрование**

**Начальная перестановка**

Начальная перестановка и ее инверсия определяются стандартной таблицей. Если М- это произвольные 64 бита, то X = IP (M)-переставленные 64 бита. Если применить обратную функцию перестановки Y = IP-1 (X) = IP -1 (IP(M)), то получится первоначальная последовательность бит.

**Последовательность преобразований отдельного раунда**

Теперь рассмотрим последовательность преобразований, используемую в каждом *раунде*.



64-битный входной блок проходит через 16 *раундов*, при этом на каждой итерации получается промежуточное 64-битное значение. Левая и правая части каждого промежуточного значения трактуются как отдельные 32-битные значения, обозначенные *L* и *R*. Каждую итерацию можно описать следующим образом:

Li = Ri-1

Ri = Li-1⊕ F(Ri-1, Ki)

Где ⊕ обозначает операцию XOR.

Таким образом, выход левой половины *Li* равен входу правой половины *Ri-1*. Выход правой половины *Ri*является результатом применения операции XOR к *Li-1* и функции *F*, зависящей от *Ri-1* и *Ki*.

Рассмотрим функцию *F* более подробно.

*Ri* , которое подается на вход функции *F*, имеет длину 32 бита. Вначале Ri расширяется до 48 бит, используя таблицу, которая определяет перестановку плюс расширение на 16 бит. Расширение происходит следующим образом. 32 бита разбиваются на группы по 4 бита и затем расширяются до 6 бит, присоединяя крайние биты из двух соседних групп. Например, если часть входного сообщения

. . . efgh ijkl mnop . . .

то в результате расширения получается сообщение

. . . defghi hijklm lmnopq . . .

После этого для полученного 48-битного значения выполняется операция XOR с 48-битным *подключом Ki*. Затем полученное 48-битное значение подается на вход функции подстановки, результатом которой является 32-битное значение.

Подстановка состоит из восьми *S-boxes,*каждый из которых на входе получает 6 бит, а на выходе создает 4 бита. Эти преобразования определяются специальными таблицами. Первый и последний биты входного значения *S-box* определяют номер строки в таблице, средние 4 бита определяют номер столбца. Пересечение строки и столбца определяет 4-битный выход. Например, если входом является 011011, то номер строки равен 01 (строка 1) и номер столбца равен 1101 (столбец 13). Значение в строке 1 и столбце 13 равно 5, т.е. выходом является 0101.

Далее полученное 32-битное значение обрабатывается с помощью перестановки *Р*, целью которой является максимальное переупорядочивание бит, чтобы в следующем *раунде* шифрования с большой вероятностью каждый бит обрабатывался другим *S-box*.

**Создание подключей**

Ключ для отдельного *раунда Ki*состоит из 48 бит. Ключи *Ki* получаются по следующему алгоритму. Для 56-битного ключа,используемого на входе алгоритма, вначале выполняется перестановка в соответствии с таблицей Permuted Choice 1 (РС-1). Полученный 56-битный ключ разделяется на две 28-битные части, обозначаемые как C0 и D0 соответственно. На каждом *раунде Ci* и *Di*независимо циклически сдвигаются влево на 1 или 2 бита, в зависимости от номера *раунда*. Полученные значения являются входом следующего *раунда*. Они также представляют собой вход в Permuted Choice 2 (РС-2), который создает 48-битное выходное значение, являющееся входом функции *F*(*Ri-1*, *Ki*).

**Дешифрование**

Процесс дешифрования аналогичен процессу шифрования. На входе алгоритма используется зашифрованный текст, но ключи *Ki* используются в обратной последовательности. *K16* используется на первом *раунде*, *K1* используется на последнем *раунде*.

**Проблемой DES** является малая длина ключа. Также без ответа пока остается вопрос, возможен ли криптоанализ с использованием существующих характеристик алгоритма *DES*. Основой алгоритма являются восемь таблиц подстановки, или *S-boxes*, которые применяются в каждой итерации. Существует опасность, что эти *S-boxes* конструировались таким образом, что криптоанализ возможен для взломщика, который знает слабые места *S-boxes*. В течение многих лет обсуждалось как стандартное, так и неожиданное поведение *S-boxes*, но все-таки никому не удалось обнаружить их фатально слабые места.

**Двойной DES *(Не уверен что он нужен)***

Простейший способ увеличить длину ключа состоит в повторном применении *DES* с двумя разными ключами. Используя незашифрованное сообщение P и два ключа K1 и K2, зашифрованное сообщение С можно получить следующим образом:

C = Ek2 [Ek1 [P]]

При дешифрования два ключа применяются в обратном порядке:

P = Dk1 [Dk2 [C]]

В этом случае длина ключа равна 56 \* 2 = 112 бит.

**Атака "встреча посередине"**

Для приведенного выше алгоритма двойного *DES* существует так называемая атака "встреча посередине". Она основана на следующем свойстве алгоритма. Мы имеем

С = Ek2 [Ek1 [P]]

Тогда

X = Ek1 [P] = Dk2 [C].

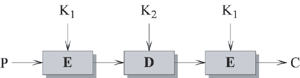
Атака состоит в следующем. Требуется, чтобы атакующий знал хотя бы одну пару незашифрованный текст и соответствующий ему зашифрованный текст: (Р, С). В этом случае, во-первых, шифруется Р для всех возможных 256 значений K1. Этот результат запоминается в таблице, и затем таблица упорядочивается по значению Х. Следующий шаг состоит в дешифровании С, с применением всех возможных 256 значений K2. Для каждого выполненного дешифрования ищется равное ему значение в первой таблице. Если соответствующее значение найдено, то считается, что эти ключи могут быть правильными, и они проверяются для следующей известной пары незашифрованный текст, зашифрованный текст.

**Тройной DES с двумя ключами**

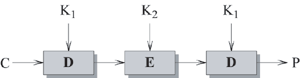
Очевидное противодействие атаке "встреча посередине" состоит в использовании третьей стадии шифрования с тремя различными ключами. Это поднимает стоимость лобовой атаки до 2168, которая на сегодняшний день считается выше практических возможностей. Но при этом длина ключа равна 56 \* 3 = 168 бит, что иногда бывает громоздко.

В качестве альтернативы предлагается метод тройного шифрования,использующий только два ключа. В этом случае выполняется последовательность зашифрование-расшифрование-зашифрование (EDE).

C = EK1 [DK2 [EK1 [P]]]



Шифрование



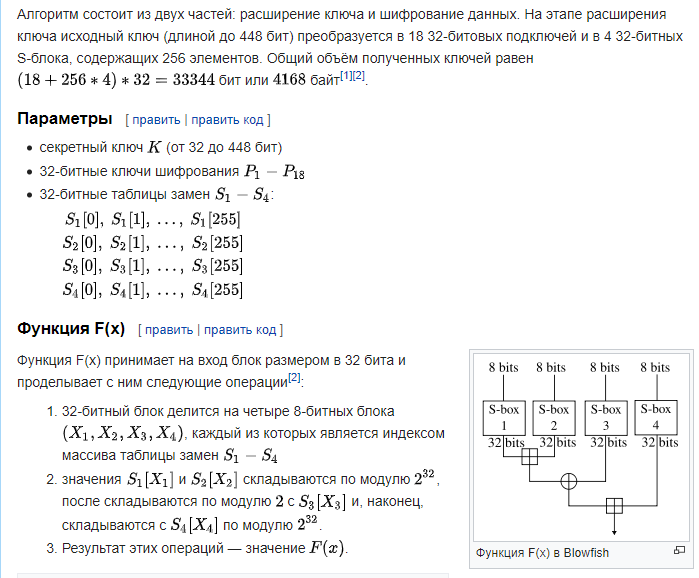
Дешифрование

Не имеет большого значения, что используется на второй стадии: шифрование или дешифрование. В случае использования дешифрования существует только то преимущество, что можно *тройной DES* свести к обычному одиночному *DES*, используя K1 = K2:

C = EK1 [DK1 [EK1 [P]]] = EK1 [P]

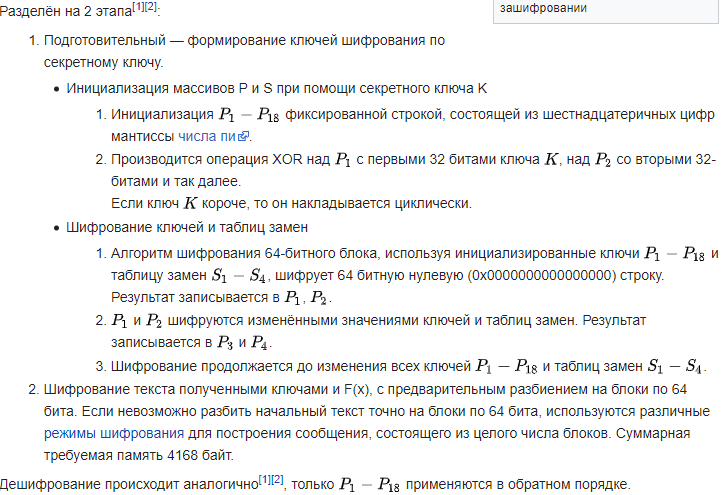
Известных криптографических атак на *тройной DES* не существует. Цена подбора ключа в *тройном DES* равна 2112.

### Алгоритм симметричного шифрования Blowfish, режимы выполнения.









**Режимы выполнения методов симметричного шифрования**

Для любого симметричного блочного алгоритма шифрования определено четыре режима выполнения.

***ECB*** - Electronic Codebook - каждый блок из 64 битов незашифрованного текста шифруется независимо от остальных блоков, с применением одного и того же *ключа шифрования*. Типичные приложения - безопасная передача одиночных значений.

***CBC*** - Cipher Block Chaining - вход криптографического алгоритма является результатом применения операции XOR к следующему блоку незашифрованного текста и предыдущему блоку зашифрованного текста. Типичные приложения - общая блокоориентированная передача.

***CFB*** - Cipher Feedback - при каждом вызове алгоритма обрабатывается J битов входного значения. Предшествующий зашифрованный блок используется в качестве входа в алгоритм; к J битам выхода алгоритма и следующему незашифрованному блоку из J битов применяется операция XOR, результатом которой является следующий зашифрованный блок из J битов. Типичные приложения - потокоориентированная передача, аутентификация.

***OFB*** - Output Feedback - аналогичен *CFB*, за исключением того, что на вход алгоритма при шифровании следующего блока подается результат шифрования предыдущего блока; только после этого выполняется операция XOR с очередными J битами незашифрованного текста. Типичные приложения - потокоориентированная передача по зашумленному каналу (например, спутниковая связь).

### Алгоритм симметричного шифрования IDEA, режимы выполнения.

***IDEA*** (International Data Encryption Algorithm) является блочным симметричным алгоритмом шифрования, разработанным Сюдзя Лай и Джеймсом Массей из швейцарского федерального института технологий. Первоначальная версия была опубликована в 1990 году. Пересмотренная версия алгоритма, усиленная средствами защиты от дифференциальных криптографических атак, была представлена в 1991году и подробно описана в 1992 году.

*IDEA* является блочным алгоритмом, который использует 128-битовый *ключ для шифрования данных блоками по 64 бита.*

Каждая операция *IDEA* выполняется над двумя 16-битными входами и создает один 16-битный выход. Этими операциями являются:

1. Побитовое исключающее OR, обозначаемое как ⊕

2. Сумма целых по модулю 216 (по модулю 65536), при этом входы и выходы трактуются как беззнаковые 16-битные целые. Эту операцию обозначим как +.

3. Умножение целых по модулю 216 + 1 (по модулю 65537), при этом входы и выходы трактуются как беззнаковые 16-битные целые, за исключением того, что блок из одних нулей трактуется как 216. Эту операцию обозначим как •.

Эти три операции являются несовместимыми в том смысле, что:

1. Не существует пары из трех операций, удовлетворяющих дистрибутивному закону. Например a • (b + c) <> (a • b) + (a • c)

2. Не существует пары из трех операций, удовлетворяющих ассоциативному закону. Например a + (b ⊕ c) <> (a + b) ⊕ c

**Шифрование**

*Алгоритм IDEA* состоит из восьми *раундов*, за которыми следует заключительное преобразование. Алгоритм разделяет блок на четыре 16-битных подблока. Каждый *раунд* получает на входе четыре 16-битных подблока и создает четыре 16-битных выходных подблока. Заключительное преобразование также получает на входе четыре 16-битных подблока и создает четыре 16-битных подблока. Каждый *раунд* использует шесть 16-битных ключей, заключительное преобразование использует четыре *подключа*, т.е. всего в алгоритме используется 52 *подключа*.

**Последовательность преобразований отдельного раунда**

Рассмотрим последовательность преобразований отдельного *раунда*.

Одним из основных элементов алгоритма, обеспечивающих диффузию,является структура, называемая МА (умножение/сложение):

На вход этой структуре подаются два 16-битных значения и два 16-битных *подключа*, на выходе создаются два 16-битных значения.

*Раунд* начинается с преобразования, которое комбинирует четыре входных подблока с четырьмя *подключами*, используя операции сложения и умножения. Четыре выходных блока этого преобразования комбинируются,используя операцию XOR для формирования двух 16-битных блоков,которые являются входами МА структуры. Кроме того, МА структура имеет на входе еще два *подключа* и создает два 16-битных выхода.

В заключении четыре выходных подблока первого преобразования комбинируются с двумя выходными подблоками МА структуры, используя XOR для создания четырех выходных подблоков данной итерации. Заметим,что два выхода, которые частично создаются вторым и третьим входами (Х2 и Х3 ), меняются местами для создания второго и третьего выходов (W 12 и W13). Это увеличивает перемешивание бит и делает алгоритм более стойким для дифференциального криптоанализа.

Рассмотрим девятый *раунд алгоритма*, обозначенный как заключительное преобразование. Это та же структура, что была описана выше. Единственная разница состоит в том, что второй и третий входы меняются местами. Это сделано для того, чтобы дешифрование имело ту же структуру, что и шифрование. Заметим, что девятая стадия требует только четыре входных *подключа*, в то время как для первых восьми стадий для каждой из них необходимо шесть входных *подключей*.

**Создание подключей**

Пятьдесят два 16-битных *подключа* создаются из 128-битного*ключа шифрования* следующим образом. Первые восемь *подключей*, которые обозначим как Z1, Z2, ..., Z8, получаются непосредственно из ключа, при этом Z 1 равен первым 16 битам, Z2 равен следующим 16 битам и т.д. Затем происходит циклический сдвиг ключа влево на 25 бит, и создаются следующие восемь *подключей*. Эта процедура повторяется до тех пор, пока не будут созданы все 52 *подключа*.

Заметим, что каждый первый *подключ раунда* получен из своего подмножества бит ключа. Если весь ключ обозначить как Z[1..128], то первыми ключами в восьми *раундах* будут:

**Z1 = Z [1..16] Z25 = Z [76..91]**

**Z7 = Z [97..112] Z31 = Z [44..59]**

**Z13 = Z [90..105] Z37 = Z [37..52]**

**Z19 = Z [83..98] Z43 = Z [30..45]**

Хотя на каждом *раунде* за исключением первого и восьмого используются только 96 бит *подключа*, множество бит ключа на каждой итерации не пересекаются, и не существует отношения простого сдвига между *подключами* разных *раундов*. Это происходит потому, что на каждом *раунде* используется только шесть *подключей*, в то время как при каждой ротации ключа получается восемь *подключей*.

**Дешифрование**

Процесс дешифрования аналогичен процессу шифрования. Дешифрование состоит в использовании зашифрованного текста в качестве входа в ту же самую структуру *IDEA*, но с другим набором ключей. Дешифрующие ключи U 1,...,U52 получаются из шифрующих ключей следующим образом:

1. Первые четыре *подключа* i-ого *раунда* дешифрования получаются из первых четырех *подключей* (10-i)-го *раунда* шифрования, где стадия заключительного преобразования считается 9-м *раундом*. Первый и четвертый ключи дешифрования эквивалентны мультипликативной инверсии по модулю (216 + 1) соответствующих первого и четвертого *подключей* шифрования. Для *раундов* со 2 по 8 второй и третий *подключи* дешифрования эквивалентны аддитивной инверсии по модулю (216) соответствующих третьего и второго *подключей* шифрования. Для *раундов* 1 и 9 второй и третий *подключи* дешифрования эквивалентны аддитивной инверсии по модулю (216) соответствующих второго и третьего *подключей* шифрования.
2. Для первых восьми *раундов* последние два *подключа* i *раунда* дешифрования эквивалентны последним двум *подключам* (9-i) *раунда* шифрования.  
    Для мультипликативной инверсии используется нотация Zj -1, т.е.:  
    **Zj • Zj-1 =1 mod (216 + 1)**

Так как 216 + 1 является простым числом, каждое ненулевое целое Zj <= 216имеет уникальную мультипликативную инверсию по модулю ( 216 + 1 ). Для аддитивной инверсии используется нотация ( -Zj ), таким образом, мы имеем: -Z j + Zj = 0 mod (216)

Для доказательства того, что алгоритм дешифрования с соответствующими *подключами* имеет корректный результат, рассмотрим одновременно процессы шифрования и дешифрования. Каждый из восьми *раундов* разбит на две стадии преобразования, первая из которых называется трансформацией, а вторая шифрованием.

При шифровании поддерживаются следующие соотношения на выходе трансформации:

Y1 = W81 • Z49 Y3 = W 82 + Z51

Y2 = W83 + Z50 Y4 = W 84 • Z52

Первая стадия первого *раунда* процесса дешифрования поддерживает следующие соотношения:

J11 = Y1 • U1 J13 = Y 3 + U3

J12 = Y2 + U2 J14 = Y 4 • U4

Подставляя соответствующие значения, получаем:

J11 = Y1 • Z49-1 = W81 • Z49 • Z49-1 = W 81

J12 = Y2 + -Z50 = W83 + Z50 = W83 + Z50 + -Z50 = W 83

J13 = Y3 + -Z51 = W82 + Z 51 + -Z51 = W82

J14 = Y4 • Z52-1 = W84 • Z52 • Z52-1 = W 84

Таким образом, выход первой стадии процесса дешифрования эквивалентен входу последней стадии процесса шифрования за исключением чередования второго и третьего блоков. Теперь рассмотрим следующие отношения:

W81 = I81 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84)

W82 = I83 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84)

W83 = I82 ⊕ MAL(I81 ⊕ I83, I82 ⊕ I84)

W84 = I84 ⊕ MAL(I81 ⊕ I83, I82 ⊕ I84)

Где MAR(X, Y)есть правый выход МА структуры с входами Х и Y, и MAL(X, Y) есть левый выход МА структуры с входами Х и Y. Теперь получаем

V11 = J11 ⊕ MAR (J11 ⊕ J13, J12 ⊕ J14) = W81 ⊕ MAR(W81 ⊕ W82, W83 ⊕ W84) = I81 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84) ⊕

MAR[ I81 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84) ⊕ I83 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84), I82 ⊕ MAL(I81 ⊕ I83, I82 ⊕ I84) ⊕ I84 ⊕ MAL(I81 ⊕ I83, I82 ⊕ I84) ] =

I81 ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84) ⊕ MAR(I81 ⊕ I83, I82 ⊕ I84) =I81

Аналогично мы имеем

V12 = I83

V13 = I82

V14 = I84

Таким образом, выход второй стадии процесса дешифрования эквивалентен входу предпоследней стадии процесса шифрования за исключением чередования второго и третьего подблоков. Аналогично можно показать,что

V81 = I11

V82 = I13

V83 = I12

V84 = I14

Наконец, так как выход трансформации процесса дешифрования эквивалентен первой стадии процесса шифрования за исключением чередования второго и третьего подблоков, получается, что выход всего процесса шифрования эквивалентен входу процесса шифрования.

### Алгоритм симметричного шифрования ГОСТ 28147, режимы выполнения. (Артём Г.)

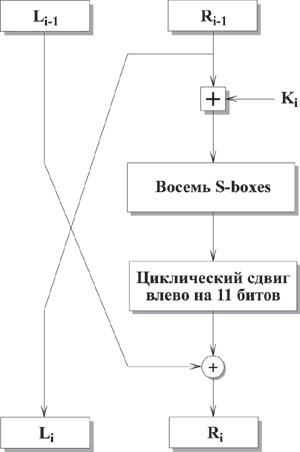
(из ЭРУД)

Алгоритм ГОСТ 28147 является отечественным стандартом для алгоритмов симметричного шифрования. ГОСТ 28147 разработан в 1989 году, является блочным алгоритмом шифрования, длина блока равна 64 битам, длина ключа равна 256 битам, количество раундов равно 32. Алгоритм представляет собой классическую сеть Фейштеля.

Li = Ri-1

Ri = Li ⊕ f (Ri-1, Ki)

Функция F проста. Сначала правая половина и i-ый подключ складываются по модулю 232. Затем результат разбивается на восемь 4-битовых значений, каждое из которых подается на вход S-box. ГОСТ 28147 использует восемь различных S-boxes, каждый из которых имеет 4-битовый вход и 4-битовый выход. Выходы всех S-boxes объединяются в 32-битное слово, которое затем циклически сдвигается на 11 битов влево. Наконец, с помощью XOR результат объединяется с левой половиной, в результате чего получается новая правая половина.



Генерация ключей проста. 256-битный ключ разбивается на восемь 32-битных подключей. Алгоритм имеет 32 раунда, поэтому каждый подключ используется в четырех раундах по следующей схеме:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Раунд | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Подключ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Раунд | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Подключ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Раунд | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Подключ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Раунд | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Подключ | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Считается, что стойкость алгоритма ГОСТ 28147 во многом определяется структурой S-boxes. Входом и выходом S-box являются 4-битные числа, поэтому каждый S-box может быть представлен в виде строки чисел от 0 до 15, расположенных в некотором порядке. Тогда порядковый номер числа будет являться входным значением S-box, а само число - выходным значением S-box.

**Режимы выполнения методов симметричного шифрования**

Для любого симметричного блочного алгоритма шифрования определено четыре режима выполнения.

***ECB*** - Electronic Codebook - каждый блок из 64 битов незашифрованного текста шифруется независимо от остальных блоков, с применением одного и того же *ключа шифрования*. Типичные приложения - безопасная передача одиночных значений.

***CBC*** - Cipher Block Chaining - вход криптографического алгоритма является результатом применения операции XOR к следующему блоку незашифрованного текста и предыдущему блоку зашифрованного текста. Типичные приложения - общая блокоориентированная передача.

***CFB*** - Cipher Feedback - при каждом вызове алгоритма обрабатывается J битов входного значения. Предшествующий зашифрованный блок используется в качестве входа в алгоритм; к J битам выхода алгоритма и следующему незашифрованному блоку из J битов применяется операция XOR, результатом которой является следующий зашифрованный блок из J битов. Типичные приложения - потокоориентированная передача, аутентификация.

***OFB*** - Output Feedback - аналогичен *CFB*, за исключением того, что на вход алгоритма при шифровании следующего блока подается результат шифрования предыдущего блока; только после этого выполняется операция XOR с очередными J битами незашифрованного текста. Типичные приложения - потокоориентированная передача по зашумленному каналу (например, спутниковая связь).

### Различные способы создания псевдослучайных чисел.

Эруд:

Большинство алгоритмов сетевой безопасности, основанных на криптографии, используют случайные числа. Например:

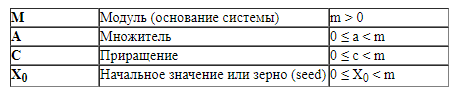
1. Схемы взаимной аутентификации.
2. Ключ сессии, созданный KDC или кем-либо из участников.

Двумя основными требованиями к последовательности случайных чисел являются *случайность* и *непредсказуемость*.

Шифрующие приложения используют для создания случайных чисел специальные алгоритмы. Эти алгоритмы детерминированы и, следовательно, создают последовательность чисел, которая не является статистически случайной. Тем не менее, если алгоритм хороший, полученная последовательность будет проходить много тестов на случайность. Такие числа часто называют *псевдослучайными числами*.

**Генераторы псевдослучайных чисел**

Первой широко используемой технологией создания случайного числа был алгоритм, предложенный Лехмером, который известен как метод линейного конгруента. Этот алгоритм параметризуется четырьмя числами следующим образом:



Последовательность случайных чисел {Xn} получается с помощью следующего итерационного равенства:

Xn+1 = (a Xn + c) mod m

Если m, а и с являются целыми, то создается последовательность целых чисел в диапазоне 0 ≤ Xn < m.

Существует три критерия, используемые при выборе генератора случайных чисел:

1. Функция должна создавать полный период, т.е. все числа между 0 и m до того, как создаваемые числа начнут повторяться.
2. Создаваемая последовательность должна появляться случайно. Последовательность не является случайной, так как она создается детерминированно, но различные статистические тесты, которые могут применяться, должны показывать, что последовательность случайна.
3. Функция должна эффективно реализовываться на 32-битных процессорах.

Значения а, с и m должны быть выбраны таким образом, чтобы эти три критерия выполнялись. В соответствии с первым критерием можно показать, что если m является простым и с = 0, то при определенном значении а период, создаваемый функцией, будет равен m-1. Для 32-битной арифметики соответствующее простое значение m = 231 - 1. Таким образом, функция создания псевдослучайных чисел имеет вид:

Xn+1 = (a Xn) mod (231 - 1)

Сила алгоритма линейного конгруента в том, что если сомножитель и модуль (основание) соответствующим образом подобраны, то результирующая последовательность чисел будет статистически неотличима от последовательности, являющейся случайной из набора 1, 2, ..., m-1. Но не может быть случайности в последовательности, полученной с использованием алгоритма, независимо от выбора начального значения Х0. Если значение выбрано, то оставшиеся числа в последовательности будут предопределены. Это всегда учитывается при криптоанализе.

Если противник знает, что используется алгоритм линейного конгруента, и если известны его параметры (а = 75, с = 0, m = 231 - 1), то, если раскрыто одно число, вся последовательность чисел становится известна. Даже если противник знает только, что используется алгоритм линейного конгруента, знания небольшой части последовательности достаточно для определения параметров алгоритма и всех последующих чисел. Предположим, что противник может определить значения Х0, Х1, Х 2, Х3. Тогда :

Х1 = (а Х0 + с ) mod m

Х2 = (а Х1 + с ) mod m

Х3 = (а Х2 + с ) mod m

Эти равенства позволяют найти а, с и m.

Таким образом, хотя алгоритм и является хорошим генератором псевдослучайной последовательности чисел, желательно, чтобы реально используемая последовательность была непредсказуемой, поскольку в этом случае знание части последовательности не позволит определить будущие ее элементы. Эта цель может быть достигнута несколькими способами. Например, использование внутренних системных часов для модификации потока случайных чисел. Один из способов применения часов состоит в перезапуске последовательности после N чисел, используя текущее значение часов по модулю m в качестве нового начального значения. Другой способ состоит в простом добавлении значения текущего времени к каждому случайному числу по модулю m.

***Генератор псевдослучайных чисел ANSI X9.17***

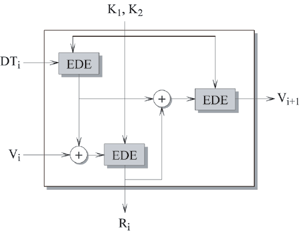
Один из наиболее сильных генераторов псевдослучайных чисел описан в ANSI X9.17. В число приложений, использующих эту технологию, входят приложения финансовой безопасности и PGP.

Алгоритмом шифрования является тройной DES. Генератор ANSI X9.17 состоит из следующих частей:

**Вход**: генератором управляют два псевдослучайных входа. Один является 64-битным представлением текущих даты и времени, которые изменяются каждый раз при создании числа. Другой является 64-битным начальным значением; оно инициализируется некоторым произвольным значением и изменяется в ходе генерации последовательности псевдослучайных чисел.

**Ключи**: генератор использует три модуля тройного DES. Все три используют одну и ту же пару 56-битных ключей, которая должна держаться в секрете и применяться только для генерации псевдослучайного числа.

**Выход**: выход состоит из 64-битного псевдослучайного числа и 64-битного значения, которое будет использоваться в качестве начального значения при создании следующего числа.



|  |
| --- |
| DTi - значение даты и времени на начало i-ой стадии генерации. |
| Vi - начальное значение для i-ой стадии генерации. |
| Ri - псевдослучайное число, созданное на i-ой стадии генерации. |
| K1, K2 - ключи, используемые на каждой стадии. |

Тогда:

Ri = EDEK1,K2 [ EDEK1,K2 [ DTi] ⊕ Vi ]

Vi+1 = EDEK1,K2 [ EDEK1,K2 [ DT i] Ri]

Схема включает использование 112-битного ключа и трех EDE-шифрований. На вход подаются два псевдослучайных значения: значение даты и времени и начальное значение очередной итерации, на выходе создаются начальное значение для следующей итерации и очередное псевдослучайное значение. Даже если псевдослучайное число Ri будет скомпрометировано, вычислить Vi+1 из Ri невозможно, и, следовательно, следующеепсевдослучайное значение Ri+1, так как для получения V i+1 дополнительно выполняются три операции EDE.

Из матмода можете вспомнить еще)

### Стандарт криптографической защиты AES.

**Advanced Encryption Standard** (**AES**) - является стандартом, основанным на алгоритме Rijndael.(см вопрос 20) Для AES длина input (блока входных данных) и State (состояния) постоянна и равна 128 бит, а длина шифроключа **K** составляет 128, 192, или 256 бит. При этом исходный алгоритм Rijndael допускает длину ключа и размер блока от 128 до 256 бит с шагом в 32 бита. Для обозначения выбранных длин input, State и Cipher Key в 32-битных словах используется нотация Nb = 4 для input и State, Nk = 4, 6, 8 для Cipher Key соответственно для разных длин ключей.

В [1997 год](https://ru.wikipedia.org/wiki/1997_%D0%B3%D0%BE%D0%B4)у Национальный Институт Стандартов и Технологий (США) объявляет о намерении выбрать преемника для шифрования [DES](https://ru.wikipedia.org/wiki/DES), являвшегося американским стандартом с [1977 года](https://ru.wikipedia.org/wiki/1977_%D0%B3%D0%BE%D0%B4). В [2000 год](https://ru.wikipedia.org/wiki/2000_%D0%B3%D0%BE%D0%B4)у было объявлено, что победителем конкурса стал алгоритм Rijndael[[4]](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard#cite_note-4), и началась процедура стандартизации. [28 февраля](https://ru.wikipedia.org/wiki/28_%D1%84%D0%B5%D0%B2%D1%80%D0%B0%D0%BB%D1%8F) [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) был опубликован проект, а [26 ноября](https://ru.wikipedia.org/wiki/26_%D0%BD%D0%BE%D1%8F%D0%B1%D1%80%D1%8F) [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) AES был принят как федеральный стандарт.

### Алгоритмы Rijndael и RC6. Математические понятия, лежащие в основе алгоритма Rijndael. Структура шифра.

Алгоритм *Rijndael* (читается "Рейндал") разработан бельгийскими специалистами. Этот шифр победил в проведенном Национальным институтом стандартов и техники (NIST) США конкурсе на звание AES (Advanced Encryption Standard) и в 2001 году был принят в качестве нового американского стандарта. Алгоритм Rijndael достаточно сложен для описания, поэтому рассмотрим только основные аспекты построения и особенности использования шифра.

Шифр *Rijndael/AES* (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа. В принципе, структуру *Rijndael* можно приспособить к любым размерам блока и ключа, кратным 32, а также изменить число раундов.

В отличие от шифров, предлагаемых *DES* и ГОСТ 28147-89, в основе *Rijndael* не лежит сеть Фейштеля. Основу *Rijndael* составляют так называемые *линейно-подстановочные преобразования*. Блок данных, обрабатываемый с использованием *Rijndael*, делится на массивы байтов, и каждая операция шифрования является байт-ориентированной. Каждый раунд состоит из трех различных обратимых преобразований, называемых слоями. Эти слои следующие.

1. Нелинейный слой. На этом слое выполняется замена байтов. Слой реализован с помощью S-блоков, имеющих оптимальную нелинейность, и предотвращает возможность использования дифференциального, линейного и других современных методов криптоанализа.
2. Линейный перемешивающий слой гарантирует высокую степень взаимопроникновения символов блока для маскировки статистических связей. На этом слое в прямоугольном массиве байтов выполняется сдвиг строк массива и перестановка столбцов.
3. Слой сложения по модулю 2 с подключом выполняет непосредственно шифрование.

Шифр начинается и заканчивается сложением с ключом. Это позволяет закрыть вход первого раунда при атаке по известному тексту и сделать криптографически значимым результат последнего раунда.

В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т.е. не зависят ни от ключа, ни от данных.

Необходимо отметить, что в отличие от шифров, построенных по сети Фейштеля, в *Rijndael* функции шифрования и расшифрования различны.

Алгоритм *Rijndael* хорошо выполняется как в программной, так и в аппаратной реализации. *Rijndael* имеет небольшие требования к памяти, что делает его пригодным для систем с ограниченными ресурсами. Надежность шифрования алгоритмом *Rijndael* очень высоко оценивается специалистами.

<https://www.intuit.ru/studies/courses/28/28/lecture/20420> (Тут про математические понятия, там очень много :3 )



(ну или вот тут, вроде тоже самое)

**RC6** — симметричный блочный [криптографический алгоритм](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), производный от алгоритма [RC5](https://ru.wikipedia.org/wiki/RC5). Был создан Роном Ривестом, Мэттом Робшау и Рэем Сиднеем для удовлетворения требований конкурса [Advanced Encryption Standard](https://ru.wikipedia.org/wiki/AES_(%D0%BA%D0%BE%D0%BD%D0%BA%D1%83%D1%80%D1%81)) (AES). Алгоритм был одним из пяти финалистов конкурса, был также представлен NESSIE и CRYPTREC. Является [собственническим](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%9F%D0%9E) (проприетарным) алгоритмом, и запатентован RSA Security.

Вариант шифра RC6, заявленный на конкурс AES, поддерживает блоки длиной 128 бит и ключи длиной 128, 192 и 256 бит, но сам алгоритм, как и RC5, может быть сконфигурирован для поддержки более широкого диапазона длин как блоков, так и ключей (от 0 до 2040 бит)[[1]](https://ru.wikipedia.org/wiki/RC6#cite_note-rc6-512-1). RC6 очень похож на RC5 по своей структуре и также довольно прост в реализации.

Является финалистом AES, однако одна из примитивных операций — операция умножения, медленно выполняемая на некотором оборудовании и затрудняет реализацию шифра на ряде аппаратных платформ и, что оказалось сюрпризом для авторов, на системах с архитектурой Intel IA-64 также реализована довольно плохо. В данном случае алгоритм теряет одно из своих ключевых преимуществ — высокую скорость выполнения, что стало причиной для критики и одной из преград для избрания в качестве нового стандарта. Однако, на системах с процессором [Pentium II](https://ru.wikipedia.org/wiki/Pentium_II), [Pentium Pro](https://ru.wikipedia.org/wiki/Pentium_Pro), [Pentium III](https://ru.wikipedia.org/wiki/Pentium_III), [PowerPC](https://ru.wikipedia.org/wiki/PowerPC) и [ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)) алгоритм RC6 опережает победителя — [Rijndael](https://ru.wikipedia.org/wiki/Rijndael)[[2]](https://ru.wikipedia.org/wiki/RC6#cite_note-rc6-aes-2).

### Понятия однонаправленной функции и однонаправленной функции с лазейкой. Функции дискретного логарифмирования и основанные на ней алгоритмы: схема Диффи-Хеллмана, схема Эль-Гамаля.

**Односторонняя функция** — [математическая функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), которая легко вычисляется для любого входного значения, но трудно найти аргумент по заданному значению функции. Здесь «легко» и «трудно» должны пониматься с точки зрения [теории сложности вычислений](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9). Разрыв между сложностью прямого и обратного преобразований определяет криптографическую эффективность односторонней функции. **(Из википедии)**

***Односторонней функцией*** называется такая функция, у которой каждый аргумент имеет единственное обратное значение, при этом вычислить саму функцию легко, а вычислить обратную функцию трудно. **(Из ЕРУДа)**

**Односторонняя функция с потайным входом** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *trapdoor function*, *TDF*) — это [односторонняя функция](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D1%8F%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) из множества  в множество , обладающая свойством (потайным входом, лазейкой), благодаря которому становится возможным найти для любого  такое, что , то есть обратить функцию. **(Из википедии)**

**Односторонняя функция с люком** принадлежит семейству *односторонних функций* fk таких, что

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

**Дискретный логарифм** определяется следующим образом. Вводится понятие *примитивного корня простого числа* Q как числа, чьи степени создают все целые от 1 до Q - 1. Это означает, что если А является *примитивным корнем простого числа* Q, тогда числа

A mod Q, A2 mod Q, . . . , AQ - 1 mod Q

являются различными и состоят из целых от 1 до Q - 1 с некоторыми перестановками. В этом случае для любого целого Y < Q и *примитивного корня* A простого числа Q можно найти единственную экспоненту Х, такую, что

Y = AХ mod Q, где 0≤X≤(Q - 1)

Экспонента X называется *дискретным логарифмом*, или индексом Y, по основанию A mod Q. Это обозначается как indA, Q (Y).

Теперь опишем алгоритм обмена ключей **Диффи-Хеллмана**.

* **Общеизвестные элементы.** *Q* : простое число; *A*: *A* < *Q* и *A* является примитивным корнем Q.
* **Создание пары ключей пользователем I.** Выбор случайного числа Хi (*закрытый ключ*), Xi < Q. Вычисление числа Yi (открытый ключ) Y i = A Хi mod Q.
* **Создание *открытого ключа* пользователем J.** Выбор случайного числа Хj (закрытый ключ) *Xj* < *Q*. Вычисление случайного числа Yj (открытый ключ) Yj = A *Xj* mod *Q*.
* **Создание общего секретного ключа пользователем I.** K = (Yj)Xi mod Q.
* **Создание общего секретного ключа пользователем J.** K = (Yi)Xj mod Q.

Предполагается, что существуют два известных всем числа: простое число Q и целое A, которое является *примитивным корнем* Q. Теперь предположим, что пользователи I и J хотят обменяться ключом для алгоритма симметричного шифрования. Пользователь I выбирает случайное число Х i < Q и вычисляет Yi = AXi mod Q. Аналогично пользователь J независимо выбирает случайное целое число Х j < Q и вычисляет Yj = AXj mod Q. Каждая сторона держит значение Х в секрете и делает значение Y доступным для другой стороны. Теперь пользователь I вычисляет ключ как К = (Y j)Xi mod Q, и пользователь J вычисляет ключ как K = (Yi)Xj mod Q. В результате оба получат одно и то же значение:

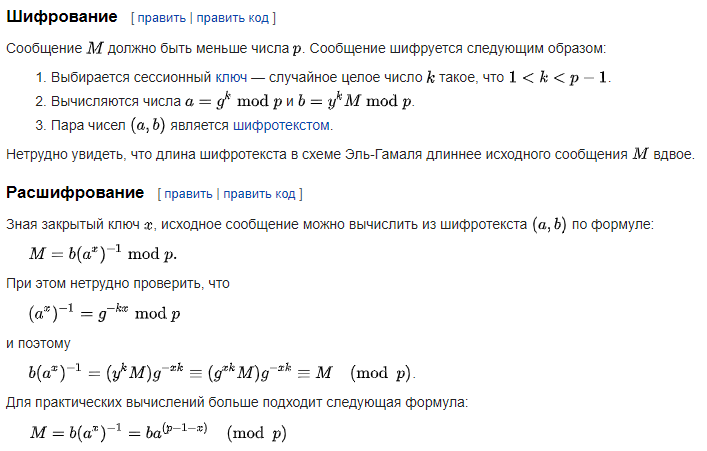
K = (Yj)Xi mod Q = (AXj mod Q)Xi mod Q = (AXj )Xi mod Q = A Xj Xi mod Q = (AXj )Xj mod Q = (AXi mod Q)Xj mod Q = (Yi)Xj mod Q

Таким образом, две стороны обменялись секретным ключом. Так как Х i и Хj являются закрытыми, противник может получить только следующие значения: Q, A, Yi и Yj. Для вычисления ключа атакующий должен взломать *дискретный логарифм*, т.е. вычислить

Xj = inda, q (Yj)

Следует заметить, что данный алгоритм уязвим для атак типа "man-in-the-middle".

**Эль-Гамаль**



### Схема RSA: алгоритм шифрования, его обратимость, вопросы стойкости.

### Основные понятия, относящиеся к обеспечению целостности сообщений с помощью МАС и хэш-функций; простые хэш-функции и сильная хэш-функция MD5.

ЭУМКД Лекция 10

### Сильные хэш-функции SHA-1, SHA-2 и ГОСТ 3411.

(из ЭРУД)

**Хэш-функция SHA-1**

Алгоритм получает на входе сообщение максимальной длины 264 бит и создает в качестве выхода дайджест сообщения длиной 160 бит.

Алгоритм состоит из следующих шагов:



***Шаг 1: добавление недостающих битов***

Сообщение добавляется таким образом, чтобы его длина была кратна 448 по модулю 512 (длина≡448 mod 512).

***Шаг 2: добавление длины***

К сообщению добавляется блок из 64 битов. Этот блок трактуется как беззнаковое 64-битное целое и содержит длину исходного сообщения до добавления.

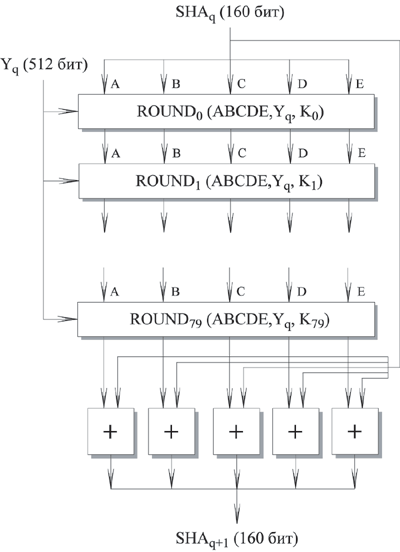
***Шаг 3: инициализация SHA-1 буфера***

Используется 160-битный буфер для хранения промежуточных и окончательных результатов хэш-функции. Буфер может быть представлен как пять 32-битных регистров A, B, C, D и E. Эти регистры инициализируются следующими шестнадцатеричными числами:

A = 67452301; B = EFCDAB89; C = 98BADCFE; D = 10325476; E = C3D2E1F0

***Шаг 4: обработка сообщения в 512-битных (16-словных) блоках***

Основой алгоритма является модуль, состоящий из 80 циклических обработок, обозначенный как HSHA. Все 80 циклических обработок имеют одинаковую структуру.



Каждый цикл получает на входе текущий 512-битный обрабатываемый блок Y q и 160-битное значение буфера ABCDE, и изменяет содержимое этого буфера.

В каждом цикле используется дополнительная константа Кt, которая принимает только четыре различных значения:

0 ≤ t ≤ 19 Kt = 5A827999 (целая часть числа [230 × 2 1/2])

20 ≤ t ≤ 39 Kt = 6ED9EBA1 (целая часть числа [230 × 3 1/2])

40 ≤ t ≤ 59 Kt = 8F1BBCDC (целая часть числа [230 × 5 1/2])

60 ≤ t ≤ 79 Kt = CA62C1D6 (целая часть числа [230 × 101/2])

Для получения SHAq+1 выход 80-го цикла складывается со значением SHAq. Сложение по модулю 232 выполняется независимо для каждого из пяти слов в буфере с каждым из соответствующих слов в SHA q.

***Шаг 5: выход***

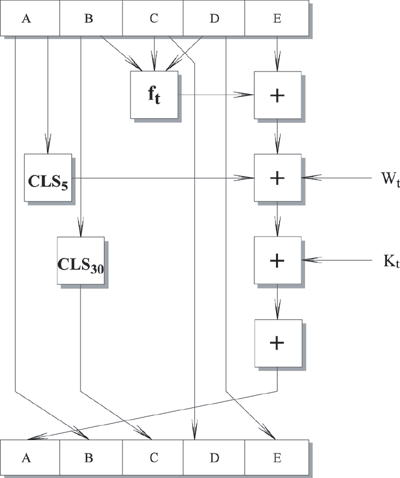
После обработки всех 512-битных блоков выходом L-ой стадии является 160-битный дайджест сообщения.

Рассмотрим более детально логику в каждом из 80 циклов обработки одного 512-битного блока. Каждый цикл можно представить в виде:

A, B, C, D, E (CLS5 (A) + ft (B, C, D) + E + W t + Kt), A, CLS30 (B), C, D

Где

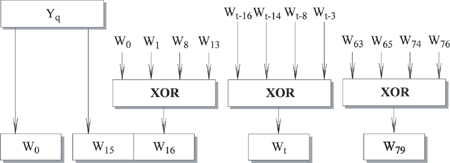
* A, B, C, D, E - пять слов из буфера.
* t - номер цикла, 0≤t≤79.
* ft – элементарная логическая функция.
* CLSs - циклический левый сдвиг 32-битного аргумента на s битов.
* Wt - 32-битное слово, полученное из текущего входного 512-битного блока.
* Kt - дополнительная константа.
* + - сложение по модулю 232.



Каждая элементарная функция получает на входе три 32-битных слова и создает на выходе одно 32-битное слово. Элементарная функция выполняет набор побитных логических операций, т.е. n-ый бит выхода является функцией от n-ых битов трех входов. Функции следующие:

|  |  |
| --- | --- |
| **Номер цикла** | **ft (B, C, D)** |
| (0≤t≤19) | (B ⊕ C) ⊕ (¬ B ⊕ D) |
| (20≤t≤39) | B ⊕ C ⊕ D |
| (40≤t≤59) | (B ⊕ C) ⊕ (B ⊕ D) ⊕ (C ⊕ D) |
| (60≤t≤79) | B ⊕ C ⊕ D |

32-битные слова Wt получаются из очередного 512-битного блока сообщения следующим образом.



Получение входных значений каждого цикла из очередного блока

Первые 16 значений Wt берутся непосредственно из 16 слов текущего блока. Оставшиеся значения определяются следующим образом:

Wt = Wt-16 ⊕ Wt-14 ⊕ Wt-8 ⊕ Wt-3

В первых 16 циклах вход состоит из 32-битного слова данного блока. Для оставшихся 64 циклов вход состоит из XOR нескольких слов из блока сообщения.

Алгоритм *SHA-1* можно суммировать следующим образом:

SHA0 = IV

SHAq+1 = Σ32 (SHAq , ABCDEq )

SHA = SHAL-1

Где

* IV - начальное значение буфера ABCDE.
* ABCDEq - результат обработки q-того блока сообщения.
* L - число блоков в сообщении, включая поля добавления и длины.
* Σ32 - сумма по модулю 232, выполняемая отдельно для каждого слова буфера.
* SHA - значение дайджеста сообщения.

**Хэш-функция SHA-2**

В 2001 году NIST принял в качестве стандарта три хэш-функции с существенно большей длиной хэш-кода. Часто эти хэш-функции называют *SHA-2* или SHA-256, SHA-384 и SHA-512 (соответственно, в названии указывается длина создаваемого ими хэш-кода). Эти алгоритмы отличаются не только длиной создаваемого хэш-кода, но и длиной обрабатываемого блока, длиной слова и используемыми внутренними функциями. Сравним характеристики этих хэш-функций.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Алгоритм | Длина сообщения  (в битах) | Длина блока  (в битах) | Длина слова  (в битах) | Длина дайджеста сообщения  (в битах) | Безопасность  (в битах) |
| SHA-1 | <264 | 512 | 32 | 160 | 80 |
| SHA-256 | <264 | 512 | 32 | 256 | 128 |
| SHA-384 | <2128 | 1024 | 64 | 384 | 192 |
| SHA-512 | <2128 | 1024 | 64 | 512 | 256 |

Под безопасностью здесь понимается стойкость к атакам типа "парадокса дня рождения".

В данных алгоритмах размер блока сообщения равен m бит. Для SHA-256 m = 512, для SHA-384 и SHA-512 m = 1024. Каждый алгоритм оперирует с w-битными словами. Для SHA-256 w = 32, для SHA-384 и SHA-512 w = 64. В алгоритмах используются обычные булевские операции над словами, а также сложение по модулю 2w, правый сдвиг на n бит SHRn (x) , где х - w-битное слово, и циклические (ротационные) правый и левый сдвиги на n бит ROTRn (x) и ROTLn (x), где х - w-битное слово.

SHA-256 использует шесть логических функций, при этом каждая из них выполняется с 32-битными словами, обозначенными как x, y и z. Результатом каждой функции тоже является 32-битное слово.

Ch (x, y, z) = (x ⊕ y) ⊕ (¬x ⊕ z)

Maj (x, y, z) = (x ⊕ y) ⊕ (x ⊕ z) ⊕ (y ⊕ z)

Σ0{256} (x) = ROTR2 (x) ⊕ ROTR13 (x) ⊕ ROTR22 (x)

Σ1{256} (x) = ROTR6 (x) ⊕ ROTR11 (x) ⊕ ROTR25 (x)

σ0{256} (x) = ROTR7 (x) ⊕ ROTR18 (x) ⊕ SHR3 (x)

σ1{256} (x) = ROTR17 (x) ⊕ ROTR19 (x) ⊕ SHR10 (x)

SHA-384 и SHA-512 также используют шесть логических функций, каждая из которых выполняется над 64-битными словами, обозначенными как x, y и z. Результатом каждой функции является 64-битное слово.

Ch (x, y, z) = (x ⊕ y) ⊕ (¬x ⊕ z)

Maj (x, y, z) = (x ⊕ y) ⊕ (x ⊕ z) ⊕ (y ⊕ z)

Σ0{512} (x) = ROTR28 (x) ⊕ ROTR34 (x) ⊕ ROTR39 (x)

Σ1{512} (x) = ROTR14 (x) ⊕ ROTR18 (x) ⊕ ROTR41 (x)

σ0{512} (x) = ROTR1 (x) ⊕ ROTR8 (x) ⊕ SHR7 (x)

σ1{512} (x) = ROTR19 (x) ⊕ ROTR61 (x) ⊕ SHR6 (x)

Предварительная подготовка сообщения, т.е. добавление определенных битов до целого числа блоков и последующее разбиение на блоки выполняется аналогично тому, как это делалось в *SHA-1* (конечно, с учетом длины блока каждой хэш-функции). После этого каждое сообщение можно представить в виде последовательности N блоков M(1), M(2), … , M (N).

Рассмотрим SHA-256. В этом случае инициализируются восемь 32-битных переменных, которые послужат промежуточным значением хэш-кода:

a, b, c, d, e, f, g, h

Основой алгоритма является модуль, состоящий из 64 циклических обработок каждого блока M(i):

T1 = h + Σ1{256}(e) + Ch(e, f, g) + K t{256} + Wt

T2 = Σ0{256}(a) + Maj(a, b, c)

h = g

g = f

f = e

e = d + T1

d = c

c = b

b = a

a = T1 + T2

где Ki{256} - шестьдесят четыре 32-битных константы, каждая из которых является первыми 32-мя битами дробной части кубических корней первых 64 простых чисел.

Wt вычисляются из очередного блока сообщения по следующим правилам:

Wt = Mt(i) , 0 ≤ t ≤ 15

Wt = σ1{256}(Wt-2) + Wt-7 + σ0{256}(Wt-15) + W t-16 , 16 ≤ t ≤ 63

i-ое промежуточное значение хэш-кода H(t) вычисляется следующим образом:

H0(i) = a + H0(i-1)

H1(i) = b + H1(i-1)

H2(i) = c + H2(i-1)

H3(i) = d + H3(i-1)

H4(i) = e + H4(i-1)

H5(i) = f + H5(i-1)

H6(i) = g + H6(i-1)

H7(i) = h + H7(i-1)

Теперь рассмотрим SHA-512. В данном случае инициализируются восемь 64-битных переменных, которые будут являться промежуточным значением хэш-кода:

a, b, c, d, e, f, g, h

Основой алгоритма является модуль, состоящий из 80 циклических обработок каждого блока M(i):

T1 = h + Σ1{512}(e) + Ch(e, f, g) + K t{512} + Wt

T2 = Σ0{512}(a) + Maj(a, b, c)

h = g

g = f

f = e

e = d + T1

d = c

c = b

b = a

a = T1 + T2

где Ki{512} - восемьдесят 64-битных констант, каждая из которых является первыми 64-мя битами дробной части кубических корней первых восьмидесяти простых чисел.

Wt вычисляются из очередного блока сообщения по следующим правилам:

Wt = Mt(i) , 0 ≤ t ≤ 15

Wt = σ1{512}(Wt-2) + Wt-7 + σ0{512}(Wt-15) + W t-16 , 16 ≤ t ≤79

i-ое промежуточное значение хэш-кода H(t) вычисляется следующим образом:

H0(i) = a + H0(i-1)

H1(i) = b + H1(i-1)

H2(i) = c + H2(i-1)

H3(i) = d + H3(i-1)

H4(i) = e + H4(i-1)

H5(i) = f + H5(i-1)

H6(i) = g + H6(i-1)

H7(i) = h + H7(i-1)

Рассмотрим SHA-384. Отличия этого алгоритма от SHA-512:

Другой начальный хэш-код H(0).

384-битный дайджест получается из левых 384 битов окончательного хэш-кода H (N): H0(N) || H1(N) || H2(N) || H3(N) || H 4(N) || H5(N).

**Хэш-функция ГОСТ 3411-94**

Алгоритм *ГОСТ 3411* является отечественным стандартом для хэш-функций. Длина хэш-кода, создаваемого алгоритмом *ГОСТ 3411*, равна 256 битам. Алгоритм разбивает сообщение на блоки, длина которых также равна 256 битам. Кроме того, параметром алгоритма является стартовый вектор хэширования *Н* - произвольное фиксированное значение длиной также 256 бит.

***Алгоритм обработки одного блока сообщения***

Сообщение обрабатывается блоками по 256 бит справа налево.

Каждый блок сообщения обрабатывается по следующему алгоритму.

* Генерация четырех ключей длиной 256 бит каждый.
* Шифрование 64-битных значений промежуточного хэш-кода H на ключах K i(i = 1, 2, 3, 4) с использованием алгоритма ГОСТ 28147 в режиме простой замены.
* Перемешивание результата шифрования.
* Для генерации ключей используются следующие данные:
  + промежуточное значение хэш-кода Н длиной 256 бит;
  + текущий обрабатываемый блок сообщения М длиной 256 бит;
  + параметры - три значения С2, С3 и С4 длиной 256 бит следующего вида: С2 и С4 состоят из одних нулей, а С3 равно
* 18 08 116 024 116 08 (08 18)2 18 08 (08 18)4 (18 0 8)4где степень обозначает количество повторений 0 или 1.

Используются две формулы, определяющие перестановку и сдвиг.

Перестановка Р битов определяется следующим образом: каждое 256-битное значение рассматривается как последовательность тридцати двух 8-битных значений.

Перестановка Р элементов 256-битной последовательности выполняется по формуле y = φ(x), где x - порядковый номер 8-битного значения в исходной последовательности; y - порядковый номер 8-битного значения в результирующей последовательности.

φ(i + 1 + 4(k - 1)) = 8i + k ; i = 0 ÷ 3, k = 1 ÷ 8

Сдвиг А определяется по формуле

A (x) = (x1 ⊕ x2) || x4 || x3 || x2

Где

* xi - соответствующие 64 бита 256-битного значения х,
* || обозначает конкатенацию.

Присваиваются следующие начальные значения:

i = 1, U = H, V = M.

W = U ⊕ V, K1 = Р (W)

Ключи K2, K3, K4 вычисляются последовательно по следующему алгоритму:

U = A(U) ⊕ Сi, V = A(A(V)), W = U ⊕ V, Ki = Р(W)

Далее выполняется шифрование 64-битных элементов текущего значения хэш-кода Н с ключами K1, K2, K3 и K4. При этом хэш-код Н рассматривается как последовательность 64-битных значений:

H = h4 || h3 || h2 || h1

Выполняется шифрование алгоритмом ГОСТ 28147:

si = EKi [hi] i = 1, 2, 3, 4

S = s1 || s2 || s3 || s4

Наконец на заключительном этапе обработки очередного блока выполняется перемешивание полученной последовательности. 256-битное значение рассматривается как последовательность шестнадцати 16-битных значений. Сдвиг обозначается Ψ и определяется следующим образом:

|  |
| --- |
| η16 || η15 || ... || η1 - исходное значение |
| η1 ⊕ η2 ⊕ η3 ⊕ η4 ⊕ η13 ⊕ η16 || η16 || ... || η2 - результирующее значение |

Результирующее значение хэш-кода определяется следующим образом:

Χ(M, H) = ψ61 (H ⊕ ψ (M ⊕ ψ 12(S)))

где

* H - предыдущее значение хэш-кода,
* М - текущий обрабатываемый блок,
* Ψi - i-ая степень преобразования Ψ.

Входными параметрами алгоритма являются:

* исходное сообщение М произвольной длины;
* стартовый вектор хэширования Н, длина которого равна 256 битам;
* контрольная сумма Σ, начальное значение которой равно нулю и длина равна 256 битам;
* переменная L, начальное значение которой равно длине сообщения.

Сообщение М делится на блоки длиной 256 бит и обрабатывается справа налево. Очередной блок i обрабатывается следующим образом:

1. H = Χ(Mi, H)
2. Σ = Σ ⊕ ' Mi
3. L рассматривается как неотрицательное целое число, к этому числу прибавляется 256 и вычисляется остаток от деления получившегося числа на 2 256. Результат присваивается L.  
   Где ⊕ ' обозначает следующую операцию: Σ и Mi рассматриваются как неотрицательные целые числа длиной 256 бит. Выполняется обычное сложение этих чисел и находится остаток от деления результата сложения на 2 256. Этот остаток и является результатом операции.

Самый левый, т.е. самый последний блок М' обрабатывается так:

1. Блок добавляется слева нулями так, чтобы его длина стала равна 256 битам.
2. Вычисляется Σ = Σ ⊕ ' Mi.
3. L рассматривается как неотрицательное целое число, к этому числу прибавляется длина исходного сообщения М и находится остаток от деления результата сложения на 2256.
4. Вычисляется Н = Χ(М', Н).
5. Вычисляется Н = Χ(L, Н).
6. Вычисляется Н = Χ(Σ, Н).

Значением функции хэширования является Н.

Следующая лекция - Коды аутентификации сообщений - МАС

### Основные понятия, относящиеся к обеспечению целостности сообщений и вычислению МАС с помощью алгоритмов симметричного шифрования, хэш-функций и алгоритма НМАС.

Лекция 10

<https://iitp-pages.github.io/mzi_erud/Theory/content/lk10/lk10.html>

### Основные требования к цифровым подписям, прямая и арбитражная цифровая подпись, стандарты цифровой подписи ГОСТ 3410 и DSS.

Лекция 12

**Стандарт цифровой подписи DSS**

Для создания цифровой подписи используется алгоритм DSA (Digital Signature Algorithm). В качестве хэш-алгоритма стандарт предусматривает использование алгоритма SHA-1 (Secure Hash Algorithm).

DSS использует алгоритм, который разрабатывался для использования только в качестве цифровой подписи.

DSS основан на трудности вычисления дискретных логарифмов и базируется на схеме, первоначально представленной ElGamal и Schnorr.

Существует три параметра, которые являются открытыми и могут быть общими для большой группы пользователей.

160-битное простое число q, т.е. 2159 < q < 2 160.

Простое число р длиной между 512 и 1024 битами должно быть таким, чтобы q было делителем (р - 1), т.е. 2L-1 < p < 2L, где 512 < L < 1024 и (p-1)/q является целым.

g = h(p-1)/q mod p, где h является целым между 1 и (р-1) и g должно быть больше, чем 1,10.

Зная эти числа, каждый пользователь выбирает закрытый ключ и создает открытый ключ.

1. Закрытый ключ отправителя  
   Закрытый ключ х должен быть числом между 1 и (q-1) и должен быть выбран случайно или псевдослучайно.  
   x - случайное или псевдослучайное целое, 0 < x < q.
2. Открытый ключ отправителя  
   Открытый ключ вычисляется из закрытого ключа как у = gx mod p. Вычислить у по известному х довольно просто. Однако, имея открытый ключ у, вычислительно невозможно определить х, который является дискретным логарифмом у по основанию g.  
   y = gx mod p
3. Случайное число, уникальное для каждой подписи.  
   *k* - случайное или псевдослучайное целое, 0 < k < q, уникальное для каждого подписывания.
4. Подписывание  
   Для создания подписи отправитель вычисляет две величины, r и s, которые являются функцией от компонент открытого ключа (p, q, g), закрытого ключа пользователя (х), хэш-кода сообщения Н (М) и целого k, которое должно быть создано случайно или псевдослучайно и должно быть уникальным при каждом подписывании.
   1. r = (gk mod p) mod q
   2. s = [ k-1 (H (M) + xr) ] mod q

Подпись = (r, s)

1. Проверка подписи  
   Получатель выполняет проверку подписи с использованием следующих формул. Он создает величину v, которая является функцией от компонент общего открытого ключа, открытого ключа отправителя и хэш-кода полученного сообщения. Если эта величина равна компоненте r в подписи, то подпись считается действительной.
   1. w = s-1 mod q
   2. u1 = [ H (M) w ] mod q
   3. u2 = r w mod q
   4. v = [ (gu1 yu2) mod p ] mod q
   5. подпись корректна, если v = r.

**Отечественный стандарт цифровой подписи ГОСТ 3410**

В стандарте *ГОСТ 3410* используется хэш-функция ГОСТ 3411, которая создает хэш-код длиной 256 бит. Это во многом обуславливает требования к выбираемым простым числам p и q:

* р должно быть простым числом в диапазоне 2509 < p < 2512 либо 21020 < p < 21024
* q должно быть простым числом в диапазоне 2254 < q < 2256, q также должно быть делителем (р-1).

Аналогично выбирается и параметр g. При этом требуется, чтобы gq (mod p) = 1.

В соответствии с теоремой Ферма это эквивалентно условию в DSS, что g = h(p-1)/q mod p.

Закрытым ключом является произвольное число х: 0 < x < q

Открытым ключом является число y: y = gx mod p

Для создания подписи выбирается случайное число k: 0 < k < q

Подпись состоит из двух чисел (r, s), вычисляемых по следующим формулам:

* r = (gk mod p) mod q
* s = (k H(M) + xr) mod q

Обратим внимание на отличия *DSS* и *ГОСТ 3410*.

1. Используются разные хэш-функции: в *ГОСТ 3410* применяется отечественный стандарт на хэш-функции ГОСТ 3411, в *DSS* используется SHA-1, которые имеют разную длину хэш-кода. Отсюда и разные требования на длину простого числа q: в *ГОСТ 3410* длина q должна быть от 254 бит до 256 бит, а в *DSS* длина q должна быть от 159 бит до 160 бит.
2. По-разному вычисляется компонента s подписи.

В *ГОСТ 3410* компонента s вычисляется по формуле: s = (k H(M) + xr) mod q

В *DSS* компонента s вычисляется по формуле: s = [k-1 (H(M) + xr)] mod q

Последнее отличие приводит к соответствующим отличиям в формулах для проверки подписи.

Получатель вычисляет

1. w = H(M)-1 mod q
2. u1 = w s mod q
3. u2 = (q-r) w mod q
4. v = [(gu1 yu2) mod p] mod q
5. Подпись корректна, если v = r.

Структура обоих алгоритмов довольно интересна. Заметим, что значение r совсем не зависит от сообщения. Вместо этого r есть функция от k и трех общих компонент открытого ключа. Мультипликативная инверсия k (mod p) (в случае *DSS*) или само значение k (в случае ГОСТ 3410) подается в функцию, которая, кроме того, в качестве входа имеет хэш-код сообщения и закрытый ключ пользователя. Эта функция такова, что получатель может вычислить r, используя входное сообщение, подпись, открытый ключ пользователя и общий открытый ключ.

В силу сложности вычисления дискретных логарифмов нарушитель не может восстановить k из r или х из s.

Другое важное замечание заключается в том, что экспоненциальные вычисления при создании подписи необходимы только для gk mod p. Так как это значение от подписываемого сообщения не зависит, оно может быть вычислено заранее. Пользователь может заранее просчитать некоторое количество значений r и использовать их по мере необходимости для подписи документов. Еще одна задача состоит в определении мультипликативной инверсии k -1 (в случае *DSS*). Эти значения также могут быть вычислены заранее.

### Математические понятия, связанные с эллиптическими кривыми, в частности задача дискретного логарифмирования на эллиптической кривой.

Преимущество подхода на основе *эллиптических кривых* в сравнении с задачей факторизации числа, используемой в RSA, или задачей целочисленного логарифмирования, применяемой в алгоритме Диффи-Хеллмана и в DSS, заключается в том, что в данном случае обеспечивается эквивалентная защита при

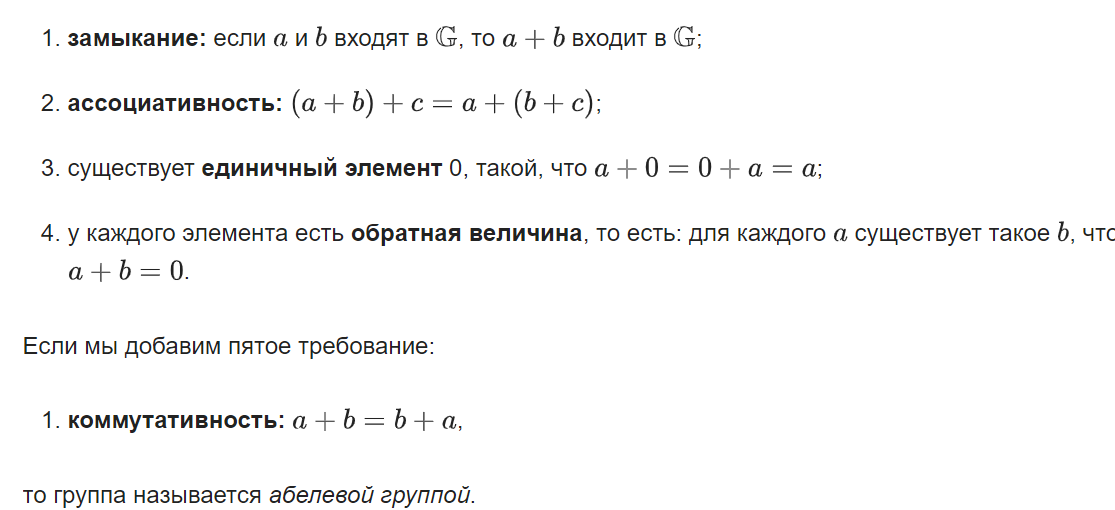
**Эллиптическая кривая** - кривая, множество точек, описываемых уравнением *эллиптической кривой* Е:

y2 + axy + by = x3 + cx2 + dx + e (в общем случае)

В криптографии с использованием *эллиптических кривых* все значения вычисляются по модулю р, где р является простым числом. Элементами данной *эллиптической кривой* являются пары неотрицательных целых чисел, которые меньше р и удовлетворяют частному виду *эллиптической кривой*:

y2 ≡ x3 + ax + b (mod p)

Такую кривую будем обозначать Ep (a,b). При этом числа а и b должны быть меньше р и должны удовлетворять условию 4a3 + 27b 2 (mod p) ≠ 0

Множество точек на кривой - это **математическая группа**. Группа — это множество, для которого мы определили двоичную операцию, называемую «сложением» и обозначаемую символом +. И у операции + следующие свойства:  


**Множество точек на *эллиптической кривой*** вычисляется следующим образом.

Для каждого такого значения х, что 0≤х≤р, вычисляется x3 + ax + b (mod p).

Для каждого из полученных на предыдущем шаге значений выясняется, имеет ли это значение квадратный корень по модулю р. Если нет, то в Ep (a,b) нет точек с этим значением х. Если корень существует, имеется два значения y, соответствующих операции извлечения квадратного корня (исключением является случай, когда единственным значением оказывается y = 0). Эти значения (x,y) и будут точками Ep (a,b).

**Множество точек Ep (a,b) обладает следующими свойствами**:

1. Р + 0 = Р
2. Если Р = (x,y), то Р + (x,-y) = 0. Точка (x,-y) является отрицательным значением точки Р и обозначается -Р. Заметим, что (x,-y) лежит на *эллиптической кривой* и принадлежит Ep (a,b).
3. Если Р = (x1,y1) и Q = (x2, y 2), где P ≠ Q, то P + Q = (x3,y3) определяется по следующим формулам:
4. x3 ≡ λ2 - x1 - x2 (mod p)
5. y3 ≡ λ (x1 - x3) - y 1 (mod p)  
   где (y2 - y1)/(x2 - x1) , если P ≠ Q  
   λ = (3x12 + a)/2y1 , если P = Q  
   Число λ есть угловой коэффициент секущей, проведенной через точки P = (x 1, y1) и Q = (x2, y2). При P = Q секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

**Задача дискретного логарифмирования на эллиптической кривой** - задача, которую должен решить атакующий (т.е. тот, кто хочет взломать).

***Формулировка*:** Даны точки P и Q на *эллиптической кривой* Ep (a,b).

Необходимо найти коэффициент k < p такой, что P = k × Q

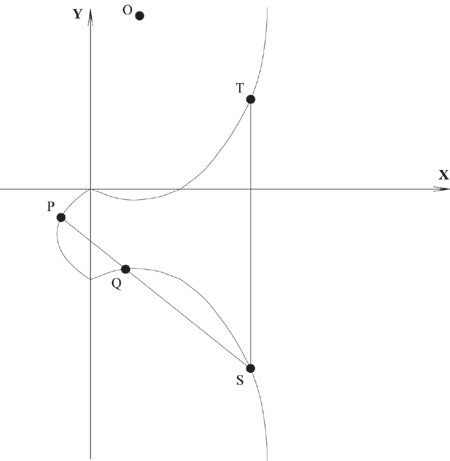
Относительно легко вычислить P по данным k и Q, но довольно трудно вычислить k, зная P и Q.

*Пример эллиптической кривой и того, как на множестве ее точек задаются операции сложения и умножения:*

В качестве примера рассмотрим *эллиптическую кривую* Е, уравнение которой имеет вид: y2 + y = x3 - x2

На этой кривой лежат только четыре точки, координаты которых являются целыми числами. Это точки

А (0, 0), В (1, -1), С (1, 0) и D (0, -1)



Для определения *операции сложения для точек на эллиптической кривой* сделаем следующие предположения:

* На плоскости существует бесконечно удаленная точка 0 ∈ Е, в которой сходятся все вертикальные прямые.
* Будем считать, что касательная к кривой пересекает точку касания два раза.
* Если три точки *эллиптической кривой* лежат на прямой линии, то их сумма есть 0.

Введем следующие правила сложения точек на *эллиптической кривой*:

* Точка 0 выступает в роли *нулевого элемента*. Так, 0 = -0 и для любой точки Р на *эллиптической кривой* Р + 0 = Р.
* Вертикальная линия пересекает кривую в двух точках с одной и той же координатой х - скажем, S = (x, y) и T = (x, -y). Эта прямая пересекает кривую и в бесконечно удаленной точке. Поэтому Р1 + Р 2 + 0 = 0 и Р1 = -Р2.
* Чтобы сложить две точки P и Q (см. рисунок 11.2) с разными координатами х, следует провести через эти точки прямую и найти точку пересечения ее с *эллиптической кривой*. Если прямая не является касательной к кривой в точках P или Q, то существует только одна такая точка, обозначим ее S. Согласно нашему предположению P + Q + S = О  
  Следовательно, P + Q = -S или P + Q = T.  
  Если прямая является касательной к кривой в какой-либо из точек P или Q, то в этом случае следует положить S = P или S = Q соответственно
* Чтобы удвоить точку Q, следует провести касательную в точке Q и найти другую точку пересечения S с *эллиптической кривой*.  
  Тогда Q + Q = 2 × Q = -S.

Еще норм тут:

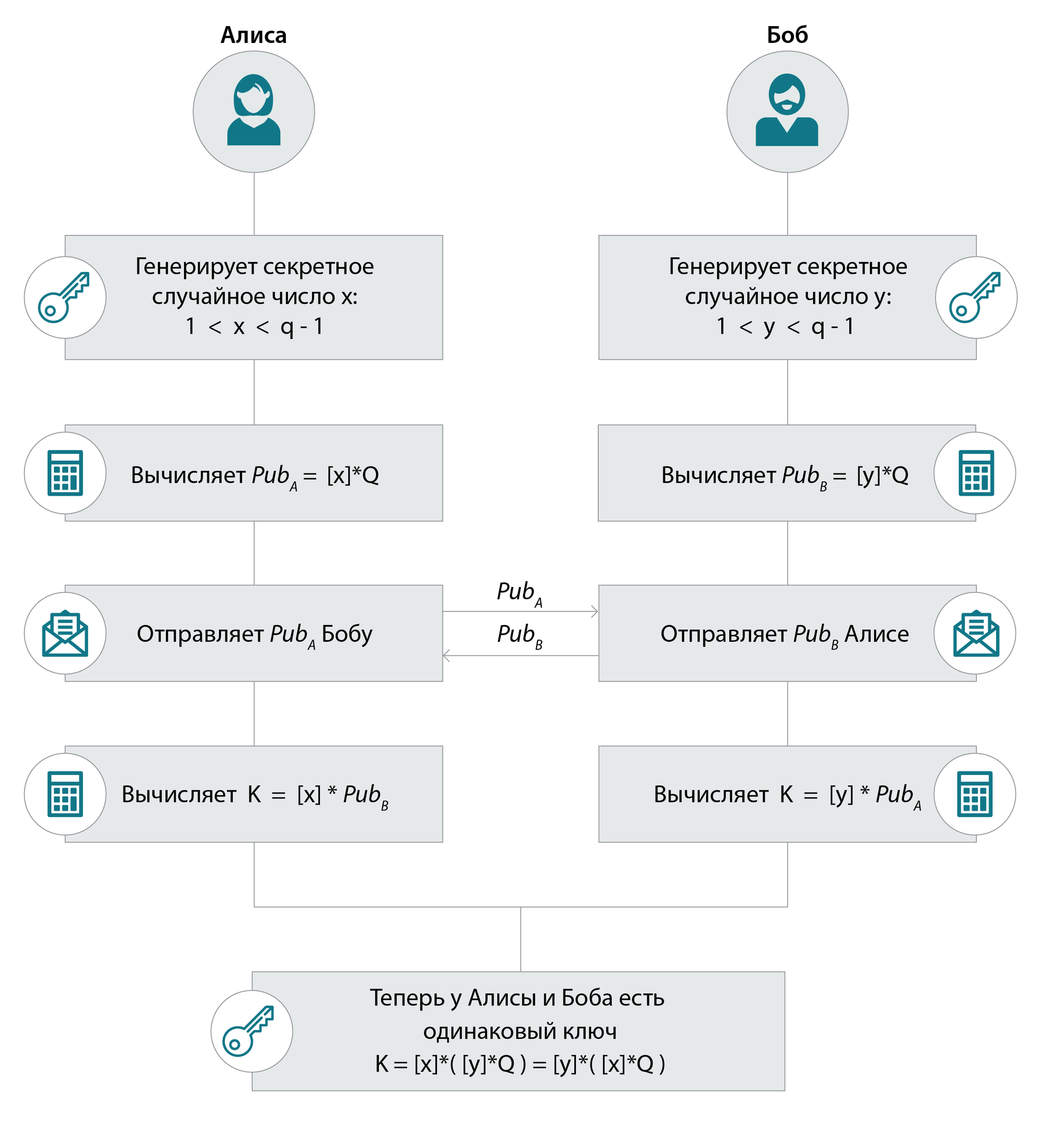
(<https://habr.com/ru/post/335906/>)

### Аналог алгоритма Диффи-Хеллмана на эллиптических кривых, алгоритма цифровой подписи на эллиптических кривых и алгоритма шифрования с открытым ключом получателя на эллиптических кривых.

Алиса и Боб выбирают Доменные параметры (domain parameters) для эллиптических кривых.

Из чего состоят Доменные параметры для эллиптических кривых:

(Лучше даже не знать...)



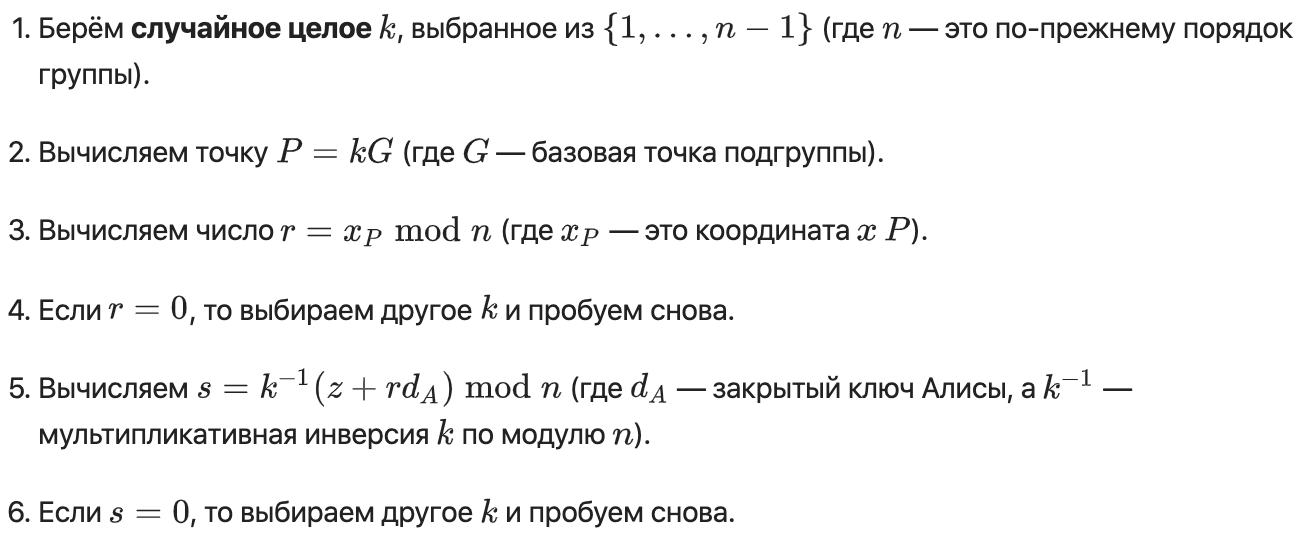
Эллиптический DH: в результате обмена ключами у Алисы и Боба появляется одинаковый элемент группы: ее генератор — точка Q, сложенная с собой x\*y раз: [x\*y]\*Q

Сценарий следующий: Алиса хочет подписать сообщение своим закрытым ключом (), а Боб хочет проверить подпись с помощью открытого ключа Алисы (). Никто, кроме Алисы не должен иметь возможности создать действительные подписи. Каждый должен иметь возможность проверить подписи.

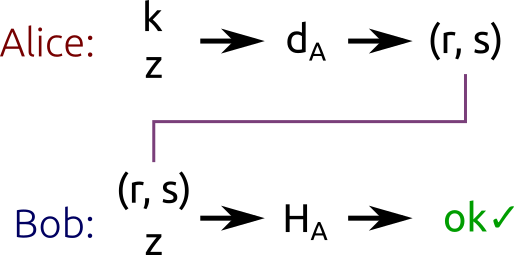
Алиса и Боб снова используют одинаковые параметры области определения. Мы рассмотрим алгоритм ECDSA, разновидность [Digital Signature Algorithm](https://en.wikipedia.org/wiki/Digital_Signature_Algorithm), применённого к эллиптическим кривым.

ECDSA работает с хешем сообщения, а не с самим сообщением. Выбор хеш-функции остаётся за нами, но, очевидно, нужно выбирать [криптографическую хеш-функцию](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F). Хеш сообщения необходимо урезать, чтобы битовая длина хеша была такой же, что и битовая длина n (порядок подгруппы). Урезанный хеш — это целое число и оно будет обозначаться как z.

Алгоритм, выполняемый Алисой для подписывания сообщения, работает следующим образом:



Пара (r,s) является подписью.



*Алиса подписывает хеш* z *с помощью закрытого ключа*  *и случайного* k*. Боб проверяет правильность подписи сообщения с помощью открытого ключа Алисы .*

Проще говоря, этот алгоритм сначала генерирует секретный ключ (k). Благодаря умножению точек (которое, как мы знаем, является «простым» в одну сторону и «сложным» в обратную) секретный ключ прячется в r. Затем r привязывается к хешу сообщения уравнением s=k−1(z+rdA)modn.

еyo

Учтите, что для вычисления s мы вычислили обратную величину k по модулю n. Как было сказано в предыдущей части, это гарантировано сработает только если n — простое число. Если подгруппа имеет порядок непростого числа, ECDSA использовать не удастся. Неслучайно все стандартизированные кривые имеют простой порядок, а имеющие непростой порядок неприменимы для ECDSA.

Алгоритм шифрования с открытым ключом получателя на эллиптических кривых:

<https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema8> (9.6 наверно, но там все сложно)) )

### Способы несанкционированного доступа к информации в компьютерных сетях. Классификация способов несанкционированного доступа и жизненный цикл атак.

### Способы противодействия несанкционированному межсетевому доступу. Функции межсетевого экранирования.

ТАНЯ

:c