Л6. Инструментальные средства управления ИБ

1.Особенности анализа и управления рисками информационных систем компаний с помощью метода CRAMM

Текущая версия CRAMM 5 , соответствует стандарту BS 7799 (ISO 17799).

Целью разработки метода являлось создание формализованной процедуры, позволяющей:

- убедиться, что требования, связанные с безопасностью, полностью проанализированы и документированы;
- избежать расходов на излишние меры безопасности, возможные при субъективной оценке

рисков;

- оказывать помощь в планировании и осуществлении защиты на всех стадиях жизненного

цикла информационных систем;

- обеспечить проведение работ в сжатые сроки;
- автоматизировать процесс анализа требований безопасности;
- представить обоснование для мер противодействия;
- оценивать эффективность контрмер, сравнивать различные их варианты;
- генерировать отчеты.

Концепция, положенная в основу метода

Анализ рисков включает идентификацию и вычисление уровней (мер) рисков на основе оценок,

присвоенных ресурсам, угрозам и уязвимостям ресурсов.

Контроль рисков состоит в идентификации и выборе контрмер, благодаря которым удается

снизить риски до приемлемого уровня.

Формальный метод, основанный на этой концепции, позволяет убедиться, что защита охватывает

всю систему и существует уверенность в том, что:

- все возможные риски идентифицированы;
- уязвимости ресурсов идентифицированы и их уровни оценены;
- угрозы идентифицированы и их уровни оценены;
- контрмеры эффективны;
- расходы, связанные с ИБ, оправданы.

Исследование ИБ системы с помощью CRAMM проводится в несколько этапов.

На первой стадии, Initiation, производится формализованное описание границ информационной системы, ее основных функций, категорий пользователей, а также персонала, принимающего участие в обследовании.

Риск определяется как – возможность потерь в результате какого-либо действия или события, способного нанести ущерб.

На стадии идентификации и оценки ресурсов, Identification and Valuation of Assets, описывается и анализируется все, что касается идентификации и определения ценности ресурсов системы. В конце этой стадии заказчик исследования будет знать, удовлетворит ли его существующая традиционная практика или он нуждается в проведении полного анализа рисков. В последнем случае будет построена модель информационной системы с позиции информационной безопасности.

Критерии оценки ценности ресурсов:

- Ущерб для репутации организации
- Безопасность персонала
- Разглашение персональных сведений
- Разглашение коммерческих сведений

- Неприятности со стороны правоохранительных органов
- Финансовые потери
- Невозможность нормальной работы организации

Стадия оценивания угроз и уязвимостей, Threat and Vulnerability Assessment, не является обязательной, если заказчика удовлетворит базовый уровень информационной безопасности. Эта стадия выполняется при проведении полного анализа рисков. Принимается во внимание все, что относится к идентификации и оценке уровней угроз для групп ресурсов и их уязвимостей. В конце стадии заказчик получает идентифицированные и оцененные уровни угроз и уязвимостей для своей системы.

Основные шаги:

- Идентификация угроз ресурсов и возможных уязвимостей.
- Группировка по угрозам или воздействиям с целью минимизации объема работы по анализу рисков.
- Измерение рисков.
- Получение отчета и обсуждение результатов с заказчиками.
- Коррекция по результатам обсуждения.

Оценка риска выполняется по двум факторам: вероятность реализации и размер ущерба.

P = P_реализации * Ущерб

Дальнейшая детализация вероятности реализации

P_реализации = P_угрозы* P_уязвимости

Угроза – действие или событие, способное нанести ущерб безопасности.

Уязвимость – слабость в защите ресурса или группы ресурсов, допускающая возможность реализации угрозы.

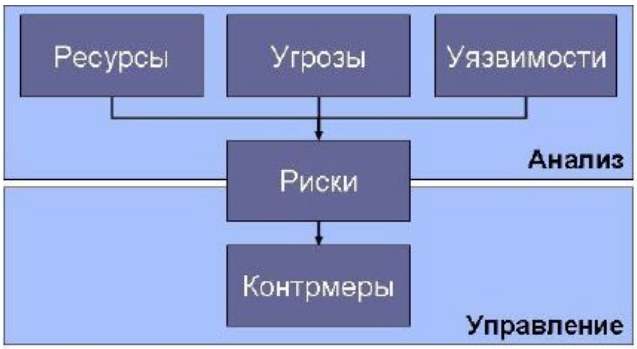
Стадия анализа рисков, Risk Analysis, позволяет оценить риски либо на основе сделанных оценок угроз и уязвимостей при проведении полного анализа рисков, либо путем использования упрощенных методик для базового уровня безопасности.

На стадии управления рисками, Risk Management, производится поиск адекватных контрмер. По существу речь идет о нахождении варианта системы безопасности, наилучшим образом удовлетворяющей требованиям заказчика. В конце стадии он будет знать, как модифицировать систему в терминах мер уклонения от риска, а также путем выбора специальных мер противодействия, ведущих к снижению или минимизации оставшихся рисков.

Выбор контрмер. Основные шаги:

- Генерация вариантов контрмер.
- Выбор подходящих вариантов и анализ их эффективности.
- Сравнительный анализ различных вариантов (What if)
- Получение отчета и обсуждение результатов с заказчиками.
- Коррекция по результатам обсуждения.

Каждая стадия объявляется законченной после детального обсуждения и согласования результатов с заказчиком.



2. Недостатки и преимущества метода CRAMM

Достоинства и недостатки метода CRAMM

Достоинства:

- хорошо апробированный метод
- удачная система моделирования ИТ
- обширная БД для оценки рисков и выбора контрмер
- возможность использования как средства аудита

Недостатки:

- большой объем отчетов
- сравнительно высокая трудоемкость

3. Особенности анализа и управления рисками информационных систем компаний с помощью программного обеспечения RiskWatch

Программное обеспечение RiskWatch является мощным средством анализа и управления рисками. В семейство RiskWatch входят программные продукты для проведения различных видов аудита безопасности.

В методе RiskWatch в качестве критериев для оценки и управления рисками используются предсказание годовых потерь (Annual Loss Expectancy, ALE) и оценка возврата от инвестиций (Return on Investment, ROI).

В отличие от CRAMM, программа RiskWatch более ориентирована на точную количественную оценку соотношения потерь от угроз безопасности и затрат на создание системы защиты. Надо также отметить, что в этом продукте риски в сфере информационной и физической безопасности компьютерной сети предприятия рассматриваются совместно.

4. Преимущества и недостатки программного обеспечения RiskWatch

Семейство программных продуктов RiskWatch имеет массу достоинств. RiskWatch помогает провести анализ рисков и сделать обоснованный выбор мер и средств защиты.

К недостаткам RiskWatch можно отнести:

- Такой метод подходит, если требуется провести анализ рисков на программно-техническом уровне защиты, без учета организационных и административных факторов.
- Полученные оценки рисков (математическое ожидание потерь) далеко не исчерпывает понимание риска с системных позиций - метод не учитывает комплексный подход к информационной безопасности.
- Программное обеспечение RiskWatch существует только на английском языке.
- Высокая стоимость лицензии (от 10 000 долл. за одно рабочее место для небольшой компании).

5. Основные этапы метода анализа рисков RiskWatch

Первый этап - определение предмета исследования. Здесь описываются такие параметры, как тип организации, состав исследуемой системы (в общих чертах), базовые требования в области безопасности. Для облегчения работы аналитика, в шаблонах, соответствующих типу организации ("коммерческая информационная система", "государственная/военная информационная система" и т.д.), есть списки категорий защищаемых ресурсов, потерь, угроз, уязвимостей и мер защиты. Из них нужно выбрать те, что реально присутствуют в организации.

Например, категории потерь:

- Задержки и отказ в обслуживании;

- Раскрытие информации;
- Прямые потери (например, от уничтожения оборудования огнем);
- Жизнь и здоровье (персонала, заказчиков и т.д.);
- Изменение данных;
- Косвенные потери (например, затраты на восстановление);
- Репутация.

Определение категорий защищаемых ресурсов.

Второй этап - ввод данных, описывающих конкретные характеристики системы. Данные могут вводиться вручную или импортироваться из отчетов, созданных инструментальными средствами исследования уязвимости компьютерных сетей.

На этом этапе:

Подробно описываются ресурсы, потери и классы инцидентов. Классы инцидентов получаются путем сопоставления категории потерь и категории ресурсов.

Для выявления возможных уязвимостей используется опросник, база которого содержит более 600 вопросов. Вопросы связаны с категориями ресурсов.

Задается частота возникновения каждой из выделенных угроз, степень уязвимости и ценность ресурсов. Все это используется в дальнейшем для расчета эффекта от внедрения средств защиты.

Пример оценок LAFE и SAFE для одной из угроз.

Третий и, наверное, самый важный этап - количественная оценка. На этом этапе рассчитывается профиль рисков, и выбираются меры обеспечения безопасности. Сначала устанавливаются связи между ресурсами, потерями, угрозами и уязвимостями, выделенными на предыдущих шагах исследования (риск описывается совокупностью этих четырех параметров).

Фактически, риск оценивается с помощью математического ожидания потерь за год. Например, если стоимость сервера $150000, а вероятность того, что он будет уничтожен пожаром в течение года равна 0.01, то ожидаемые потери составят $1500.

Общеизвестная формула (m=p*v , где m-математическое ожидание, p - вероятность возникновения угрозы, v - стоимость ресурса) претерпела некоторые изменения, в связи с тем, что RiskWatch использует определенные американским институтом стандартов NIST оценки, называемые LAFE и SAFE. LAFE (Local Annual Frequency Estimate) - показывает, сколько раз в год в среднем данная угроза реализуется в данном месте (например, в городе). SAFE (Standard Annual Frequency Estimate) - показывает, сколько раз в год в среднем данная угроза реализуется в этой "части мира" (например, в Северной Америке). Вводится также поправочный коэффициент, который позволяет учесть, что в результате реализации угрозы защищаемый ресурс может быть уничтожен не полностью, а только частично.

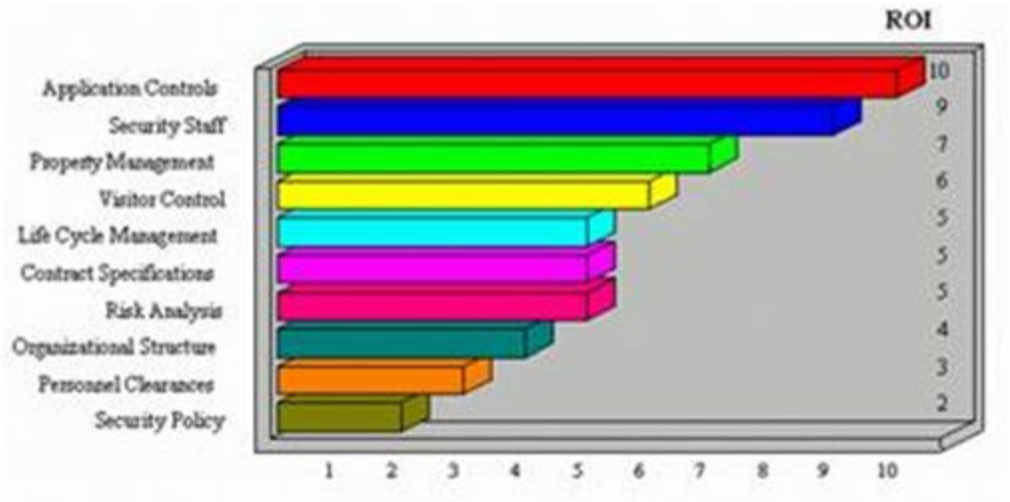
Дополнительно рассматриваются сценарии "что если:", которые позволяют описать аналогичные ситуации при условии внедрения средств защиты. Сравнивая ожидаемые потери при условии внедрения защитных мер и без них можно оценить эффект от таких мероприятий.

RiskWatch включает в себя базы с оценками LAFE и SAFE, а также с обобщенным описанием
различных типов средств защиты.
Эффект от внедрения средств защиты количественно описывается с помощью показателя ROI
(Return on Investment - отдача от инвестиций), который показывает отдачу от сделанных инвестиций за
определенный период времени. Рассчитывается он по формуле:

$$ROI = \sum_i NVP(Benefits_i) - \sum_i NVP(Costs_i)$$

где Costsi - затраты на внедрение и поддержание i-меры защиты; Benefitsi - оценка той пользы
(т.е. ожидаемого снижения потерь), которую приносит внедрение данной меры защиты; NPV (Net
Present Value) - дает поправку на инфляцию.
Четвертый этап - генерация отчетов. Типы отчетов:

- Краткие итоги.
- Полные и краткие отчеты об элементах, описанных на стадиях 1 и 2.
- Отчет от стоимости защищаемых ресурсов и ожидаемых потерях от реализации угроз.
- Отчет об угрозах и мерах противодействия.
- Отчет о ROI.
- Отчет о результатах аудита безопасности.



Пример графика показателя ROI для различных мер защиты.

6. Краткая характеристика современных пакетов управления рисками и их особенности

vsRisk. Программное обеспечение от британской компании Vigilant Software. Продукт позиционируется в первую очередь как программное обеспечение для оценки рисков в соответствии с требованиями **ISO 27001** и BS7799-3 (ныне ISO27005). На сайте можно скачать триалку на 15 дней (размер - 390 МБ). Система довольно примитивна и не дружественна неподготовленному пользователю. В программе, например, есть довольно **обширные списки** возможных **угроз, уязвимостей и контрмер,** но при этом сама **система не определяет никаких взаимосвязей между ними,** это делается вручную самим пользователем. Стоимость – около 1700 Евро.

PTA. Разработка компании PTA Technologies**.** Интересный продукт, который не привязан к какому-либо стандарту и **реализует механизм количественной оценки**

рисков (!). **Возможность качественной оценки не предусмотрена**. В системе предусмотрен **механизм** оценки эффективности предлагаемых контрмер, на основании чего можно, например, видеть как меняется карта рисков когда мы добавляем или убираем те или иные контрмеры.

На сайте разработчика указано, что продукт платный и для скачивания доступна только триалка на 30 дней.

RSA Archer. Разработка компании Archer, с недавних пор принадлежащей гиганту RSA. Archer - это такой **огромный GRC-комбайн**, который включает в себя множество различных модулей, один из которых обеспечивает управление рисками. Триалок для скачивания не предоставляется, на сайте есть презентационные видео.

Modulo Risk Manager. Разработка компании Modulo. На сайте доступно довольно бедное описание функционала. Никакой триальной версии, никаких подробных видео-роликов. Тем не менее **продукт отмечен наградой SC Magazine**.

RM Studio. Продукт одноименной организации (сайт). **Д**ля скачивания доступна 30-дневная триалка, помимо этого на сайте можно посмотреть видео-ролики, демонстрирующие сценарии использования продукта.

Вопросы по Л7 – Л9 Основные понятия криптологии

27.Назовите и охарактеризуйте основные разделы криптологии как науки.

Криптология:

1. Криптография (разработка шифров)
2. Криптоанализ (раскрытие шифров)

Криптография – наука о методах обеспечения конфиденциальности и аутентичности информации. Изначально криптография изучала методы шифрования информации – обратимого преобразования открытого текста на основе секретного алгоритма и/или ключа в шифрованный текст.

Криптография изучает:

- симметричные криптосистемы;
- ассиметричные криптосистемы (системы с отрытыми ключами);
- системы электронной цифровой подписи (ЭЦП);
- построение хеш функция;
- управление ключами;
- получений скрытой информации (стеганография);
- квантовую криптографию.

Криптоанализ — наука о методах расшифровки зашифрованной информации без предназначенного для такой расшифровки ключа.

Основные методы криптоанализа:

1. Атака на основе шифротекста
2. Атака на основе открытых текстов и соответствующих шифротекстов
3. Атака на основе подобранного открытого текста (возможность выбрать текст для шифрования)
4. Атака на основе адаптивно подобранного открытого текста

28.Предмет и основные понятия криптологии

Открытый (исходный) текст – данные (не обязательно текстовые), передаваемые без использования криптографии

Шифротекст, шифрованный (закрытый) текст – данные, полученные после применения криптосистемы (обычно с некоторым указанным ключом)

Шифрование – процесс нормально применения криптографического преобразования текста на основе алгоритма и ключа, в результате которого возникает шифрованный текст

Расшифровывание - процесс нормального применения криптографического преобразования, шифрованного текст в открытый.

Дешифрования (дешифровка) – процесс извлечения открытого текста на основе известного шифрованного без знания криптографического ключа.

Шифр, криптосистема – это множество преобразования открытого текста в шифрованный, с целью защиты от несанкционированного доступа.

29.Этапы развития криптографии. Особенности современного этапа развития криптографии.

1. Простейшие алгоритмы шифрования.
2. Использование полиалфавитных шифров.
3. Внедрение электромеханических устройств в работу шифровальщиков.
4. Переход к математической криптографии.
5. Современный период развития криптографии – криптография с открытым ключом.

Современный период развития криптографии (с конца 1970-х годов по настоящее время) отличается зарождением и развитием нового направления — криптография с открытым ключом. Её появление знаменуется не только новыми техническими возможностями, но и сравнительно широким распространением криптографии для использования частными лицами. Правовое регулирование использования криптографии частными лицами в разных странах сильно различается — от разрешения до полного запрета.

30.Что такое криптографический протокол и в чем заключаются его основные функции. Классификация криптографических протоколов.

Криптографический протокол — это абстрактный или конкретный протокол, включающий набор криптографических алгоритмов. В основе протокола лежит набор правил, регламентирующих использование криптографических преобразований и алгоритмов в информационных процессах.

Задачи

- Обеспечение различных режимов аутентификации
- Генерация, распределение и согласование криптографических ключей
- Защита взаимодействий участников
- Разделение ответственности между участниками

Классификация[\\\

- Протоколы шифрования / расшифрования
- Протоколы электронной цифровой подписи (ЭЦП)
- Протоколы идентификации / аутентификации
- Протоколы аутентифицированного распределения ключей

31.Управление ключами. Цели управления. Виды ключей. Понятие жизненного цикла ключа.

Управление ключами состоит из процедур, обеспечивающих:

- **включение пользователей в систему;**
- **выработку, распределение и введение в аппаратуру ключей;**
- **контроль использования ключей;**
- **смену и уничтожение ключей;**
- **архивирование, хранение и восстановление ключей.**

Управление ключами играет важнейшую роль в криптографии как основа для обеспечения конфиденциальности обмена информацией, идентификации и целостности данных. Важным свойством хорошо спроектированной системы управления ключами является сведение сложных проблем обеспечения безопасности многочисленных ключей к проблеме обеспечения безопасности нескольких ключей, которая может быть относительно просто решена путем обеспечения их физической изоляции в выделенных помещениях и защищенном от проникновения оборудовании. В случае использования ключей для обеспечения безопасности хранимой информации субъектом может быть единственный пользователь, который осуществляет работу с данными в последовательные промежутки времени. Управление ключами в сетях связи включает, по крайней мере, двух субъектов — отправителя и получателя сообщения.

Целью управления ключами является нейтрализация таких угроз, как:

- **компрометация конфиденциальности закрытых ключей;**
- **компрометация аутентичности закрытых или открытых ключей. При этом под аутентичностью понимается знание или возможность проверки идентичности корреспондента, для обеспечения конфиденциальной связи с которым используется данный ключ;**
- **несанкционированное использование закрытых или открытых ключей, например использование ключа, срок действия которого истек.**

Жизненный цикл

лючевая информация должна быть сменена до момента истечения срока действия ключа. Для этого может быть использована действующая ключевая информация, протоколы распределения ключей и ключевые уровни . Для того чтобы ограничить ущерб от компрометации ключей, следует избегать зависимостей между действующей и устанавливаемой ключевой информацией. Например, не рекомендуется защищать очередной сеансовый ключ с помощью действующего сеансового ключа. При хранении закрытых ключей должны быть приняты меры по обеспечению их конфиденциальности и аутентичности. При хранении открытых ключей должны быть приняты меры, позволяющие проверить их аутентичность. Конфиденциальность и аутентичность могут быть обеспечены криптографическими, организационными и техническими мерами.

Все криптосистемы, за исключением простейших, в которых используемые ключи зафиксированы раз и навсегда, нуждаются в периодической замене ключей. Эта замена

проводится с помощью определенных процедур и протоколов, в ряде которых используются и протоколы взаимодействия с третьей стороной. Последовательность стадий, которые проходят ключи от момента установления до следующей замены, называется жизненным циклом ключей.

1. ***Регистрация пользователей*. Эта стадия включает обмен первоначальной ключевой информацией, такой, как общие пароли или PIN-коды, путем личного общения или пересылки через доверенного курьера.**
2. ***Инициализация*. На этой стадии пользователь устанавливает аппаратное оборудование и/или программные средства в соответствии с установленными рекомендациями и правилами.**
3. ***Генерация ключей*. При генерации ключей должны быть приняты меры по обеспечению их необходимых криптографических качеств. Ключи могут генерироваться как самостоятельно пользователем, так и специальным защищенным элементом системы, а затем передаваться пользователю по защищенному каналу.**
4. ***Установка ключей*. Ключи устанавливаются в оборудование тем или иным способом. При этом первоначальная ключевая информация, полученная на стадии регистрации пользователей, может либо непосредственно вводиться в оборудование, либо использоваться для установления защищенного канала, по которому передается ключевая информация. Эта же стадия используется в последующем для смены ключевой информации.**
5. ***Регистрация ключей*. Ключевая информация связывается регистрационным центром с именем пользователя и сообщается другим пользователям ключевой сети. При этом для открытых ключей создаются сертификационным центром ключевые сертификаты, и эта информация публикуется тем или иным способом.**
6. ***Обычный режим работы*. На этой стадии ключи используются для защиты информации в обычном режиме.**
7. ***Хранение ключа*. Эта стадия включает процедуры, необходимые для хранения ключа в надлежащих условиях, обеспечивающих его безопасность до момента его замены.**
8. ***Замена ключа*. Замена ключа осуществляется до истечения его срока действия и включает процедуры, связанные с генерацией ключей, протоколами обмена**

ключевой информацией между корреспондентами, а также с доверенной третьей стороной. Для открытых ключей эта стадия обычно включает обмен информацией по защищенному каналу с сертификационным центром.

9. ***Архивирование*. В отдельных случаях ключевая информация после её использования для защиты информации может быть подвергнута архивированию для её извлечения со специальными целями (например, рассмотрения вопросов, связанных с отказами от цифровой подписи).**
10. ***Уничтожение ключей*. После окончания сроков действия ключей они выводятся из обращения, и все имеющиеся их копии уничтожаются. При этом необходимо следить, чтобы в случае уничтожения закрытых ключей тщательно уничтожалась и вся информация, по которой возможно их частичное восстановление.**
11. ***Восстановление ключей*. Если ключевая информация уничтожена, но не скомпрометирована (например, из-за неисправности оборудования или из-за того, что оператор забыл пароль) должны быть предусмотрены меры, дающие возможность восстановить ключ из хранимой в соответствующих условиях его копии.**
12. ***Отмена ключей*. В случае компрометации ключевой информации возникает необходимость прекращения использования ключей до окончания срока их действия. При этом должны быть предусмотрены необходимые меры оповещения абонентов сети. При отмене открытых ключей, снабженных сертификатами, одновременно производится прекращение действия сертификатов.**

Криптографические ключи различаются согласно алгоритмам, в которых они используются.

- **Секретные (Симметричные) ключи — ключи, используемые в симметричных алгоритмах (шифрование, выработка кодов аутентичности). Главное свойство симметричных ключей: для выполнения как прямого, так и обратного криптографического преобразования (шифрование/расшифровывание, вычисление MAC/проверка MAC) необходимо использовать один и тот же ключ (либо же ключ для обратного преобразования легко вычисляется из ключа для прямого преобразования, и наоборот). С одной стороны, это обеспечивает более**

высокую конфиденциальность сообщений, с другой стороны, создаёт проблемы распространения ключей в системах с большим количеством пользователей.

- **Асимметричные ключи — ключи, используемые в асимметричных алгоритмах (шифрование, ЭЦП). Более точно, они являются ключевой парой, поскольку состоят из двух ключей:**
    - **Закрытый ключ (en:Private key) — ключ, известный только своему владельцу. Только сохранение пользователем в тайне своего закрытого ключа гарантирует невозможность подделки злоумышленником документа и цифровой подписи от имени заверяющего.**
    - **Открытый ключ (en:Public key) — ключ, который может быть опубликован и используется для проверки подлинности подписанного документа, а также для предупреждения мошенничества со стороны заверяющего лица в виде отказа его от подписи документа. Открытый ключ подписи вычисляется, как значение некоторой функции от закрытого ключа, но знание открытого ключа не дает возможности определить закрытый ключ.**

32.Теоретические основы развития криптографии

33. Охарактеризовать элементы теории информации, используемые в криптографии

34. Проблемы и особенности применения теории сложности в криптографии
35. Теория чисел и криптография – применение, особенности, необходимость
36. Принципы разложения на множители и их применение в криптографии
37. Проблемы и особенности использования генераторов случайных чисел в криптографии.
38. Что такое симметричное шифрование. Дайте определение и укажите наиболее известные симметричные алгоритмы
39. Что такое асимметричное шифрование. Дайте определение и укажите наиболее известные асимметричные алгоритмы
40. Укажите основные отличия принципов симметричного и асимметричного шифрования
41. Криптографические библиотеки. Краткий обзор и применение.
42. Блочные шрифты. Основные типы, особенности, способы реализации.
43. Принципы построения блочных шрифтов. Область применения
44. Понятия цикловой функции и циклового ключа. Получение и применение циклового ключа.

45. Схема работы электронной кодовой книги – пояснение и особенности.

Самый простой режим работы назван режимом электронной кодовой книги (ECB — ELECTRONIC CODEBOOK). Исходный текст разделен на N блоков. Размер блока — n бит. Этот размер исходного текста не является кратным числом размера блока, текст дополняется, чтобы сделать последний блок по размеру таким же, как другие блоки. Один и тот же ключ используется, чтобы зашифровать и расшифровывать каждый блок.

Соотношение между исходным и зашифрованным текстами:

Шифрование: $C_i = E_K(P_i)$

Дешифрование: $P_i = D_K(C_i)$

Преимущества

- Нет необходимости в последовательном применении функции шифрования к потоку открытого текста. Допустимо сначала зашифровать, например, начало файла, потом конец, потом середину. Как следствие, шифрование может быть параллельным.

Недостатки

- Блоки могут пропадать или появляться. Злоумышленник может перехватить блок и продублировать его, и со стороны приёмника он будет воспринят как «правильный».
- При использовании одного ключа идентичные блоки открытого текста шифруются в идентичные блоки зашифрованного текста; таким образом, этот метод плохо скрывает структуру данных, что также делает его неустойчивым к статистическому анализу. Если шифруемое сообщение содержит два повторяющихся элемента с периодом повторения, кратным размеру блока, то в зашифрованном тексте появится два одинаковых блока. Многие форматы файлов подразумевают использование стандартных

заголовков или наличие блоков одинаковых символов, и шифрование таких файлов приведёт к появлению повторяющихся блоков в шифротексте. Данная особенность режима ECB делает его непригодным для безопасного практического применения в большинстве случаев.[1]

Режим устойчив к ошибкам, связанным с изменением битов блока (ошибка не распространяется на другие блоки), но неустойчив — к ошибкам, связанным с потерей или вставкой битов, если не используется дополнительный механизм выравнивания блоков.

46. Схема CBC – пояснение и особенности

CBC — один из режимов шифрования для симметричного блочного шифра с использованием механизма обратной связи. Каждый блок открытого текста (кроме первого) побитово складывается по модулю 2 (операция XOR) с предыдущим результатом шифрования.

Шифрование может быть описано следующим образом: $C_0 = IV$

$C_i = E_k\left(P_i \oplus C_{i-1}\right)$

где — i номера блоков, — IV вектор инициализации (синхропосылка), C_i и P_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функци яблочного шифрования. Расшифровка:

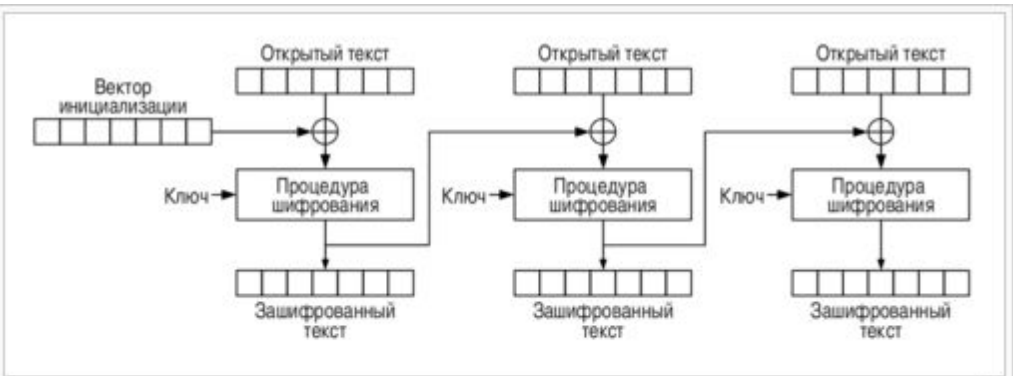
$P_i = C_{i-1} \oplus D_k\left(C_i\right)$

Для преодоления недостатков *ECB* используют способ, при котором одинаковые незашифрованные блоки преобразуются в различные зашифрованные. Для этого в качестве *входа алгоритма* используется результат применения операции *XOR* к текущему незашифрованному блоку и предыдущему зашифрованному блоку.

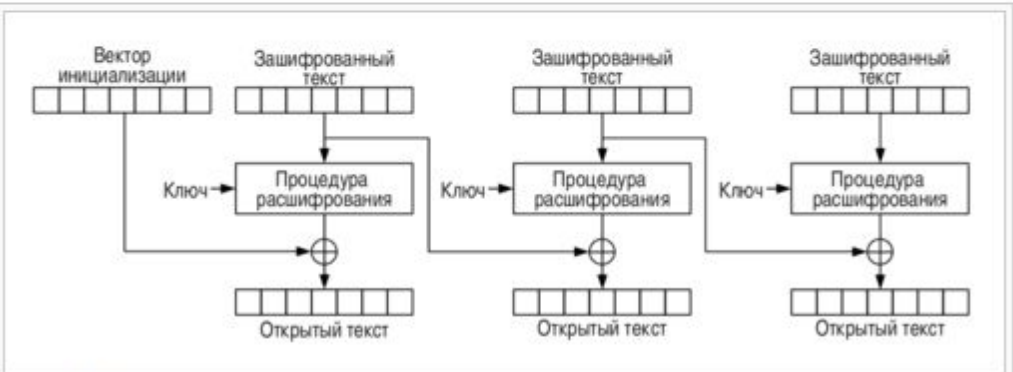
Для получения первого блока зашифрованного сообщения используется

инициализационный вектор (IV), для которого выполняется операция *XOR* с первым блоком незашифрованного сообщения. При дешифровании для IV выполняется операция *XOR* с выходом дешифрирующего алгоритма для получения первого блока незашифрованного текста.

IV должен быть известен как отправителю, так и получателю. Для максимальной безопасности IV должен быть защищен так же, как ключ.



Шифрование в режиме CBC



Расшифрование в режиме CBC

Особенности:

- Наличие механизма распространения ошибки: если при передаче произойдёт изменение одного бита шифротекста, данная ошибка распространится и на следующий блок. Однако на последующие блоки (через один) ошибка не распространится, поэтому режим CBC также называют **самовосстанавливающимся**.
- Неустойчив к ошибкам, связанным с потерей или вставкой битов, если не используется дополнительный механизм выравнивания блоков.
- Злоумышленник имеет возможность добавить блоки к концу зашифрованного сообщения, дополняя тем самым открытый текст (однако без ключа получается мусор)
- Для очень крупных сообщений (32 Гбайта при длине блока 64 бита) всё-таки возможно применение атак, основанных на структурных особенностях открытого текста (следствие парадокса дней рождений).

47. Схема CFB – пояснение и особенности

Режим обратной связи по шифротексту, режим гаммирования с обратной связью (*CFB*) — один из вариантов использования симметричного блочного шифра, при котором для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным (блочным шифром) результатом шифрования предыдущего блока.

$$C_0 = IV$$
$$C_i = E_k\left(C_{i-1}\right) \oplus P_i$$
$$P_i = E_k\left(C_{i-1}\right) \oplus C_i$$

Шифрование может быть описано следующим образом:

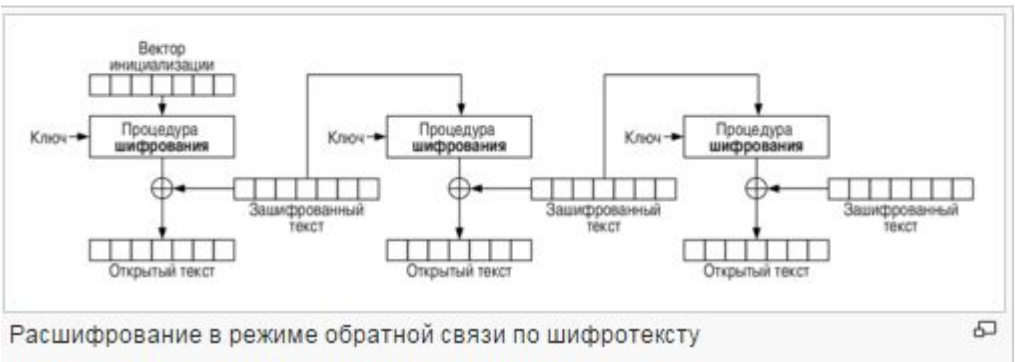
где — i номера блоков, — IV вектор инициализации (синхропосылка), C_i и P_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функция блочного шифрования.

Блочный алгоритм предназначен для шифрования блоков определенной длины. Однако можно преобразовать блочный алгоритм в поточный *алгоритм шифрования*, используя последние два режима. Поточный *алгоритм шифрования* устраняет необходимость разбивать сообщение на целое число блоков достаточно большой длины, следовательно, он может работать в реальном времени. Таким образом, если передается поток символов, каждый символ может шифроваться и передаваться сразу, с использованием символьно ориентированного режима блочного алгоритма шифрования.

Рассмотрим шифрование. Входом функции шифрования является регистр сдвига, который первоначально устанавливается в инициализационный вектор IV. Для левых J битов *выхода алгоритма* выполняется операция *XOR* с первыми J битами незашифрованного текста P1 для получения первого блока зашифрованного текста C_1. Кроме того, содержимое регистра сдвигается влево на J битов, и C_1 помещается в правые J битов этого регистра. Этот процесс продолжается до тех пор, пока не будет зашифровано все сообщение.

При дешифровании используется аналогичная схема, за исключением того, что для блока получаемого зашифрованного текста выполняется операция *XOR* с *выходом алгоритма* для получения незашифрованного блока.

Шифрование в режиме обратной связи по шифротексту



Расшифрование в режиме обратной связи по шифротексту

Особенностью данного режима является распространение ошибки на весь последующий текст. Рекомендованные значения k: 1 <= k <= 8.

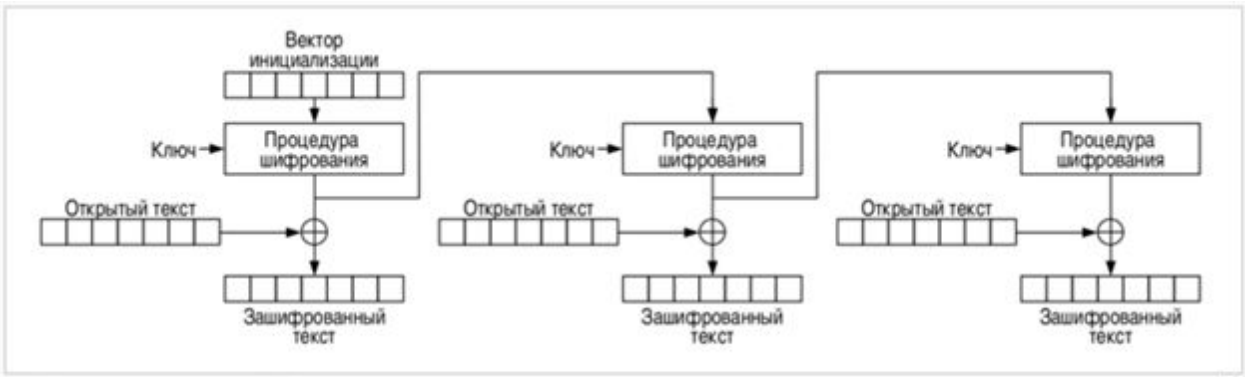
Применяется, как правило, для шифрования потоков информации типа оцифрованной речи, видео.

48. Схема OFB – пояснение и особенности

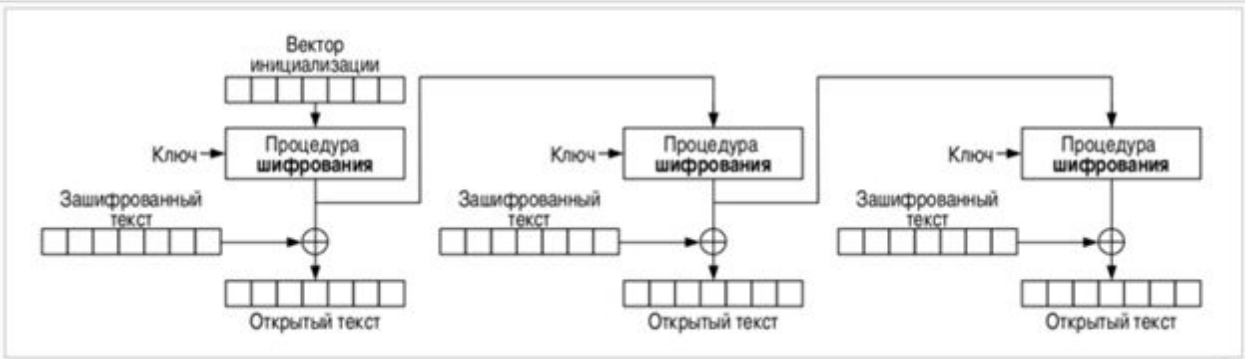
Режим (OFB) обратной связи вывода превращает блочный шифр в синхронный шифр потока: он генерирует ключевые блоки, которые являются результатом сложения с блоками открытого текста, чтобы получить зашифрованный текст. Так же, как с другими шифрами потока, зеркальное отражение в зашифрованном тексте производит зеркально отраженный бит в открытом тексте в том же самом местоположении. Это свойство позволяет многим кодам с исправлением ошибок функционировать как обычно, даже когда исправление ошибок применено перед кодированием.

Из-за симметрии операции сложения, шифрование и расшифрование похожи:

$$C_i = P_i \oplus O_i$$
$$P_i = C_i \oplus O_i$$
$$O_i = E_k(O_{i-1})$$
$$O_0 = IV$$



Шифрование в режиме OFB



Расшифрование в режиме OFB

Каждая операция блочного шифра обратной связи вывода зависит от всех предыдущих и поэтому не может быть выполнена параллельно. Однако, из-за того, что открытый текст или зашифрованный текст используются только для конечного сложения, операции блочного шифра могут быть выполнены заранее, позволяя выполнить заключительное шифрование параллельно с открытым текстом.

Данный метод называется также «режим обратной связи по выходу».

OFB также предполагает некое усовершенствование касающееся метода генерации независимой последовательности блоков: для получения очередного блока предлагается шифровать не с O_i, а с $O_i + IV(mod 2^{64})$, где некоторый вектор инициализации.

Особенности режима

- Значение вектора инициализации должно быть уникальным для каждой процедуры шифрования одним ключом. Его необязательно сохранять в секрете и оно может быть передано вместе с шифротекстом.
- Алгоритм дешифрования в режиме OFB полностью совпадает с алгоритмом шифрования. Функция дешифрования блочного алгоритма не используется в данном режиме, т.к. ключевой поток генерируется только функцией шифрования блока.
- Режим OFB наглядно демонстрирует одну из проблем потоковых шифров.[1] При использовании одного и того же вектора инициализации для шифрования нескольких сообщений будет сгенерирован одинаковый поток ключей. Предположим, что P_1 и P_2 — два разных сообщения для шифрования ключом K. Зашифруем исходные сообщения в режиме OFB и получим два шифротекста — C_1 и C_2, соответственно. Таким образом, будет справедливо следующее тождество:
- $C_1 \oplus C_2 = E(K, K_{i-1}) \oplus P_1 \oplus E(K, K_{i-1}) \oplus P_2 = P_1 \oplus P_2$
- Следовательно, если потенциальному злоумышленнику известна хотя бы одна пара шифрованного и открытого текста, вычисление любых открытых текстов, зашифрованных таким же ключом и с идентичным вектором инициализации, становится тривиальной задачей.
- Появление коллизии в ключевом потоке (или совпадение вектора инициализации и одного из ключевых блоков) приведёт к циклическому повторению ключевой последовательности, что может вызвать нарушение безопасности режима шифрования, как показано в предыдущем пункте.
- Распространение ошибки в данном режиме не происходит. Изменение одного бита в шифрованном тексте приведет к изменению одного бита при дешифровании. Однако, потеря бита в шифротексте приведет к некорректному дешифрованию всех последующих битов.

49. Алгоритм RSA. Этапы работы алгоритма RSA и область его применения.

Применение RSA

Система RSA используется для защиты программного обеспечения и в схемах цифровой подписи.

Также она используется в открытой системе шифрования PGP и иных системах шифрования (к примеру, DarkCryptTC и формат xdc) в сочетании с симметричными алгоритмами.

Из-за низкой скорости шифрования (около 30 кбит/с при 512 битном ключе на процессоре 2 ГГц), сообщения обычно шифруют с помощью более производительных симметричных алгоритмов со случайным *сеансовым ключом* (например, AES, IDEA, Serpent, Twofish), а с помощью RSA шифруют лишь этот ключ, таким образом реализуется гибридная криптосистема. Такой механизм имеет потенциальные уязвимости ввиду необходимости использовать криптографически стойкий генератор псевдослучайных чисел для формирования случайного сеансового ключа симметричного шифрования.

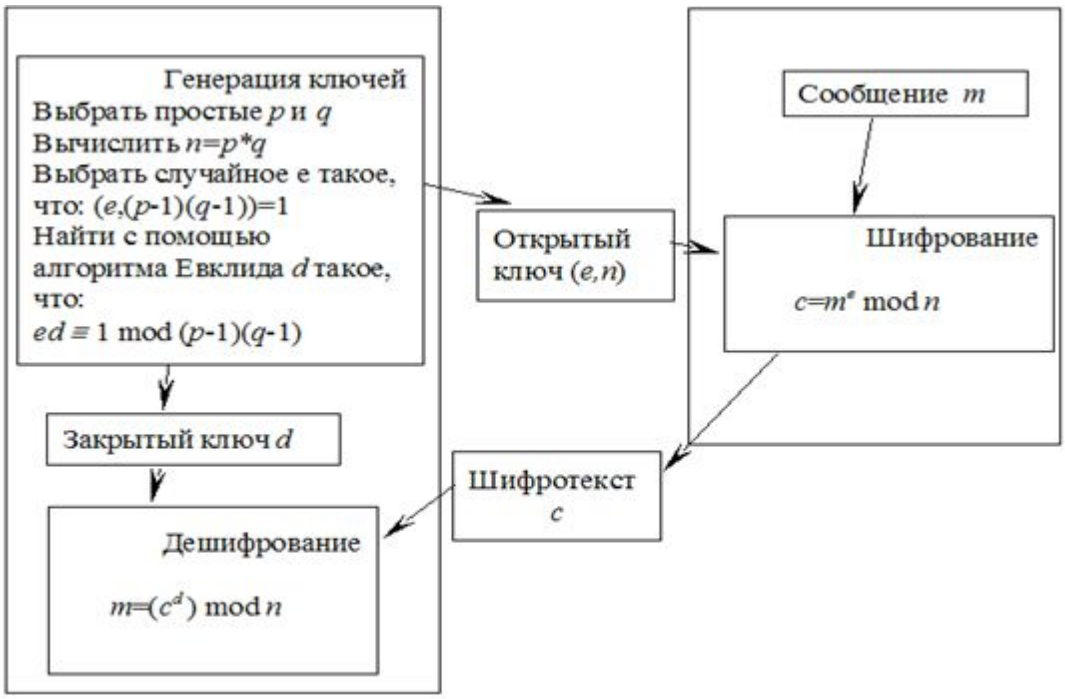


Рис.2.15. Схема шифрования алгоритма RSA

50. Алгоритм формирования ключей для алгоритма RSA

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
2. Вычисляется их произведение $n = p \cdot q$, которое называется *модулем*.
3. Вычисляется значение функции Эйлера от числа n : $\varphi(n) = (p-1) \cdot (q-1)$.
4. Выбирается целое число $\varphi(n)$ ($1 < e < \varphi(n)$), взаимно простое со значением функции . Обычно в качестве берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые числа Ферма 17, 257 или 65537.

· Число e называется *открытой экспонентой* (англ. *public exponent*)

· Время, необходимое для шифрования с использованием быстрого возведения в степень, пропорционально числу единичных бит в e .

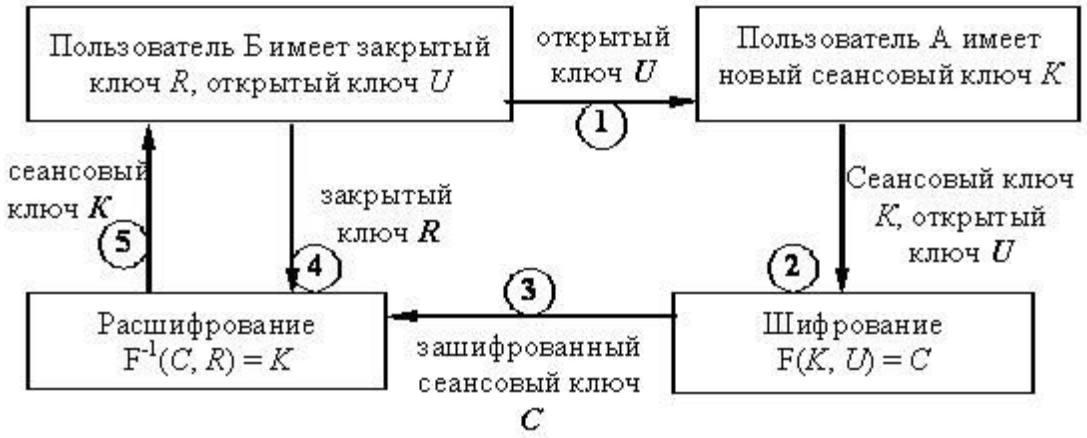
· Слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA

5. Вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению: $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

·Число d называется *секретной экспонентой*. Обычно, оно вычисляется при помощи расширенного алгоритма Евклида.

6.Пара $\{e, n\}$ публикуется в качестве *открытого ключа RSA* (англ. *RSA public key*).

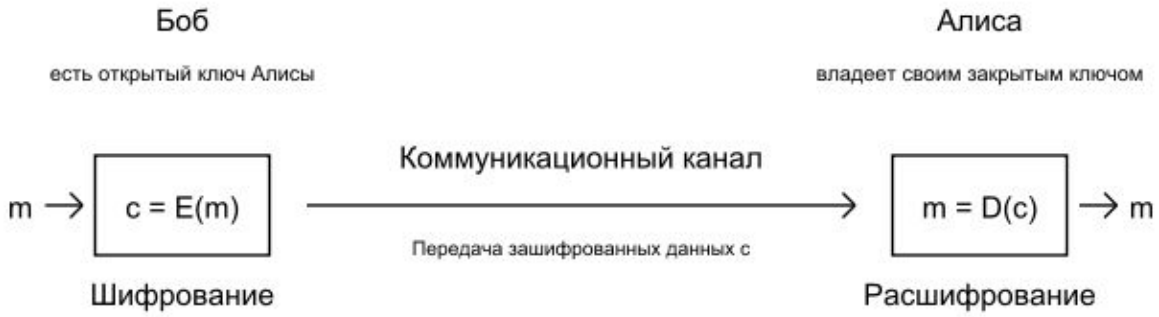
7. Пара $\{d, n\}$ играет роль *закрытого ключа RSA* (англ. *RSA private key*) и держится в секрете.



51. Алгоритм RSA. Шифрование сообщения по алгоритму RSA

Предположим, Боб хочет послать Алисе сообщение m.

Сообщениями являются целые числа в интервале от 0 до $n-1$, т.е $m \in \mathbb{Z}_n$.



Алгоритм:

Взять *открытый ключ* Алисы (e, n)

Взять *открытый текст* m

Зашифровать сообщение с использованием открытого ключа Алисы:

$$c = E(m) = m^e \mod n \quad (1)$$

Алгоритм:

· Принять зашифрованное сообщение c

· Взять свой *закрытый ключ* (d, n)

· Применить закрытый ключ для расшифрования сообщения:

$$m = D(c) = c^d \mod n \quad (2)$$

Данная схема на практике не используется по причине того, что она не является *практически надёжной*(semantically secured). Действительно, односторонняя функция E(m) является *детерминированной* — при одних и тех же значениях входных параметров

(ключа и сообщения) выдаёт одинаковый результат —, а это значит, что не выполняется необходимое условие практической (семантической) надёжности шифра.

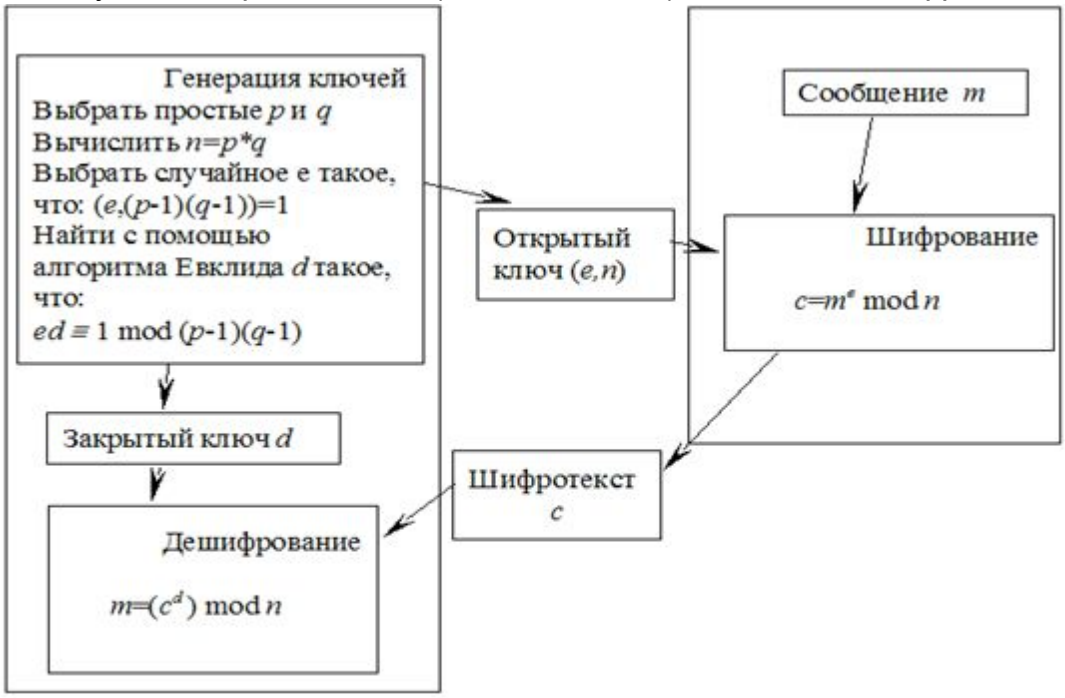


Рис.2.15. Схема шифрования алгоритма RSA

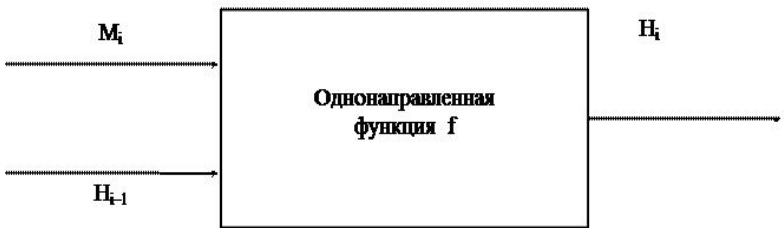
52. Алгоритм RSA. Восстановление сообщения по алгоритму RSA

см выше

Л10-Хэш-функции

53. Однонаправленные Хэш-функции. Проблемы и способы решения

Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция h(·) принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение h(M)=H фиксированной длины. Обычно хэшированная информация является сжатым двоичным представлением основного сообщения произвольной длины.



Для того, чтобы хэш-функция H считалась криптографически стойкой, она должна удовлетворять трем основным требованиям, на которых основано большинство применений хэш-функций в криптографии:

Необратимость или стойкость к восстановлению прообраза: для заданного значения хэш-функции m должно быть вычислительно невозможно найти блок данных X, для которого H(X)=m.

Стойкость к коллизиям первого рода или восстановлению вторых прообразов: для заданного сообщения M должно быть вычислительно невозможно подобрать другое сообщение N, для которого H(N)=H(M).

Стойкость к коллизиям второго рода: должно быть вычислительно невозможно подобрать пару сообщений (M, M'), имеющих одинаковый хэш.

Данные требования не являются независимыми:

Обратимая функция нестойка к коллизиям первого и второго рода.

Функция, нестойкая к коллизиям первого рода, нестойка к коллизиям второго рода; обратное неверно.

Следует отметить, что не доказано существование необратимых хэш-функций, для которых вычисление какого-либо прообраза заданного значения хэш-функции теоретически невозможно. Обычно нахождение обратного значения является лишь вычислительно сложной задачей.

Для криптографических хэш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось (лавинный эффект). В частности, значение хэша не должно давать утечки информации даже об отдельных битах аргумента. Это требование является залогом криптостойкости алгоритмов хэширования пользовательских паролей для получения ключей.

54. Структура и алгоритмы MD-функций. Функция MD4.

MD2(The MD2 Message Digest Algorithm)- хэш-функция , разработанная Бертом Калиски (RSA Laboratories) в 1992 году, и описанная в RFC 1319. Размер хэша - 128 бит. Размер блока входных данных - 512 бит. Быстродействие (220 байт/с) - 0.709.

Безопасность MD2 опирается на случайную фиксированую перестановку битов.

Хэширование сообщения заключается в следующем:

1. Дополняем сообщение i байтами, причем значение i должно быть таким, чтобы длина полученного сообщения была кратна 16 байтам

2. Добавляем к сообщению 16 байтов контрольной суммы.

3. Инициализируем 48-байтный блок: X[0],X[1],X[2],...X[47]. Заполняем первые 16 байтов X нулями, во вторые 16 байтов X копируем первые 16 байтов сообщения, в третьи 16 байтов X должны быть равны XOR первых и вторых 16 байтов X.

4 Функция сжатия:

```
t = 0
for j=0 to 17
  for k = 0 to 47
    t = X XOR S
    X=t
    t=(t+j)mod256
```

5. Копируем во вторые 16 байтов X вторые 16 байтов сообщения, а в третьи 16 байтов X должны быть равны XOR первых и творых 16 байтов X. Выполнить этап 4. Повторять этапы 5 и 4 по очереди для каждых 16 байт сообщения.

6. Выходом являются первые 16 байтов X

55. Структура и алгоритмы MD-функций. Функция MD5.

MD5 (англ. Message Digest 5) — 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского Технологического Института (MIT, Massachusetts Institute of Technlogy) в 1991 году. Предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины. Пришёл на смену MD4, который был несовершенен. Работает на 32-битных машинах. Зная MD5 невозможно восстановить входное сообщение, так как разным сообщениям может соответствовать один MD5. Используется для проверки подлинности опубликованных сообщений, путем сравнения дайджеста сообщения с опубликованным.

На вход алгоритма поступает входной поток данных, хеш которого необходимо найти. Длина сообщения может быть любой (в том числе нулевой). Запишем длину сообщения в L. Это число целое и не отрицательное. Кратность каким-либо числам не обязательна. После поступления даных идет процесс подготовки потока к вычислениям.

Ниже приведены 5 шагов алгоритма:

Шаг 1. Выравнивание потока

Входные данные выравниваются так, чтобы их размер был сравним с 448 по модулю 512 (L’ = 512 × N + 448). Сначала дописывают единичный бит в конец потока, затем необходимое число нулевых бит (выравнивание происходит, даже если длина уже конгруэнтна — сравнима с 448).

Шаг 2. Добавление длины сообщения

В оставшиеся 64 бита дописывают 64-битное представление длины данных до выравнивания. Если длина превосходит 2^{64}-1, то дописывают только младшие биты. После этого длина потока станет кратной степеням двойки — 16, 32. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит.

Шаг 3. Инициализация буфера

Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения шестнадцатеричными числами:

A = 01 23 45 67;

B = 89 AB CD EF;

C = FE DC BA 98;

D = 76 54 32 10.

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором.

Определим еще функции и константы, которые нам понадобятся для вычислений.

Потребуются 4 функции для четырех раундов. Введем функции от трех параметров — слов, результатом также будет слово.

1 раунд $FunF(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$

2 раунд $FunG(X,Y,Z) = (X \wedge Z) \vee (\neg Z \wedge Y)$

3 раунд $FunH(X,Y,Z) = X \oplus Y \oplus Z$

4 раунд $FunI(X,Y,Z) = Y \oplus (\neg Z \vee X)$

Определим таблицу констант T[1..64] — 64-элементная таблица данных, построенная следующим образом: T[i]=int(4294967296*|sin(i)|) и s — циклический сдвиг влево на s бит полученого 32-битного аргумента.

Выравненные данные разбиваются на блоки (слова) по 32 бита, и каждый блок проходит 4 раунда из 16 операторов. Все операторы однотипны и имеют вид: [abcd k s i], определяемый как a = b + ((a + Fun(b, c, d) + X[k] + T[i]) <<< s), где X — блок данных. X[k] = M [n * 16 + k], где k — номер 32-битного слова из n-го 512-битного блока сообщения.

Шаг 4. Вычисление в цикле

Заносим в блок данных элемент n из массива. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый).

AA = A

BB = B

CC = C

DD = D

Раунд 1

/*[abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]
[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

Раунд 2

/*[abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]
[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]
[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]
[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

Раунд 3

/*[abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]
[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]
[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]
[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

Раунд 4

/*[abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]
[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]
[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]
[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

Суммируем с результатом предыдущего цикла:

A = AA + A
B = BB + B
C = CC + C
D = DD + D

После окончания цикла необходимо проверить, есть ли еще блоки для вычислений. Если да, то изменяем номер элемента массива (n++) и переходим в начало цикла.

Шаг 5. Результат вычислений

Результат вычислений находится в буфере ABCD, это и есть хеш. Если вывести слова в обратном порядке DCBA, то мы получим наш MD5 хеш.

56. Структура и алгоритмы MD-функций. Функция MD2.

MD4 (Message Digest 4) — хеш-функция, разработанная профессором Массачусетского университета Рональдом Ривестом в 1990 году, и впервые описанная в RFC 1186. Для произвольного входного сообщения функция генерирует 128-разрядное хеш-значение, называемое дайджестом сообщения. Этот алгоритм используется в протоколе аутентификации MS-CHAP, разработанном корпорацией Майкрософт для выполнения процедур проверки подлинности удаленных рабочих станций Windows.

Предполагается, что на вход подано сообщение, состоящее из b бит, хеш которого нам предстоит вычислить. Здесь b — произвольное неотрицательное целое число; оно может быть нулем, не обязано быть кратным восьми, и может быть сколь угодно большим. Запишем сообщение побитово, в виде $m_0 m_1 .. m_{b-1}$:

Ниже приведены 5 шагов, используемые для вычисления хеша сообщения.

Шаг 1. Добавление недостающих битов.

Сообщение расширяется так, чтобы его длина в битах по модулю 512 равнялась 448. Таким образом, в результате расширения, сообщению недостает 64 бита до длины, кратной 512 битам. Расширение производится всегда, даже если сообщение изначально имеет нужную длину.

Расширение производится следующим образом: один бит, равный 1, добавляется к сообщению, а затем добавляются биты, равные 0, до тех пор, пока длина сообщения не

станет равной 448 по модулю 512. В итоге, к сообщению добавляется как минимум 1 бит, и как максимум 512.

Шаг 2. Добавление длины сообщения.

64-битное представление b (длины сообщения перед добавлением набивочных битов) добавляется к результату предыдущего шага. В маловероятном случае, когда b больше, чем 2^{64}, используются только 64 младших бита. Эти биты добавляются в виде двух 32-битных слов, и первым добавляется слово, содержащее младшие разряды.

На этом этапе (после добавления битов и длины сообщения) мы получаем сообщение длиной кратной 512 битам. Это эквивалентно тому, что это сообщение имеет длину, кратную 16-ти 32-битным словам. Пусть M[0…N-1] означает массив слов получившегося сообщения (здесь N кратно 16).

Шаг 3. Инициализация MD-буфера.

Для вычисления хеша сообщения используется буфер, состоящий из 4 слов (32-битных регистров): (A,B,C,D). Эти регистры инициализируются следующими шестнадцатеричными числами (младшие байты сначала):

word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

Шаг 4. Обработка сообщения блоками по 16 слов.

Для начала определим три вспомогательные функции, каждая из которых получает на вход три 32-битных слова, и по ним вычисляет одно 32-битное слово.

$$F(X,Y,Z) = XY \vee \neg X\,Z$$
$$G(X,Y,Z) = XY \vee XZ \vee YZ$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$

На каждую битовую позицию F действует как условное выражение: если X, то Y; иначе Z. Функция F могла бы быть определена с использованием + вместо V, поскольку XY и ⌐XZ не могут равняться 1 одновременно. G действует на каждую битовую позицию как функция максимального значения: если по крайней мере в двух словах из X, Y, Z соответствующие биты равны 1, то G выдаст 1 в этом бите, а иначе G выдаст бит, равный 0. Интересно отметить, что если биты X, Y и Z статистически независимы, то биты F(X,Y,Z) и G(X,Y,Z) будут также статистически независимы. Функция H реализует побитовый xor, она обладает таким же свойством, как F и G.

Шаг 5. Формирование хеша.

Результат (хеш-функция) получается как ABCD. То есть, мы выписываем 128 бит, начиная с младшего бита A, и заканчивая старшим битом D.

57. Структура и алгоритмы MD-функций. Функция SHA.

Secure Hash Algorithm 1 — алгоритм криптографического хеширования. Описан в RFC 3174. Для входного сообщения произвольной длины (максимум 2^{64}-1 бит) алгоритм генерирует 160-битное хеш-значение, называемое также дайджестом сообщения. Используется во многих криптографических приложениях и протоколах.

SHA-1 реализует однонаправленную хеш-функцию, простроенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш блока M_i равен $h_i = f(M_i, h_{i-1})$. Хеш-значением всего сообщения является выход последнего блока.

Исходное сообщение разбивается на блоки по 512 бит в каждом. Последний блок дополняется до длины, кратной 512 бит. Сначала добавляется 1 а потом нули, чтобы

длина блока стала равной 512 — 64 бит. В оставшиеся 64 бита записывается длина исходного сообщения. Дополнение последнего блока осуществляется всегда, даже если сообщение уже имеет нужную длину. Таким образом, число добавляемых битов находится в диапазоне от 1 до 512.

Инициализируются пять 32-битовых переменных.

A = a = 0x67452301
B = b = 0xEFCDAB89
C = c = 0x98BADCFE
D = d = 0x10325476
E = e = 0xC3D2E1F0

Определяются четыре нелинейные операции и четыре константы.

$F_t(m,l,k) = (m \wedge l) \vee (\neg m \vee k)$	K_t = 0x5A827999	0≤t≤19
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0x6ED9EBA1	20≤t≤39
$F_t(m,l,k) = (m \wedge l) \vee (m \wedge k) \vee (l \wedge k)$	K_t = 0x8F1BBCDC	40≤t≤59
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0xCA62C1D6	60≤t≤79

Главный цикл

Главный цикл итеративно обрабатывает каждый 512-битный блок. Итерация состоит из четырех этапов по двадцать операций в каждом. Блок сообщения преобразуется из 16 32-битовых слов M_i в 80 32-битовых слов W_i по следующему правилу:

$$W_t = M_t \qquad \text{при } 0 \le t \le 15$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) << 1 \qquad \text{при } 16 \le t \le 79$$

здесь << – это циклический сдвиг влево

```
для t от 0 до 79
        temp = (a<<5) + F_t(b,c,d) + e + W_t + K_t
        e = d
        d = c
        c = b<<30
        b = a
        a = temp
```

После этого a, b, c, d, e прибавляются к A, B, C , D , E соответственно. Начинается следующая итерация.

Итоговым значением будет объединение пяти 32-битовых слов в одно 160-битное хеш-значение.

58. Область применения и назначение Хэш-функций

Хеш-функцией (hash function) называется математическая или иная функция, которая для строки произвольной длины вычисляет некоторое целое значение или некоторую другую строку фиксированной длины. Математически это можно записать так:

h=H(M),

где M – исходное сообщение, называемое иногда прообразом, а h – результат, называемый значением хеш-функции (а также хеш-кодом или дайджестом сообщения (от англ. message digest)).

Смысл хеш-функции состоит в определении характерного признака прообраза – значения хеш-функции. Это значение обычно имеет определенный фиксированный размер, например, 64 или 128 бит. Хеш-код может быть в дальнейшем проанализирован для решения какой-либо задачи.

Областью применения хэш функций является электронная подпись и проверка парольной фразы.

Электронная подпись (ЭП) — по сути шифрование сообщения алгоритмом с открытым ключом. Текст, зашифрованный секретным ключом, объединяется с исходным сообщением. Тогда проверка подписи — расшифрование открытым ключом; если получившийся текст аналогичен исходному тексту — подпись верна.

Использование хеш-функции позволяет оптимизировать данный алгоритм. Производится шифрование не самого сообщения, а значение хеш-функции, взятой от сообщения. Данный метод обеспечивает следующие преимущества:

·Понижение вычислительной сложности. Как правило, документ значительно больше его хеша.

· Повышение криптостойкости. Криптоаналитик не может, используя открытый ключ, подобрать подпись под сообщение, а только под его хеш.

· Обеспечение совместимости. Большинство алгоритмов оперирует со строками бит данных, но некоторые используют другие представления. Хеш-функцию можно использовать для преобразования произвольного входного текста в подходящий формат.

В большинстве случаев **парольные фразы** не хранятся на целевых объектах, хранятся лишь их хеш-значения. Хранить парольные фразы нецелесообразно, так как в случае несанкционированного доступа к файлу с фразами злоумышленник узнает все парольные фразы и сразу сможет ими воспользоваться, а при хранении хеш-значений он узнает лишь хеш-значения, которые не обратимы в исходные данные, в данном случае в парольную фразу. В ходе процедуры аутентификации вычисляется хеш-значение введённой парольной фразы, и сравнивается с сохранённым.

Л11- Политики Информационной безопасности

59. Понятие «Политика информационной безопасности». Сферы действия политик информационной безопасности

Уровни Информационной безопасности

Законодательный	Административный	Процедурный
Законы и нормативные акты, для всех субъектов информационных отношений независимо от их организационной принадлежности	Направлен на всех субъектов информационных отношений в пределах организации. Это действие общего характера, предпринимаемые руководством организации.	Направлен на оборудование и программное обеспечение

Политика информационной безопасности - совокупность руководящих принципов, правил, процедур и практических приемов в области безопасности, которые регулируют управление, защиту и распределение ценной информации.

Политика безопасности зависит:

- от конкретной технологии обработки информации;
- от используемых технических и программных средств;
- от расположения организации;

Политика ИБ строится на основе анализа рисков ИБ, которые признается реальными для данной организации.

Сферы действия ПИБ:

- аппаратно-программные средства;

- информационные ресурсы;
- локальная сеть предприятия.

60. Политика информационной безопасности. Уровни политик безопасности и их цели

Уровни политик информационной безопаности

Верхний	Средний	Нижний
Решения затрагивающие организацию в целом. Формируются руководством. Независим от технологий.	Решение вопросов, касающихся отдельных вопросов ИБ, важных для различных систем.	Конкретные сервисы. Цели и правила их достижения
	доступ в Инт. Дом ПК.	Конкретная реализация ПИБ

Цели верхнего уровня:

- управление рисками и ресурсами безопасности;
- координация использования этих ресурсов;
- выделение специального персонала для защиты критически важных систем;
- поддержание контактов с организациями, обеспечивающими или контролирующими режимы безопасности.

Аспекты Верхнего уровня:

1. соблюдение существующих законов
2. контроль действия лиц, ответственных за разработку программ безопасности
3. обеспечение исполнительской дисциплины с помощью системы поощрений и наказаний (контроль деятельности в области ИБ).

В рамках программы верхнего уровня принимаются стратегически важные решения по обеспечению ИБ.

Пример задач Верхнего уровня:

- сформировать или пересмотреть комплексную программу обеспечения ИБ;
- назначить ответственных за продвижение программы;
- сформулировать цели, которые преследует организация в области ИБ;
- определить общие направления ИБ;
- обеспечить базу для соблюдения законов и правил;
- сформулировать административные решения по тем вопросам реализации программы безопасности, которые должны рассматриваться на уровне организации в целом.

Пример задач Среднего уровня:

- отношение у передовым (но недостаточно проверенным) технологиям;
- доступ в Интернет;
- использование домашних компьютеров;
- применение нелицензионного ПО;

Пример задач Нижнего уровня:

- кто имеет право доступа к объектам поддерживаемым сервисом;
- при каких условиях можно читать и модифицировать данные;
- как организовать удаленный доступ к сервису.

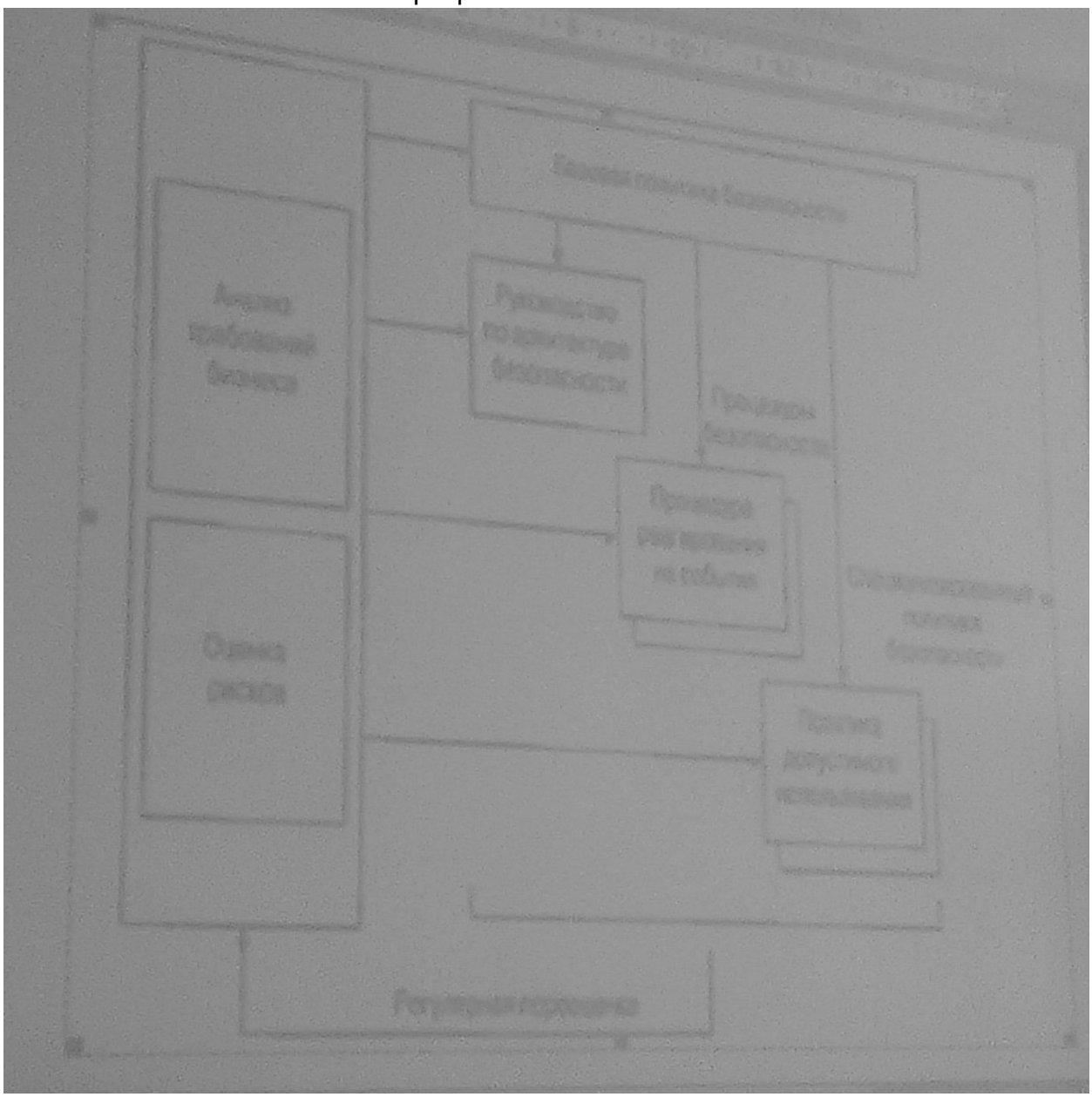
На этом уровне решается, какие следует использовать механизмы защиты, закупаются и устанавливаются технические средства, выполняется повседневное администрирование, антивирусная проверка и т.п.

61. Политика информационной безопасности. Порядок построения политики ИБ организации

Порядок построения политики ИБ организации

- составление карты информационной системы
- анализ рисков
- определение стратегии защиты
- составление плана организации ИБ
- выделение ресурсов
- назначение ответственных
- определение мер контроля за соблюдением Политик

схема разработки политики безопасности



62. Охарактеризуйте методы исследования ИБ предприятия

методы исследования ИБ предприятия

исследование сверху вниз	Исследование снизу вверх

Детальный анализ всей существующей схемы хранения и обработки информации	Поочередная проверка возможности осуществления различных видов атак

Этапы исследования сверху вниз:

1. Определить, какие информационные объекты и потоки необходимо защищать
2. изучение текущего состояния системы ИБ (что из классических методик защиты информации уже реализовано и на каком уровне)
3. классификация всех информационных объектов на классы в соответствии с требованиями к конфиденциальности, доступности и целостности.
4. Вычисление рисков (Предполагаемого ущерба)
5. Анализ таблицы рисков предприятия:

Y - Уровень допустимого риска
X - интегральный риск
Z - двойной риск (Z=Y^2) (???)
Pi - i-я политика безопасности
Ri - вычисленный риск по i-й строке.
Если Ri<Y - необходимо ужесточение (разработка) Pi
Если X>Z - система безопасности в целом не работает =>
найти max Ri - изменить Pi => повторить с П.4

63. Перечислите и охарактеризуйте компоненты архитектуры безопасности

Компоненты архитектуры безопасности
Физическая безопасность:

- центральные процессоры и системные блоки
- компоненты инфраструктуры локальной системы свызанные с LAN
- медиа - память.

Логическая безопасность:
средства безопасности осуществляющие идентификацию и аутентификацию пользователей, управление доступом, межсетевое экранирование, аудит и мониторинг сети, управление удаленным доступом и т.д.
Защита ресурсов: файлы, БД, программы, данные.
Полномочия:

- полномочия системного администратора;
- полномочия администратора безопасности

Роли и ответственности в безопасности сети:

- провайдер
- менеджер данных
- аудитор
- администратор безопасности
- пользователь

Л12. Программный комплекс анализа и контроля рисков информационной безопасности «ГРИФ»

64. Инструментальные средства анализа рисков информационной безопасности. Назначение и особенности Digital Security Office 2006

Наиболее известные автоматизированные средства по анализу рисков ИБ:
Cobra, CRAMM, Risk Advisor, Risk Watch, BCM-Analyser, АванГард, Digital Secutity Office 2006

Digital Security Office 2006

Digital Secutity Office 2006 - система управления информационными рисками и оценками соответствия системы управления ИБ международным, национальным и корпоративным стандартам в области информационной безопасности.

Система делится на две подсистемы система ГРИФ КОНДОР.

Система ГРИФ позволяет построить приближённую модель информационной системы, содержащую наиболее критичные ресурсы и основные угрозы и уязвимости, с учётом вероятности их реализации.

Подсистема КОНДОР - используется для оценки соответствия системы управления ИБ требованиям стандартов

Задачи системы “ГРИФ”

- оценка защищённости информационной системы
- формирование бюджета ИБ
- Обоснование необходимости инвестиций в информационную безопасность компании

65. Этапы работы и возможности системы ГРИФ

Этап 1: Определение полного списка информационных ресурсов, представляющих ценность для компании.

Этап 2: Ввод в систему ГРИФ всех видов информации, представляющей ценность для компании. Введенные группы ценной информации должны быть размещены пользователем на ранее указанных на предыдущем этапе объектах хранения информации (серверах, рабочих станциях и т.д.). Заключительная фаза – указание ущерба по каждой группе ценной информации, расположенной на соответствующих ресурсах, по всем видам угроз.

Этап 3: Вначале проходит определение всех видов пользовательских групп (и число пользователей в каждой группе). Затем определяется, к каким группам информации на ресурсах имеет доступ каждая из групп пользователей. В заключение определяются виды (локальный и/или удаленный) и права (чтение, запись, удаление) доступа пользователей ко всем ресурсам, содержащих ценную информацию.

Этап 4: Определение средств защиты информации, которыми защищена ценная информация на ресурсах. Кроме того, в систему вводится информация о разовых затратах на приобретение всех применяющихся средств защиты информации и ежегодные затраты на их техническую поддержку, а также - ежегодные затраты на сопровождение системы информационной безопасности компании

Этап 5: Определение соответствия политик безопасности, реализованных в системе, оценка реального уровня защищённости, предложение контрмер и оценка их эффективности.

Результат: Сформирована полная модель информационной системы с точки зрения информационной безопасности с учетом реального выполнения требований комплексной политики безопасности,

66. Преимущества и недостатки Digital Security Office 2006

Преимущества ГРИФ:

- задание ущерба введётся следствие нарушения свойств активов, что значительно уменьшает возможность что один и тот же ущерб может быть учтен несколько раз
- Возможность определения рисков по модели информационной системы
- возможность задавать собственные уязвимости и угрозы, опираясь на заложенную в программу классификацию.
- не требует специальной подготовки и высокой квалификации аудитора и имеют русскоязычный интерфейс. При этом стоимость минимальной лицензии на одну рабочую станции у ГРИФ гораздо ниже, чем у других систем.

К недостаткам ГРИФ можно отнести:

- отсутствие привязки к бизнесс-процессам (запланировано в следующей версии)
- нет возможности сравнения отчетов на разных этапах внедрения комплекса мер по обеспечению защищенности (запланировано в следующей версии)
- отсутствует возможность добавить специфичные для данной компании требования политики безопасности

67. Перечень и особенности отчетов системы ГРИФ

Отчёт по системе

Первая часть - "Информационные риски Ресурсов:

- Информационные риски ресурсов ПО Информации
- Информационные риски ресурсов по классу угроз,
- Суммарные риски по ресурсам
- Информационные риски системы по классу угроз

Вторая часть - "Соотношение ущерба и риска:

- Ущерб по ресурсам - по информации и классу угроз
- Ущерб по ресурсам - суммарный ущерб по ресурсам
- Ущерб и Риск системы - по классу угроз
- Ущерб и Риск системы
- Риск системы и затраты на обеспечение ИБ системы

Третья часть - "Общий вывод о существуюцю рисках информационной системы":

- Максимальные значения риска и ущерба
- Недостатки существующей политики безопасности

Темы задач к КР-2

1. Описать и отобразить графически схему работы электронной кодовой книги

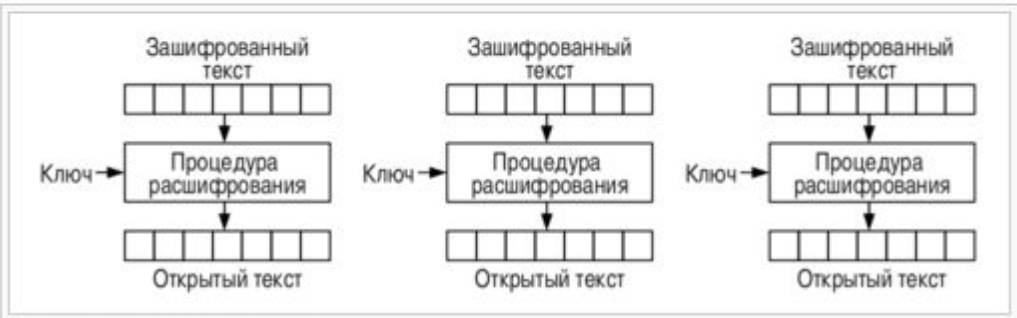
Самый простой режим работы назван режимом электронной кодовой книги (ECB — ELECTRONIC CODEBOOK). Исходный текст разделен на N блоков. Размер блока — n бит. Этот размер исходного текста не является кратным числом размера блока, текст дополняется, чтобы сделать последний блок по размеру таким же, как другие блоки. Один и тот же ключ используется, чтобы зашифровать и расшифровывать каждый блок.

Соотношение между исходным и зашифрованным текстами:

Шифрование: $C_i = E_K(P_i)$

Дешифрование: $P_i = D_K(C_i)$

Шифрование в режиме ECB



Расшифрование в режиме ECB

2. Описать и отобразить графически алгоритм CBC

CBC — один из режимов шифрования для симметричного блочного шифра с использованием механизма обратной связи. Каждый блок открытого текста (кроме первого) побитово складывается по модулю 2 (операция XOR) с предыдущим результатом шифрования.

Шифрование может быть описано следующим образом: $C_0 = IV$

$C_i = E_k\left(P_i \oplus C_{i-1}\right)$

где — i номера блоков, — IV вектор инициализации (синхропосылка), C_i и P_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функци яблочного шифрования. Расшифровка:

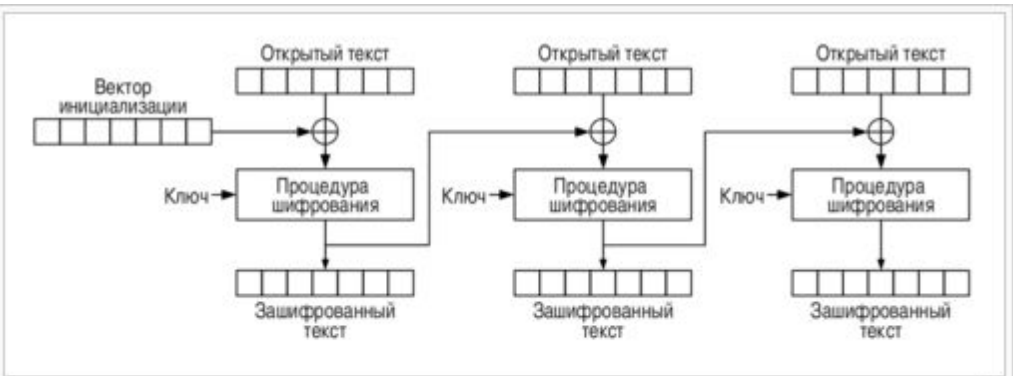
$P_i = C_{i-1} \oplus D_k\left(C_i\right)$

Для преодоления недостатков *ECB* используют способ, при котором одинаковые незашифрованные блоки преобразуются в различные зашифрованные. Для этого в качестве *входа алгоритма* используется результат применения операции *XOR* к текущему незашифрованному блоку и предыдущему зашифрованному блоку.

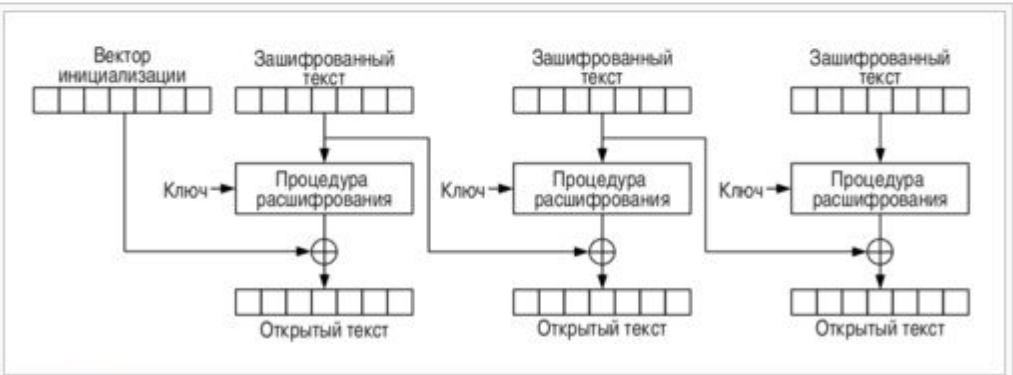
Для получения первого блока зашифрованного сообщения используется

инициализационный вектор (IV), для которого выполняется операция *XOR* с первым блоком незашифрованного сообщения. При дешифровании для IV выполняется операция *XOR* с выходом дешифрирующего алгоритма для получения первого блока незашифрованного текста.

IV должен быть известен как отправителю, так и получателю. Для максимальной безопасности IV должен быть защищен так же, как ключ.



Шифрование в режиме CBC



Расшифрование в режиме CBC

3. Описать и отобразить графически алгоритм CFB

Режим обратной связи по шифротексту, режим гаммирования с обратной связью (*CFB*) — один из вариантов использования симметричного блочного шифра, при котором для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным (блочным шифром) результатом шифрования предыдущего блока.

$$C_0 = IV$$
$$C_i = E_k\left(C_{i-1}\right) \oplus P_i$$
$$P_i = E_k\left(C_{i-1}\right) \oplus C_i$$

Шифрование может быть описано следующим образом:

где — i номера блоков, — IV вектор инициализации (синхропосылка), C_i и P_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функция блочного шифрования.

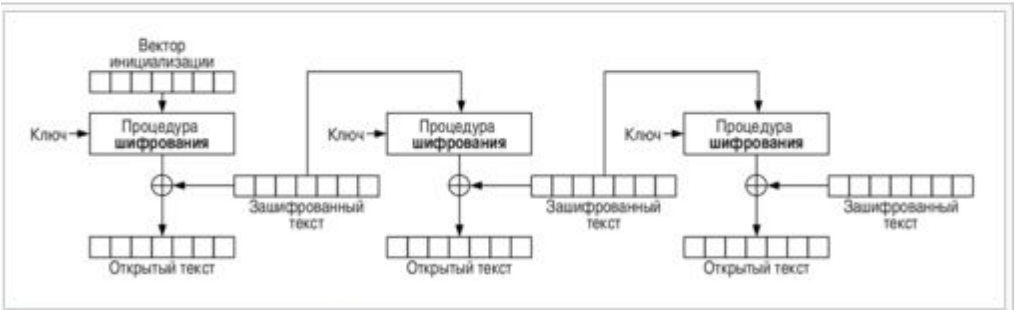
Блочный алгоритм предназначен для шифрования блоков определенной длины. Однако можно преобразовать блочный алгоритм в поточный *алгоритм шифрования*, используя последние два режима. Поточный *алгоритм шифрования* устраняет необходимость разбивать сообщение на целое число блоков достаточно большой длины, следовательно, он может работать в реальном времени. Таким образом, если передается поток символов, каждый символ может шифроваться и передаваться сразу, с использованием символьно ориентированного режима блочного алгоритма шифрования.

Рассмотрим шифрование. Входом функции шифрования является регистр сдвига, который первоначально устанавливается в инициализационный вектор IV. Для левых J битов *выхода алгоритма* выполняется операция *XOR* с первыми J битами

незашифрованного текста P1 для получения первого блока зашифрованного текста C_1. Кроме того, содержимое регистра сдвигается влево на J битов, и C_1 помещается в правые J битов этого регистра. Этот процесс продолжается до тех пор, пока не будет зашифровано все сообщение.

При дешифровании используется аналогичная схема, за исключением того, что для блока получаемого зашифрованного текста выполняется операция *XOR* с *выходом алгоритма* для получения незашифрованного блока.

Шифрование в режиме обратной связи по шифротексту



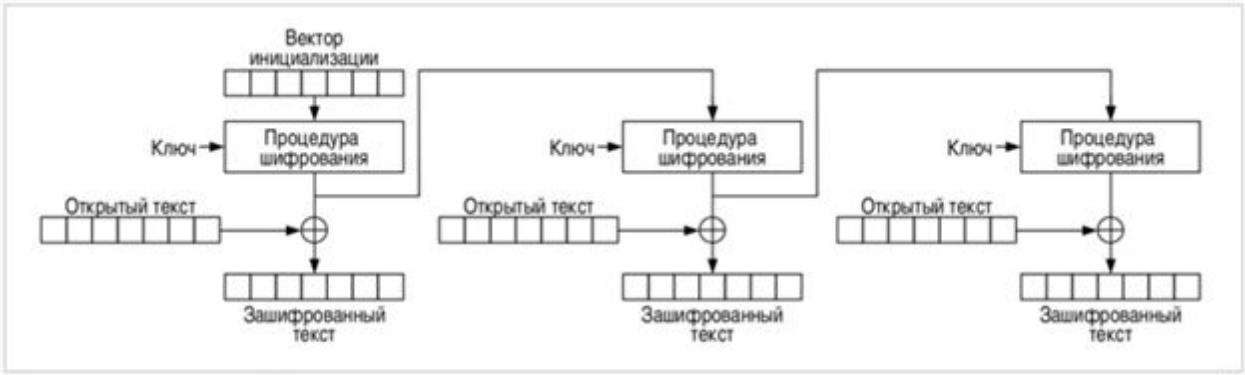
Расшифрование в режиме обратной связи по шифротексту

4. Описать и отобразить графически алгоритм OFB

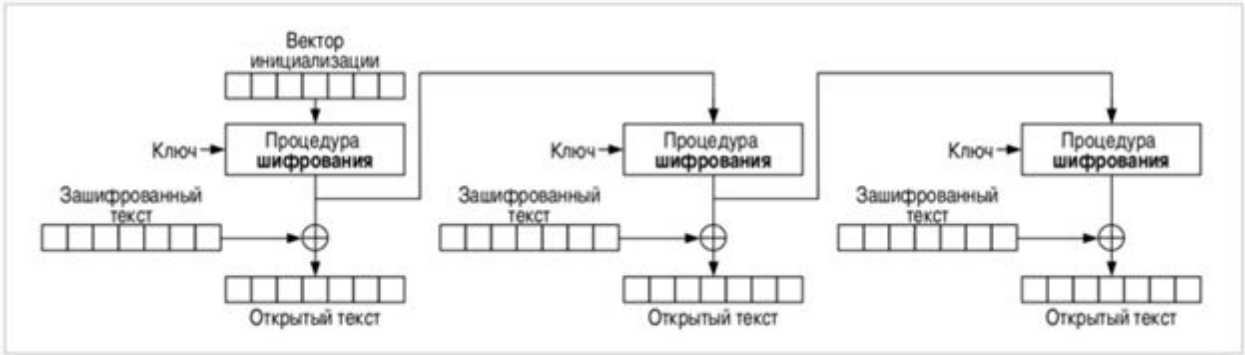
Режим (OFB) обратной связи вывода превращает блочный шифр в синхронный шифр потока: он генерирует ключевые блоки, которые являются результатом сложения с блоками открытого текста, чтобы получить зашифрованный текст. Так же, как с другими шифрами потока, зеркальное отражение в зашифрованном тексте производит зеркально отраженный бит в открытом тексте в том же самом местоположении. Это свойство позволяет многим кодам с исправлением ошибок функционировать как обычно, даже когда исправление ошибок применено перед кодированием.

Из-за симметрии операции сложения, шифрование и расшифрование похожи:

$$C_i = P_i \oplus O_i$$
$$P_i = C_i \oplus O_i$$
$$O_i = E_k(O_{i-1})$$
$$O_0 = IV$$



Шифрование в режиме OFB



Расшифрование в режиме OFB

Каждая операция блочного шифра обратной связи вывода зависит от всех предыдущих и поэтому не может быть выполнена параллельно. Однако, из-за того, что открытый текст или зашифрованный текст используются только для конечного сложения, операции блочного шифра могут быть выполнены заранее, позволяя выполнить заключительное шифрование параллельно с открытым текстом.

Данный метод называется также «режим обратной связи по выходу».

OFB также предполагает некое усовершенствование касающееся метода генерации независимой последовательности блоков: для получения очередного блока предлагается шифровать не с O_i, а с $O_i + IV (mod 2^{64})$, где некоторый вектор инициализации.

5. Описать и отобразить графически схему формирования ключей для алгоритма RSA

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
2. Вычисляется их произведение $n = p \cdot q$, которое называется *модулем*.
3. Вычисляется значение функции Эйлера от числа n : $\varphi(n) = (p-1) \cdot (q-1)$.
4. Выбирается целое число $\varphi(n)$ ($1 < e < \varphi(n)$), взаимно простое со значением функции . Обычно в качестве берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые числа Ферма 17, 257 или 65537.

· Число e называется *открытой экспонентой* (англ. *public exponent*)

· Время, необходимое для шифрования с использованием быстрого возведения в степень, пропорционально числу единичных бит в e .

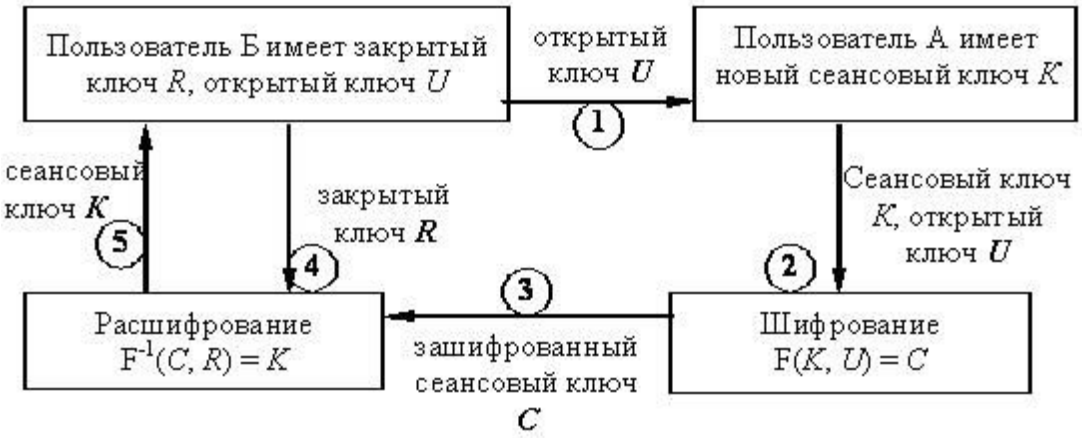
· Слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA

5. Вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению: $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

·Число d называется *секретной экспонентой*. Обычно, оно вычисляется при помощи расширенного алгоритма Евклида.

6.Пара $\{e, n\}$ публикуется в качестве *открытого ключа RSA* (англ. *RSA public key*).

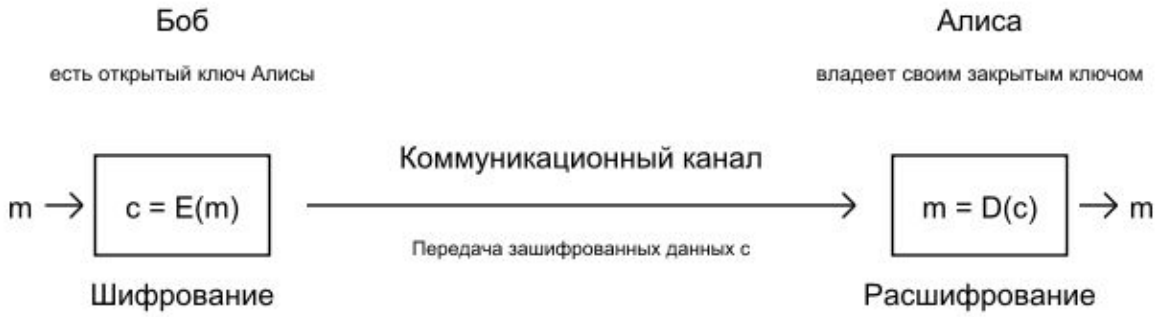
7. Пара $\{d, n\}$ играет роль *закрытого ключа RSA* (англ. *RSA private key*) и держится в секрете.



6. Описать и отобразить графически алгоритм шифрования и расшифровки сообщения по алгоритму RSA

Предположим, Боб хочет послать Алисе сообщение m.

Сообщениями являются целые числа в интервале от 0 до $n-1$, т.е $m \in \mathbb{Z}_n$.



Алгоритм:

Взять *открытый ключ* Алисы (e, n)

Взять *открытый текст* m

Зашифровать сообщение с использованием открытого ключа Алисы:

$$c = E(m) = m^e \mod n \quad (1)$$

Алгоритм:

· Принять зашифрованное сообщение c

· Взять свой *закрытый ключ* (d, n)

· Применить закрытый ключ для расшифрования сообщения:

$$m = D(c) = c^d \mod n \quad (2)$$

Данная схема на практике не используется по причине того, что она не является *практически надёжной*(semantically secured). Действительно, односторонняя функция E(m) является *детерминированной* — при одних и тех же значениях входных параметров

(ключа и сообщения) выдаёт одинаковый результат —, а это значит, что не выполняется необходимое условие практической (семантической) надёжности шифра.

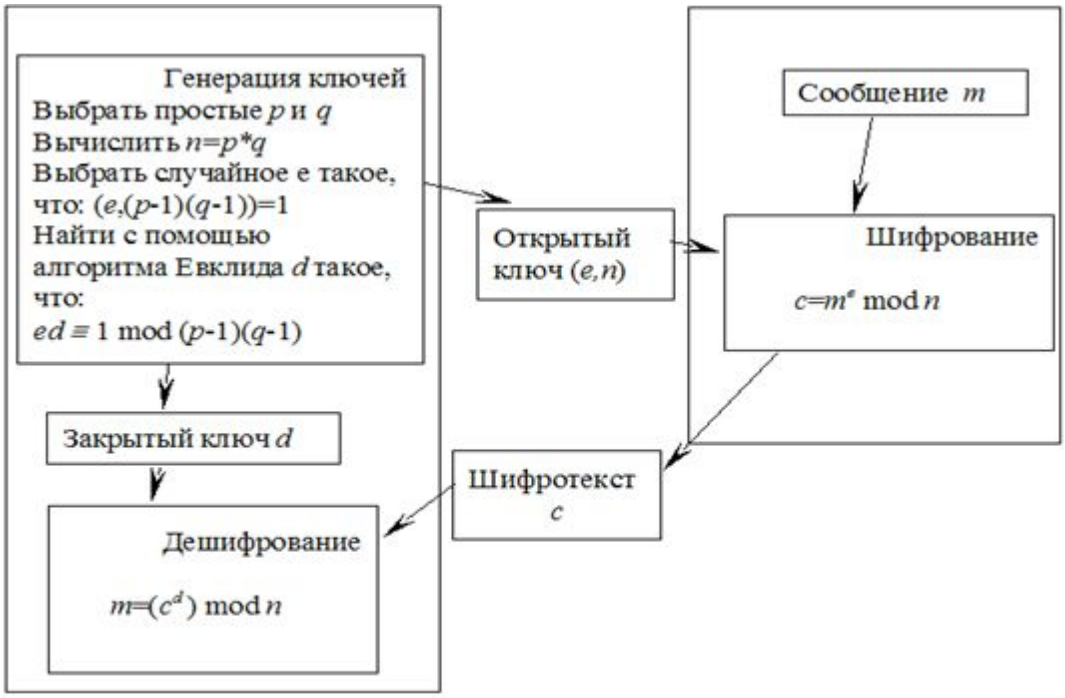


Рис.2.15. Схема шифрования алгоритма RSA

7. Описать и отобразить графически алгоритм работы функции MD2

Предполагается, что на вход подано сообщение, состоящее из b байт, хеш которого нам предстоит вычислить. Здесь b — произвольное неотрицательное целое число; оно может быть нулем или сколь угодно большим. Запишем сообщение побайтово, в виде: $m_0 m_1 \dots m_{b-1}$

Ниже приведены 5 шагов, используемые для вычисления хеша сообщения.

Шаг 1. Добавление недостающих бит.

Сообщение расширяется так, чтобы его длина в байтах по модулю 16 равнялась 0. Таким образом, в результате расширения длина сообщения становится кратной 16 байтам. Расширение производится всегда, даже если сообщение изначально имеет нужную длину.

Расширение производится следующим образом: i байт, равных i, добавляется к сообщению, так чтобы его длина в байтах стала равной 0 по модулю 16. В итоге, к сообщению добавляется как минимум 1 байт, и как максимум 15.

Шаг 2. Добавление контрольной суммы

16-байтная контрольная сумма сообщения добавляется к результату предыдущего шага.

Контрольная сумма вычисляется следующим образом: для каждого 16-байтного блока дополненного сообщения 16 раз выполняются следующие действия: $c = M[i * 16 + j]$

$$C[j] = S[c \oplus L] \oplus C[j],$$
$$L = C[j],$$

, где i — номер 16-байтного блока данных; j — номер текущего шага цикла;

$M[x]$ — x-й байт сообщения, то есть $M[i * 16 + j]$ — это j-й байт текущего блока данных;

c и L — временные переменные, L изначально содержит значение 0;

$C[j]$— j-й байт массива контрольной суммы; перед вычислением контрольной суммы массив содержит нулевые байты;

$S[i]$— i-й элемент 256-байтной матрицы из «случайно» переставленных цифр числа пи:

16-байтная контрольная сумма $C[0 \dots 15]$добавляется к сообщению. Теперь сообщение можно записать в виде $M[0 \dots N_1 - 1]$, где $M[0 \dots N_1 - 1]$.

Шаг 3. Инициализация MD-буфера

48-байтный буфер X используется для вычисления хеша. Он инициализируется нулями.

Шаг 4. Обработка сообщения блоками по 16 байт.

На этом шаге используется та же 256-байтная перестановочная матрица S, как и на шаге 2.

Каждый 16-байтный i-й блок дополненного сообщения, включая блок контрольной суммы, накладывается на буфер X следующим образом:

1. Блок данных копируется в буфер следующим образом:

$$X[16 + j] = M[i * 16 + j], \text{ j = 0…15,}$$

$$X[32 + j] = (X[16 + j] \oplus X[j]), \text{ j = 0…15.}$$

2. Выполняется обработка буфера, которая состоит из 18 раундов преобразований, в рамках каждого из которых выполняется обновление каждого байта буфера следующим образом:

$$X[k] = (X[k] \oplus S[t]), t = X[k] ,$$

где k = 0…47,

t — временная переменная, которая изначально имеет нулевое значение, а после выполнения каждого j-го раунда (j = 0…17) t модифицируется по правилу:

$$t = t + j \mod 256.$$

Шаг 5. Формирование хеша.

Хеш вычисляется как результат $X[0 \dots 15]$; в начале идет байт $X[0]$, а конце $X[15]$

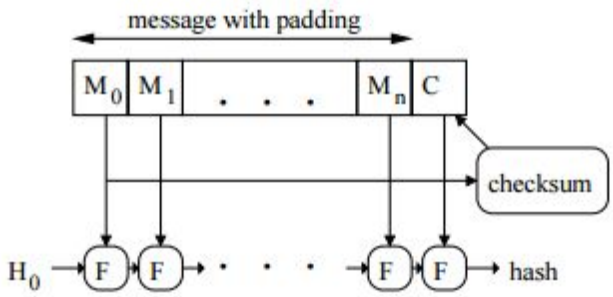


Fig. 1. The MD2 Hash Function

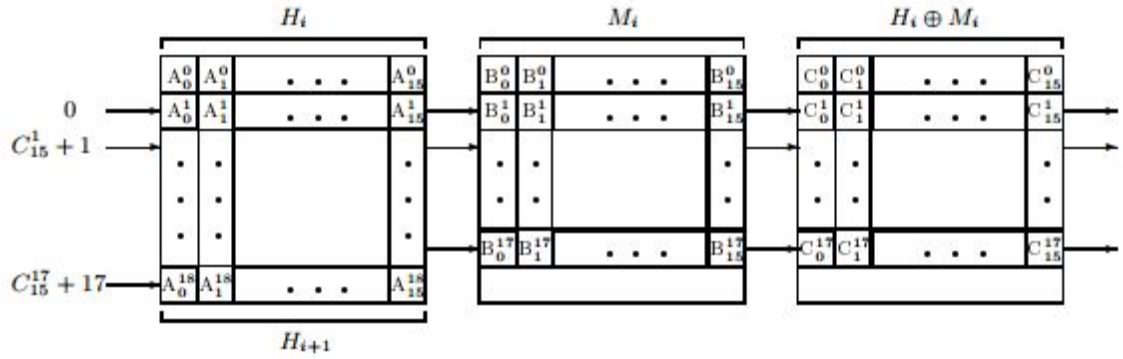


Fig. 2. The Compression Function of MD2

8. Описать и отобразить графически алгоритм работы функции MD4

Алгоритм MD4

Предполагается, что на вход подано сообщение, состоящее из b бит, хеш которого нам предстоит вычислить. Здесь b — произвольное неотрицательное целое число; оно может быть нулем, не обязано быть кратным восьми, и может быть сколь угодно большим. Запишем сообщение побитово, в виде: $m_0 m_1 \ldots m_{b-1}$

Ниже приведены 5 шагов, используемые для вычисления хеша сообщения.

Шаг 1. Добавление недостающих битов.

Сообщение расширяется так, чтобы его длина в битах по модулю 512 равнялась 448. Таким образом, в результате расширения, сообщению недостает 64 бита до длины, кратной 512 битам. Расширение производится всегда, даже если сообщение изначально имеет нужную длину.

Шаг 2. Добавление длины сообщения.

64-битное представление b (длины сообщения перед добавлением набивочных битов) добавляется к результату предыдущего шага. В маловероятном случае, когда b больше, чем 2^{64}, используются только 64 младших бита. Эти биты добавляются в виде двух 32-битных слов, и первым добавляется слово, содержащее младшие разряды.

На этом этапе (после добавления битов и длины сообщения) мы получаем сообщение длиной кратной 512 битам. Это эквивалентно тому, что это сообщение имеет длину, кратную 16-ти 32-битным словам. Каждое 32-битное слово содержит четыре 8-битных, но следуют они не подряд, а наоборот (например, из восьми 8-битных слов (a b c d e f g h) мы получаем два 32-битных слова (dcba hgfe)). Пусть $M[0 \ldots N-1]$ означает массив слов получившегося сообщения (здесь N кратно 16).

Шаг 3. Инициализация MD-буфера.

Для вычисления хеша сообщения используется буфер, состоящий из 4 слов (32-битных регистров): (A, B, C, D). Эти регистры инициализируются следующими шестнадцатеричными числами (младшие байты сначала):

word : 01 23 45 67
word : 89 ab cd ef
word : fe dc ba 98
word : 76 54 32 10

Шаг 4. Обработка сообщения блоками по 16 слов.

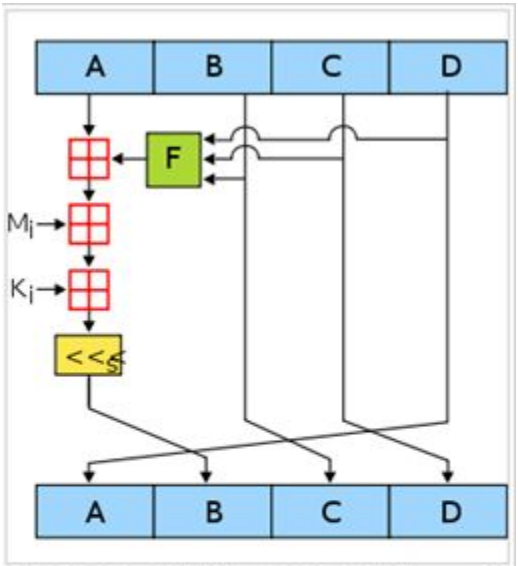
Для начала определим три вспомогательные функции, каждая из которых получает на вход три 32-битных слова, и по ним вычисляет одно 32-битное слово.

$$F(X, Y, Z) = XY \vee \neg X\, Z$$

$$G(X,Y,Z) = XY \vee XZ \vee YZ$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$

Шаг 5. Формирование хеша.

Результат (хеш-функция) получается как ABCD. То есть, мы выписываем 128 бит, начиная с младшего бита A, и заканчивая старшим битом D.



Одна операция MD4. Хеширование с MD4 состоит из 48 таких операций, сгруппированных в 3 раунда по 16 операций. *F* — нелинейная функция; в каждом раунде функция меняется. M_i означает 32-битный блок входного сообщения, а K_i — 32-битная константа, различная для каждой операции.

9. Описать и отобразить графически алгоритм работы функции MD5

На вход алгоритма поступает входной поток данных, хеш которого необходимо найти. Длина сообщения может быть любой (в том числе нулевой). Запишем длину сообщения в *L*. Это число целое и неотрицательное. Кратность каким-либо числам необязательна. После поступления данных идёт процесс подготовки потока к вычислениям.

Ниже приведены 5 шагов алгоритма:

Шаг 1. Выравнивание потока

Сначала дописывают единичный бит в конец потока (байт 80h), затем необходимое число нулевых бит. Входные данные выравниваются так, чтобы их новый размер был сравним с 448 по модулю 512 ($L' = 512 \times N + 448$). Выравнивание происходит, даже если длина уже сравнима с 448.

Шаг 2. Добавление длины сообщения

В конец сообщения дописывают 64-битное представление длины данных (количество бит в сообщении) до выравнивания. Сначала записывают младшие 4 байта, затем старшие. Если длина превосходит $2^{64} - 1$, то дописывают только младшие биты (эквивалентно взятию по модулю 2^{64}). После этого длина потока станет кратной 512. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит.

Шаг 3. Инициализация буфера

Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения шестнадцатеричными числами

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором.

Шаг 4. Вычисление в цикле

Заносим в блок данных элемент *n* из массива 512-битных блоков. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый).

После окончания цикла необходимо проверить, есть ли ещё блоки для вычислений. Если да, то переходим к следующему элементу массива (*n* + 1) и повторяем цикл.

Шаг 5. Результат вычислений

Результат вычислений находится в буфере ABCD, это и есть хеш. Если выводить побайтово, начиная с младшего байта A и закончив старшим байтом D, то мы получим MD5-хеш. 1, 0, 15, 34, 17, 18…

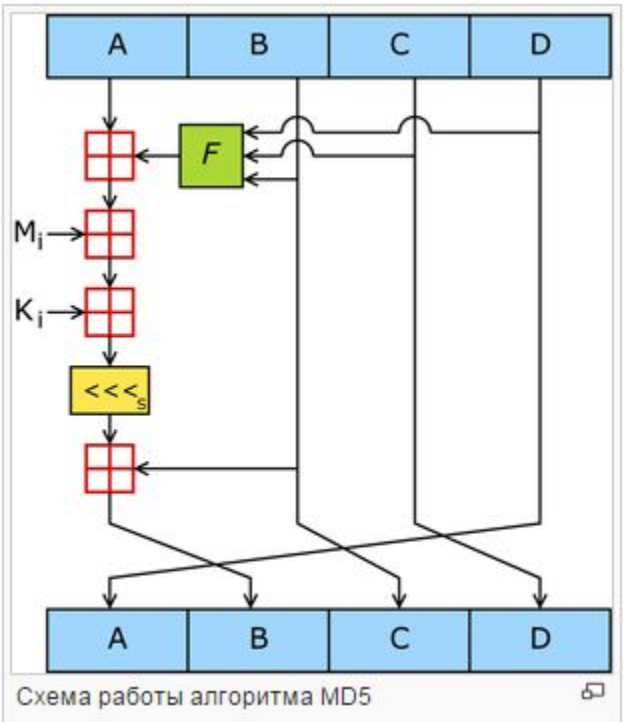


Схема работы алгоритма MD5

10. Описать и отобразить графически алгоритм работы функции SHA

Исходное сообщение разбивается на блоки по 512 бит в каждом. Последний блок дополняется до длины, кратной 512 бит. Сначала добавляется 1 (бит), а потом нули, чтобы длина блока стала равной (512 - 64 = 448) бит. В оставшиеся 64 бита записывается длина исходного сообщения в битах (в little-endian формате). Если последний блок имеет длину более 448, но менее 512 бит, то дополнение выполняется следующим образом: сначала добавляется 1 (бит), затем нули вплоть до конца 512-битного блока; после этого создается ещё один 512-битный блок, который заполняется вплоть до 448 бит нулями, после чего в оставшиеся 64 бита записывается длина исходного сообщения в битах (в little-endian формате). Дополнение последнего блока осуществляется всегда, даже если сообщение уже имеет нужную длину.

Инициализируются пять 32-битовых переменных.

A = a = 0x67452301

B = b = 0xEFCDAB89

C = c = 0x98BADCFE

D = d = 0x10325476
E = e = 0xC3D2E1F0
Определяются четыре нелинейные операции и четыре константы.

$F_t(m,l,k) = (m \wedge l) \vee (\neg m \wedge k)$	K_t = 0x5A827999	0≤t≤19
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0x6ED9EBA1	20≤t≤39
$F_t(m,l,k) = (m \wedge l) \vee (m \wedge k) \vee (l \wedge k)$	K_t= 0x8F1BBCDC	40≤t≤59
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0xCA62C1D6	60≤t≤79

Главный цикл

Главный цикл итеративно обрабатывает каждый 512-битный блок. Итерация состоит из четырех этапов по двадцать операций в каждом. Блок сообщения преобразуется из 16 32-битовых слов в 80 32-битовых слов по следующему правилу:

$W_t = M_t$ при 0≤t≤15

W_t = (W_t-3 $\oplus$ W_t-8 $\oplus$-14 $\oplus$ W_t-16) << 1 при 16≤t≤79

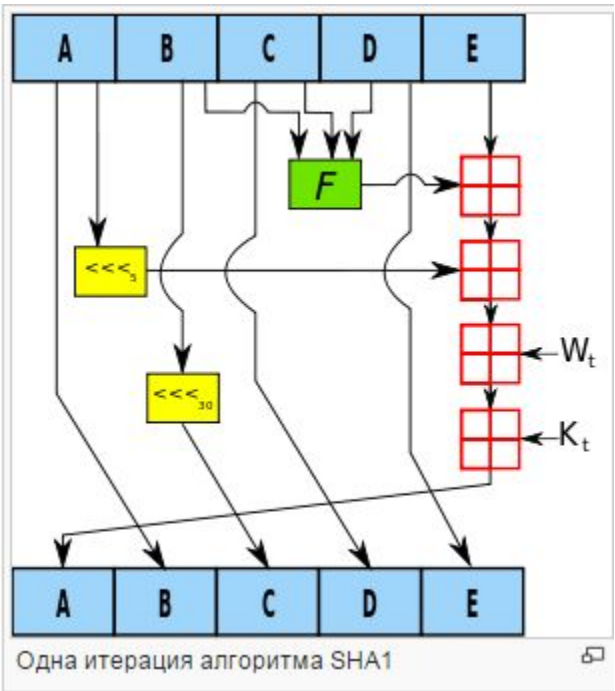
здесь << — это циклический сдвиг влево

для от 0 до 79

temp = (a<<5) + (b,c,d) + e +
e = d
d = c
c = b<<30
b = a
a = temp

После этого a, b, c, d, e прибавляются к A, B, C , D , E соответственно. Начинается следующая итерация.

Итоговым значением будет объединение пяти 32-битовых слов в одно 160-битное хеш-значение.



Одна итерация алгоритма SHA1