Белорусский национальный технический университет
Факультет информационных технологий и робототехники
Кафедра программного обеспечения информационных систем и технологий

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Основы информационной безопасности

для специальностей:
1 – 40 01 01 «Программное обеспечение информационных технологий»
1 – 40 05 01 «Информационные системы и технологии»
6-05-0612-01 «Программная инженерия»
6-05-0611-01 «Информационные системы и технологии»

Составитель: Белова С.В.

Минск ◊ БНТУ ◊ 2024

Перечень материалов

Электронный учебно-методический комплекс включает:

- перечень и краткое содержание основных разделов курса,
- теоретический раздел,
- лабораторный практикум,
- список литературы,
- вопросы к зачету,
- тесты к разделам курса.

Пояснительная записка

Электронный учебно-методический комплекс разработан для студентов специальностей 1 – 40 01 01 «Программное обеспечение информационных технологий», 1 – 40 05 01 «Информационные системы и технологии», 6-05-0612-01 «Программная инженерия», 6-05-0611-01 «Информационные системы и технологии».

Информационное наполнение ЭУМК соответствует программе дисциплины «Основы информационной безопасности».

ЭУМК может использоваться как при проведении занятий по дисциплине «Основы информационной безопасности», так и для организации самостоятельной работы студентов. Внедрение ЭУМК будет способствовать более эффективному овладению теоретическими и практическими основами информационной безопасности и криптографическими средствами защиты.

Информация в ЭУМК хорошо структурирована. Теоретический раздел включает основные темы курса. Лабораторный практикум содержит необходимый базовый методический материал (цель лабораторной работы, краткие теоретические сведения, задания на лабораторную работу, контрольные вопросы). В ЭУМК приводится также список литературы, соответствующий программе дисциплины. Модуль контроля знаний состоит из вопросов к зачету и тестов по основным разделам курса.

ЭУМК разработан в виде pdf-файла и не требует установки специального программного обеспечения.

СОДЕРЖАНИЕ

1 Программа дисциплины

1.1 Цели и задачи дисциплины

Целью изучения учебной дисциплины является получение студентами систематизированных знаний о средствах и методах защиты информационных ресурсов и основах управления интеллектуальной собственностью.

Основными задачами преподавания учебной дисциплины являются: изучение методологических основ информационной безопасности, моделей угроз, методов защиты, особенностей инженерно-технической защиты объектов от несанкционированного доступа, криптографических средств обеспечения безопасности, а также основ управления интеллектуальной собственностью. Теоретические знания должны быть закреплены на практике в ходе выполнения лабораторных работ.

Учебная дисциплина базируется на знаниях, полученных при изучении таких дисциплин как: «Основы алгоритмизации и программирования», «Объектно-ориентированное программирование», «Теория информации», «Компьютерные сети». Знания и умения, полученные студентами при изучении данной дисциплины, необходимы для освоения последующих специальных дисциплин и дисциплин специализаций, связанных с проектированием, разработкой и обеспечением безопасности современных программных продуктов, таких как: «Программирование сетевых приложений», «Разработка Internet и WEB-приложений» и др.

В результате изучения учебной дисциплины «Основы информационной безопасности студент должен:

знать:

- системную методологию и правовое обеспечение защиты информации;
- организационно-технические методы и технические средства защиты информации;
- основы криптографической защиты информации;
- особенности защиты информации в автоматизированных системах;
- основные положения международного и национального законодательства в области интеллектуальной собственности;
- порядок оформления и защиты прав на объекты интеллектуальной собственности;

уметь:

- определять возможные каналы утечки информации и обоснованно выбирать средства их блокирования;
- разрабатывать рекомендации по защите объектов различного типа от несанкционированного доступа;
- проводить патентные исследования;

- составлять заявки на выдачу охранных документов на объекты промышленной собственности;
- оформлять договора на передачу имущественных прав на объекты интеллектуальной собственности;

владеть:

- основными приемами анализа вероятных угроз информационной безопасности для заданных объектов;
- способами введения объектов интеллектуальной собственности в гражданский оборот;
- способами передачи прав на использование объектов интеллектуальной собственности.

1.2 Краткое содержание учебного материала

Раздел I. ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИОННЫХ РЕСУРСОВ

Тема 1.1. Введение в информационную безопасность

Предмет и содержание дисциплины, ее место, роль и значение для формирования специалиста. Понятие и модели информационной безопасности. Классификация угроз. Общая схема процесса обеспечения безопасности. Управление рисками.

Тема 1.2. Комплексный подход к обеспечению безопасности

Уровни информационной безопасности. Правовые и организационные методы защиты информации. Политика информационной безопасности. Стандарты и спецификации в области информационной безопасности. Основные программно-технические меры.

Тема 1.3. Технические каналы утечки информации

Классификация и характеристика технических каналов утечки информации. Пассивные и активные методы защиты информации от утечки по техническим каналам. Организация инженерно-технической защиты объектов от несанкционированного доступа.

Раздел II.КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Тема 2.1. Основы криптографической защиты

Криптография как наука. Основные понятия и термины. Стеганография. Типы криптографических систем. Типы криптографических атак и стойкость алгоритмов.

Тема 2.2. Криптография с секретным ключом

Модель традиционного шифрования. Сети Файстеля. Стандарт шифрования данных DES. Режимы работы блочных шифров. Объединение блочных шифров. Характеристика современных симметричных блочных шифров.

Тема 2.3. Криптография с открытым ключом

Принципы построения криптосистем с открытым ключом. Математические основы шифрования с открытым ключом. Алгоритм RSA.

Тема 2.4. Управление криптографическими ключами

Криптографические протоколы. Распределение секретных ключей. Распределение открытых ключей. Обмен ключами по схеме Диффи-Хеллмана

Тема 2.5. Аутентификация сообщений и функции хеширования

Понятие аутентификации. Методы аутентификации сообщений. Функции хеширования. Обобщенная структура функции хеширования. Алгоритмы хеширования.

Тема 2.6. Электронная цифровая подпись (ЭЦП)

Требования к ЭЦП. Непосредственная и арбитражная ЭЦП. Протоколы ЭЦП. Стандарт цифровой подписи DSS. Алгоритм ЭЦП RSA.

Тема 2.7. Идентификация и аутентификация пользователей

Основные понятия. Пароли. Приложения аутентификации. Протокол аутентификации Kerberos. Служба аутентификации X.509.

Тема 2.8. Защита в операционных системах

Управление доступом. Защита объектов в операционных системах. Политики безопасности, регистрация событий, журналы аудита. Безопасность баз данных. Защита информации в технологии .NET.

Тема 2.9. Защита на сетевом уровне

Иерархия сервисов защищенного канала. Преимущества IPSec. Архитектура IPSec. Транспортный и туннельный режимы. Протоколы AH и ESP. Межсетевые экраны.

Тема 2.10. Защита прикладных протоколов

Характеристика и функции PGP. Многоцелевые расширения электронной почты. Система S/MIME. Проблемы защиты Web.Протоколы SSL и TLS. Защита информации в автоматизированных системах.

Раздел III. ОСНОВЫ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТЬЮ

Тема 3.1. Интеллектуальная собственность

Понятие, функции и значение интеллектуальной собственности в развитии общества. Становление и развитие законодательства в области интеллектуальной собственности. Цели и задачи государственного управления интеллектуальной собственностью. Правовая охрана объектов интеллектуальной собственности. Авторское право. Патентная информация. Правовая охрана программ и баз данных.

2 Теоретический раздел

2.1 Введение в информационную безопасность

Понятие информационной безопасности

В наше время движущей силой и главным объектом всех отраслей человеческой деятельности становится информация. Огромные информационные потоки буквально захлестывают людей.

Информация – сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления.

Справедлив афоризм: «Кто владеет информацией, тот владеет миром».

Защищаемая информация – информация, являющаяся предметом собственности и подлежащая защите в соответствии с требованиями правовых документов или требованиями, устанавливаемыми собственниками информации. Собственниками информации могут быть: государство, юридические и физические лица.

Огромные объемы информации, в том числе и критически важной для отдельных людей, организаций или государств, хранятся, обрабатываются и передаются с использованием автоматизированных систем (АС) обработки информации.

Постоянно повышается зависимость общества от степени безопасности используемых им информационных технологий. От них порой зависит благополучие и даже жизнь многих людей.

Некоторые современные формы бизнеса полностью базируются на сетевых информационных технологиях (электронная торговля, сетевое провайдерство и т.д.) и по этой причине особенно уязвимы.

Как соотносятся развитие ИТ и обеспечение безопасности?

Анализ состояния дел в сфере защиты информации показывает, что злоумышленные действия над информацией не только не уменьшаются, но имеют устойчивую тенденцию к росту.

Технологии развиваются. Центральные процессоры стали намного быстрее работать, что дает возможность применять шифрование почти повсеместно. Например, можно полностью зашифровать цифровую сотовую связь с помощью сильных алгоритмов без видимого замедления работы.

Технологии сетевой безопасности тоже совершенствуются. Сегодняшние брандмауэры намного эффективнее разработанных 10 лет назад и т.д..

Но кое-что остается неизменным — основы технологий и люди, использующие их. Криптография всегда будет не больше, чем математика. Недостатки безопасности всегда будут присутствовать в программном

обеспечении. Пользователи никогда не захотят запоминать длинные пароли. Люди будут всегда уязвимы для манипуляций.

Ситуация ухудшается. Будущее цифровых систем — сложность, а сложность — главный враг безопасности. И поскольку киберпространство продолжает усложняться, безопасность будет становиться все более хрупкой.

Повышение быстродействия микросхем, развитие архитектур с высокой степенью параллелизма позволяет методом грубой силы преодолевать барьеры (прежде всего криптографические), ранее казавшиеся неприступными.

Быстрое развитие информационных технологий, одной стороны, предоставляет новые возможности по защите информации, с другой стороны, объективно затрудняет обеспечение надежной защиты.

Безопасность является таким же свойством системы, как надежность или производительность.

Словосочетание информационная безопасность в разных контекстах может иметь различный смысл.

Например, в Доктрине информационной безопасности страны или других законодательных актах термин «информационная безопасность» обычно используется в широком смысле.

Информационная безопасность – это состояние защищенности информационной среды общества, обеспечивающее ее формирование, использование и развитие в интересах личности, общества и государства.

В нашем курсе термин **информационная безопасность** будет использоваться в узком смысле.

Информационная безопасность – защищенность информации и поддерживающей инфраструктуры от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести неприемлемый ущерб субъектам информационных отношений (владельцам и пользователям информации и поддерживающей инфраструктуры).

Трактовка проблем, связанных с информационной безопасностью, для разных категорий субъектов может существенно различаться. Достаточно сопоставить режимные государственные организации и учебные заведения. В первом случае «пусть лучше все сломается, чем враг узнает хоть один секретный бит», во втором – «нет никаких секретов, лишь бы все работало».

На защиту сетевой игры требуется меньше времени и усилий, чем на обеспечение безопасности приложения для военной разведки или медицины. Необходимость и уровень защиты определяются конкретной ситуацией.

Согласно определению, информационная безопасность зависит не только от компьютеров, но и от поддерживающей инфраструктуры (системы электро-, водо-, теплоснабжения, средства коммуникаций, обслуживающий персонал).

Обратим внимание, что в определении указывается на неприемлемый ущерб. Застраховаться от всех видов ущерба невозможно. Чаще всего порог

неприемлемости имеет денежное выражение. Стоимость средств защиты не должна превышать размер ожидаемого ущерба.

Согласно определению, приведенному в Законе Республики Беларусь «Об информации, информатизации и защите информации»:

Защита информации – комплекс правовых, организационных и технических мер, направленных на обеспечение конфиденциальности, целостности, подлинности, доступности и сохранности информации. ***Источник: https://pravo.by/document/?guid=3871&p0=h10800455 – Национальный правовой Интернет-портал Республики Беларусь.***

Целями защиты информации являются:

обеспечение национальной безопасности, суверенитета Республики Беларусь;

сохранение и неразглашение информации о частной жизни физических лиц и персональных данных, содержащихся в информационных системах;

обеспечение прав субъектов информационных отношений при создании, использовании и эксплуатации информационных систем и информационных сетей, использовании информационных технологий, а также формировании и использовании информационных ресурсов;

недопущение неправомерного доступа, уничтожения, модификации (изменения), копирования, распространения и (или) предоставления информации, блокирования правомерного доступа к информации, а также иных неправомерных действий.

Источник: https://pravo.by/document/?guid=3871&p0=h10800455 – Национальный правовой Интернет-портал Республики Беларусь

Модель КЦД

Понятие информационной безопасности может быть пояснено с помощью **моделей безопасности**.

Система считается безопасной, если она может противостоять нарушениям. Но нарушений безопасности может быть очень много. Суть моделей безопасности в том, что множество всех видов нарушений безопасности делится на несколько базовых групп. Затем любое из нарушений относится к одной из групп. Если система может противостоять любой из групп нарушений, то она считается безопасной.

Одной из первых и наиболее популярных моделей безопасности является **модель «Триада КЦД» (Конфиденциальность, Целостность, Доступность)** или в англоязычной форме – **CIA** (Confidentiality, Integrity, Availability).

Модель КЦД предложена Зальтцером и Шредером в 1975г.

Согласно этой модели, все возможные нарушения информационной безопасности могут быть отнесены к одной из 3 групп: нарушения конфиденциальности, нарушения целостности, нарушения доступности.

Конфиденциальность – защита от несанкционированного доступа к информации или гарантия того, что информация будет доступна только тем субъектам, которым разрешен доступ.

Целостность – защищенность информации от разрушения и несанкционированного изменения или гарантия сохранения данными правильных значений.

Доступность – возможность за приемлемое время получить требуемую информационную услугу или гарантия того, что авторизованные пользователи всегда получат доступ к хранящейся в компьютерной системе информации (в любое время, по первому требованию).

Доступность – наиболее важный элемент ИБ, т.к. информационные системы создаются для получения информационных услуг. Почти для всех субъектов, кто реально использует информационные системы, на первом месте стоит доступность.

Целостность оказывается важнейшим аспектом ИБ в тех случаях, когда информация служит «руководством к действию». Рецептура лекарств, набор и характеристики комплектующих изделий, ход технологического процесса – это примеры информации, нарушение целостности которой может оказаться в буквальном смысле смертельным.

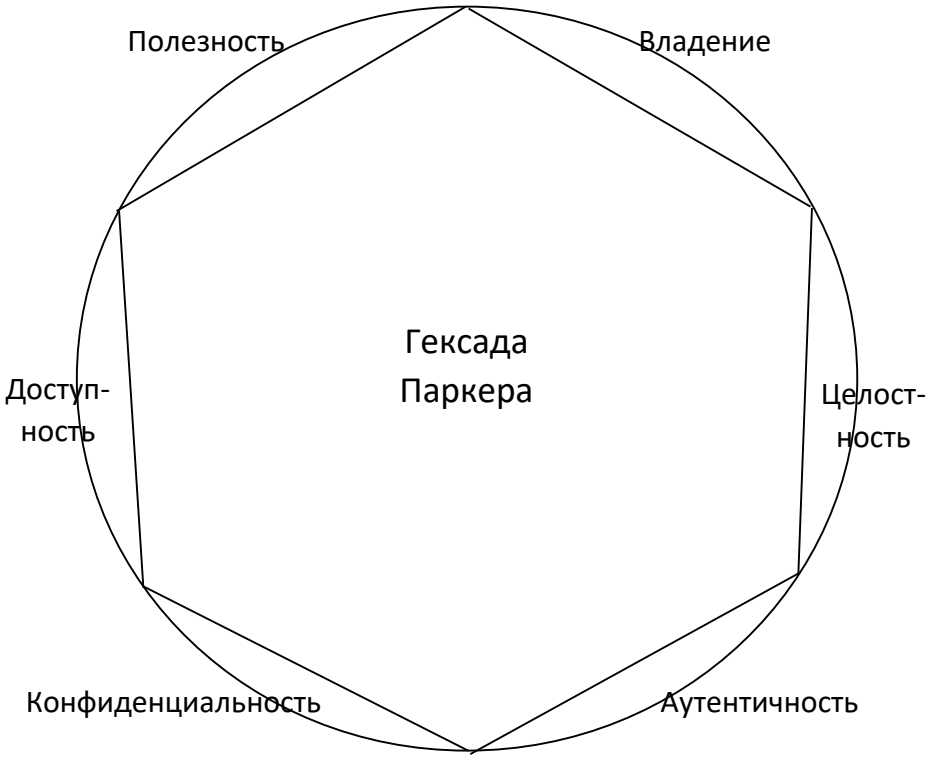
Конфиденциальность – самый проработанный аспект ИБ. Однако, практическая реализация его наталкивается на многочисленные законодательные препоны и технические проблемы.

Безопасная информационная система по определению обладает свойствами конфиденциальности, доступности и целостности. Это основные составляющие информационной безопасности.

Гексада Паркера и модель STRIDE

Одной из популярных моделей безопасности является гексада Паркера.

«Гексада Паркера» определяет **6 базовых видов нарушений**: конфиденциальность, целостность, доступность, аутентичность, владение, полезность.



Аутентичность – гарантия того, что субъект не может выдать себя за другого, а документ всегда имеет достоверную информацию об авторе (источнике). Аналог неотказуемости.

Владение – гарантия того, что физический контроль над устройством или другой средой хранения информации предоставляется только тем, кто имеет на это право.

Полезность – такое состояние ИС, при котором обеспечивается удобство практического использования как собственно информации, так и связанных с ее обработкой и поддержкой процедур.

Например, предпринимаемые для защиты системы меры не должны значительно затруднять работу сотрудников, иначе они будут восприниматься как помехи, и сотрудники попытаются их обойти.

Существует еще одна модель безопасности - **модель STRIDE.** Эта модель используется компанией Microsoft для разработки безопасного ПО.

Согласно модели STRIDE ИС находится в безопасности, если она защищена от следующих типов нарушений: подмена данных, изменение данных, отказ от ответственности, разглашение сведений, отказ в обслуживании, захват привилегий.

Spoofing	Подмена данных
Tampering	Изменение данных
Repudiation	Отрицание авторства
Information disclosure	Разглашение сведений
Denial of Service	Отказ в обслуживании
Elevation of Privilege	Захват привилегий

Подмена данных – такое нарушение, при котором субъект путем подмены данных выдает себя за другого, получая возможность нанесения вреда системе. Например, подмена IP-адреса отправителя в IP-пакете.

Изменение данных – нарушение целостности.

Отрицание авторства – негативная форма свойства неотказуемости.

Разглашение сведений – нарушение конфиденциальности.

Отказ в обслуживании – нарушение доступности.

Захват привилегий - такое нарушение, при котором субъект несанкционированным образом повышает свои полномочия в системе. Например, незаконное присвоение злоумышленником прав сетевого администратора.

Российский государственный стандарт ГОСТ 13335-1:2006 «Информационная технология. Методы и средства обеспечения безопасности» дает определение информационной безопасности на основе гексады Паркера:

Информационная безопасность – все аспекты, связанные с определением, достижением, и поддержанием конфиденциальности, целостности, доступности, неотказуемости, подотчетности, аутентичности и достоверности информации и средств ее обработки.

Уязвимости и угрозы. Виды угроз

Уязвимость – это слабое звено информационной системы.

Например:

- сбои и отказы оборудования ИС;
- последствия ошибок проектирования и разработки компонентов ИС: аппаратных средств, технологий обработки информации, алгоритмов, программ, структур данных и т.п. (ошибка в программе, использование слабого алгоритма шифрования)

- ошибки настройки и эксплуатации (примитивный пароль, неправильное назначение прав доступа к файлу с важными данными)

Уязвимости могут быть: скрытыми, известными теоретически, общеизвестными и активно используемыми злоумышленниками.

Промежуток времени от момента, когда об уязвимости становится известно, и до момента, когда уязвимость ликвидируется, называется **окном опасности**.

Пока существует окно опасности, возможны успешные атаки на ИС. Для большинства уязвимых мест окно опасности существует сравнительно долго (дни, недели). За это время должны быть выпущены и затем установлены соответствующие **заплаты в ПО**, называемые **патчами** (patch – заплатка).

Общеизвестные, но не исправленные ошибки в ПО – один из самых распространенных типов уязвимостей.

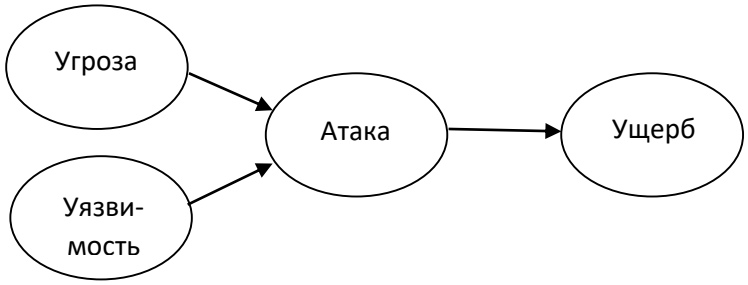
Другой тип уязвимостей – ошибки в конфигурировании программных и аппаратных средств. (В окне входа не должно показываться последнее имя пользователя, учетную запись Гость надо отключить, Администратор – переименовать).

Поиск уязвимостей – важная часть задачи обеспечения безопасности. Это трудоемкая задача. Поэтому для автоматизации поиска уязвимостей используют различные программные инструменты – **средства сканирования уязвимостей**. Например, McAfee, Nessus и др.

Угроза – потенциальная возможность нарушить информационную безопасность (конфиденциальность, целостность, доступность информации).

Атака – реализованная угроза.

Атака может произойти только тогда, когда одновременно существуют уязвимость и направленная на использование этой уязвимости угроза.



Любая угроза направлена на поиск и/или использование уязвимостей системы.

Злоумышленник может действовать «на ощупь», пытаясь обнаружить уязвимость системы. А может использовать специальные инструментальные средства - эксплоиты.

Эксплоит (exploit) – программа, которая автоматизируют процесс вторжения в систему.

В Интернете можно даже найти предложения о сдаче в аренду целых бот-сетей, предназначенных для реализации мощных кибератак.

Бот-сеть или **ботнет (botnet)** – организованная совокупность компьютеров, связанных через Интернет и способных согласованно решать задачи, поставленные злоумышленником.

Все виды угроз могут быть классифицированы по разным признакам.

Универсальной классификации угроз не существует. Каждый день применяются новые способы незаконного проникновения в сеть, разрабатываются новые средства мониторинга сетевого трафика, появляются новые вирусы и т.д. В ответ на это разрабатываются все более изощренные средства защиты, которые ставят преграду на пути многих типов угроз, но затем сами становятся новыми объектами атак.

По источнику различают:

1) природные (стихийные бедствия - наводнение, ураган, землетрясение, пожар и т.п.);
2) техногенные угрозы (аварии).
3)

По расположению источника угроз различают:

1) внутренние (внутри ИС);
2) внешние.

Практика показывает, что 80% усилий тратится на противодействие внешним угрозам, а 70% атак, наносящих ущерб, производится из локальной сети.

По способу осуществления различают:

1) Неумышленные (случайные);
2) Умышленные (преднамеренные).

Неумышленные угрозы вызываются ошибочными действиями лояльных сотрудников, становятся следствием их низкой квалификации или безответственности.

В реальном мире у нас всегда имеется потенциально слабое звено, которое мы любовно называем «пользователем».

Люди – наиболее уязвимое звено любой системы безопасности. Чем больше людей пользуются системой, тем меньше ее надежность.

Никакие меры безопасности не в состоянии защитить систему от недобросовестного или ленивого пользователя. Причиной нарушения безопасности сплошь и рядом являются не плохие алгоритмы и протоколы, а обычные человеческие слабости (небрежность, недостаточная подготовка, низкая дисциплина, излишняя доверчивость, неопытность и наивность).

Самыми частыми и самыми опасными (с точки зрения размера ущерба) являются непреднамеренные ошибки штатных пользователей, операторов, системных администраторов, программистов и т.д..

Иногда такие ошибки являются собственно угрозами (неправильно введенные данные или ошибка в программе, вызвавшая крах системы), иногда они создают уязвимости (окна опасности), которыми могут воспользоваться

злоумышленники (таковы обычно ошибки администрирования). По некоторым данным, до 65% потерь – следствие непреднамеренных ошибок.

Пожары и наводнения не приносят столько бед, сколько безграмотность и небрежность в работе.

Кроме того, к такому роду угроз относятся последствия ненадежной работы программных (ошибки в ПО) и аппаратных средств системы (отказы и сбои аппаратуры). Есть потенциально опасные объекты, не создаваемые специально для нанесения вреда. Это программы, разработанные непрофессионалами и содержащие ошибки. Любая ошибка в ПО может стать объектом атаки хакера. Использование несертифицированных программ несет достаточно высокий уровень риска.

Поэтому вопросы безопасности тесно переплетаются с вопросами надежности и отказоустойчивости технических и программных средств, вопросами обучения сотрудников.

К неумышленным угрозам также относятся аварийные ситуации из-за стихийных бедствий и отключений электропитания.

Подведем итог.

Причинами случайных угроз могут быть:

- ошибки в работе обслуживающего персонала и пользователей;
- ошибки в программном обеспечении;
- аварийные ситуации из-за стихийных бедствий и отключения электропитания;
- отказы и сбои аппаратуры;
- помехи в линиях связи из-за воздействий внешней среды.

Умышленные угрозы связаны с целенаправленными действиями нарушителя.

Преднамеренные угрозы могут реализовать как внутренние для системы участники процесса обработки данных (персонал, сервисное звено и т. д.), так и люди, внешние по отношению к системе, так называемые «хакеры».

Действия нарушителя могут быть вызваны разными мотивами: недовольством служащего своей карьерой, материальным интересом, конкурентной борьбой и т.д..

Весьма опасны так называемые «обиженные» сотрудники – нынешние и бывшие. Как правило они стремятся нанести вред организации-«обидчику» (испортить оборудование, удалить данные и т.д.). Знакомство с порядками в организации позволяет им нанести немалый ущерб. Необходимо следить за тем, чтобы при увольнении сотрудника его права доступа к информационным ресурсам аннулировались.

Противники

Кто же все-таки угрожает цифровому миру? Противники — те же самые, что и в обычном мире: уголовные преступники, жаждущие обогащения; промышленные шпионы, охотящиеся за секретами, хакеры, разведка,

добывающая военные сведения. Они не изменились, просто киберпространство стало новым полем их деятельности.

Цели противников могут быть различны: причинение ущерба, финансовая выгода, информация и т. д.

Хакеры

Термин «хакер» имеет широкий спектр значений.

Хакер в современном значении - человек, взламывающий компьютеры. Надо заметить, что раньше быть хакером не считалось чем-то противозаконным, скорее, это была характеристика человека, умеющего профессионально обращаться с компьютерами. В наши дни хакерами мы называем тех, кто ищет пути вторжения в компьютерную систему или выводит ее из строя. И все же слово скорей используется как комплимент, нежели как оскорбление.

В последнее время используется слово «крекер» (взломщик программной защиты) для плохих парней и «хакер» — для хороших.

White-hats – «белые шляпы», «хорошие» хакеры, исследующие защиту систем в «мирных» целях.

Шнайер: «Я определяю хакера как индивидуума, который экспериментирует с недостатками системы ради интеллектуального любопытства или собственного удовольствия; это слово описывает человека со специфическим набором навыков и неспецифической моралью. Есть хорошие хакеры и плохие хакеры, аналогично хорошим водопроводчикам и плохим водопроводчикам. (Есть также «хорошие плохие» хакеры и «плохие хорошие» хакеры... но не берите это в голову.)»

Хакеры стары как любопытство, хотя сам по себе этот термин современен. Галилео Галилей был хакером. Мария Кюри тоже. Аристотель не был. (Аристотель приводил некие теоретические доказательства, что у женщины меньшее количество зубов, чем у мужчины. Хакер просто посчитал бы зубы своей жены. *Хороший* хакер посчитал бы зубы своей жены без ее ведома, в то время когда она спала бы. Хороший *плохой* хакер мог бы удалить некоторые из них, только бы доказать свое теоретическое предположение.)

Исследования показали, что хакерами чаще всего становятся:

- мужчины;
- в возрасте от 16 до 35 лет;
- одинокие;
- образованные;
- технически грамотные.

Они имеют свою собственную культуру: хакерские имена-прозвища, язык, правила. И, что характерно для любой субкультуры, только маленький процент ее представителей действительно что-то собой представляет. К сожалению, большинство хакеров совершают незаконные действия.

Атаки разделяют на активные и пассивные.

Активные атаки – явные воздействия на систему, изменяющие ее состояние.

На пример:

- изменение, добавление, переупорядочение, удаление данных;
- изменение содержимого передаваемого по сети сообщения или внесение дополнительных сообщений (сетевых пакетов и т.п.). Такие действия называются активным прослушиванием;
- модификация программы с целью изменения ее функций и характеристик;
- внедрение вредоносного ПО.

Пассивные атаки – атаки, не нарушающие нормальную работу системы, связанные со сбором информации о системе или перехватом трафика.

При пассивном вторжении (перехвате информации) нарушитель только наблюдает за прохождением информации в ВС, не вторгаясь ни в информационный поток, ни в содержание передаваемой информации.

Например, подсматривание, подслушивание, подключение к сетевому кабелю с целью перехвата данных и анализа трафика (sniffing), незаконное копирование файлов и программ, кражи оборудования.

Обычно такие атаки не оставляют следов, их сложно выявить.

На практике редко «в чистом виде» используются пассивные и активные атаки. Сначала идет сбор информации о системе, а затем активное внедрение.

Для сбора информации может использоваться социальный инжиниринг.

Социальный инжиниринг – получение несанкционированного доступа к информации или системе без применения технических средств. Хакер играет на человеческих слабостях. Оружие – приятный голос и актерские способности. Хакер может позвонить по телефону сотруднику компании под видом службы технической поддержки и узнать его пароль "для решения небольшой проблемы в компьютерной системе сотрудника". В большинстве случаев этот номер проходит.

Иногда хакер под видом служащего компании звонит в службу технической поддержки. Если ему известно имя служащего, то он говорит, что забыл свой пароль, и в результате либо узнает пароль, либо меняет его на нужный. Учитывая, что служба технической поддержки ориентирована на безотлагательное оказание помощи, вероятность получения хакером хотя бы одной учетной записи весьма велика.

Общая схема процесса обеспечения безопасности. Управление рисками

Абсолютная безопасность информационной системы не может быть достигнута никакими средствами. Всегда остается вероятность появления уязвимостей и проведения атак.

Цель обеспечения информационной безопасности – минимизация возможного негативного влияния угроз.

Для этого надо знать уязвимости и угрозы, какими из них можно пренебречь, а какие очень опасны. Мерой опасности угроз и атак является возможный ущерб.

Ущерб (loss, impact) – это негативное влияние на систему проведенной атакой.

В качестве ущерба рассматриваются не столько потери на восстановление системы, сколько бизнес-потери, которые в результате нарушений понесло предприятие

Риск – вероятностная оценка величины возможного ущерба, который может понести предприятие в результате успешно проведенной атаки.

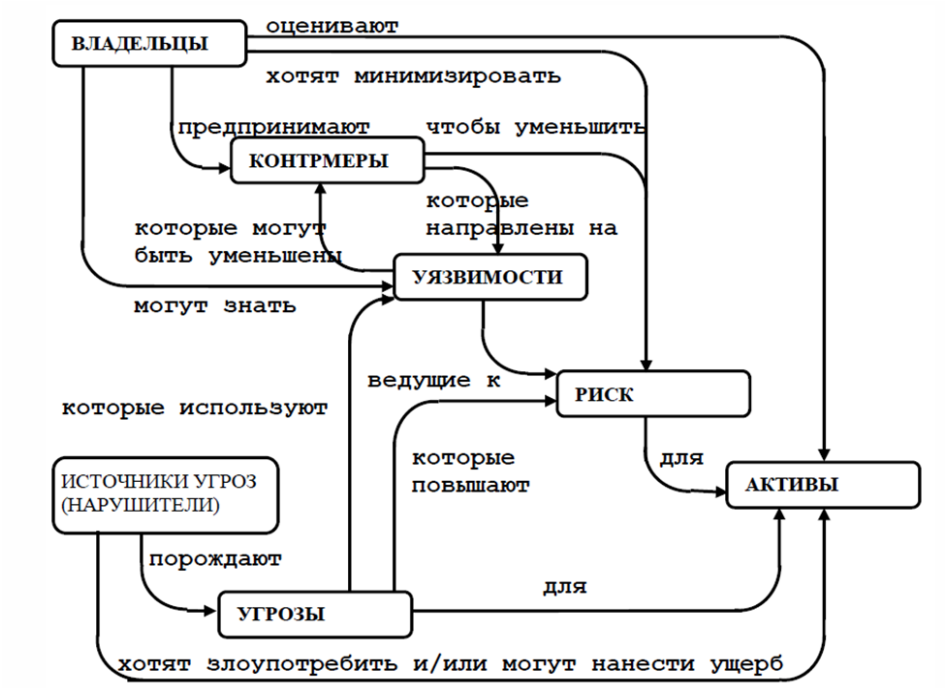
Чем более уязвимой является существующая система безопасности, тем выше вероятность реализации атаки и, следовательно, тем выше значение риска.

Риск характеризуется парой:

(Ущерб от атаки, Вероятность атаки)

Рассмотрим взаимосвязь основных субъектов и объектов обеспечения безопасности, как это предлагается в международном стандарте ISO/IEC-15408 (в России он принят как ГОСТ Р ИСО/МЭК 15408-2002 [4]).

Представим общую схему процесса обеспечения безопасности.



Безопасность связана с защитой активов от угроз. Разработчики стандарта отмечают, что следует рассматривать все разновидности угроз, но в сфере безопасности наибольшее внимание уделяется тем из них, которые связаны с действиями человека.

За сохранность активов отвечают их владельцы, для которых они имеют ценность. Существующие или предполагаемые нарушители также могут придавать значение этим активам и стремиться использовать их вопреки интересам их владельца. Действия нарушителей приводят к появлению угроз. Как уже отмечалось выше, угрозы реализуются через имеющиеся в системе уязвимости.

Владельцы активов анализируют возможные угрозы, чтобы определить, какие из них могут быть реализованы в отношении рассматриваемой системы.

В результате анализа определяются риски (т. е. события или ситуации, которые предполагают возможность ущерба) и проводится их анализ.

Владельцы актива предпринимают контрмеры для уменьшения уязвимостей и выполнения политики безопасности. Но и после введения этих контрмер могут сохраняться остаточные уязвимости и соответственно – остаточный риск.

Управление рисками – это системный анализ угроз, прогнозирование и оценка их последствий для предприятия и выбор контрмер, направленных на уменьшение возможного негативного воздействия нарушений на деятельность предприятия.

Управление рисками включает 3 этапа:

1) Анализ уязвимостей;
2) Оценка рисков;
3) Риск-менеджмент.

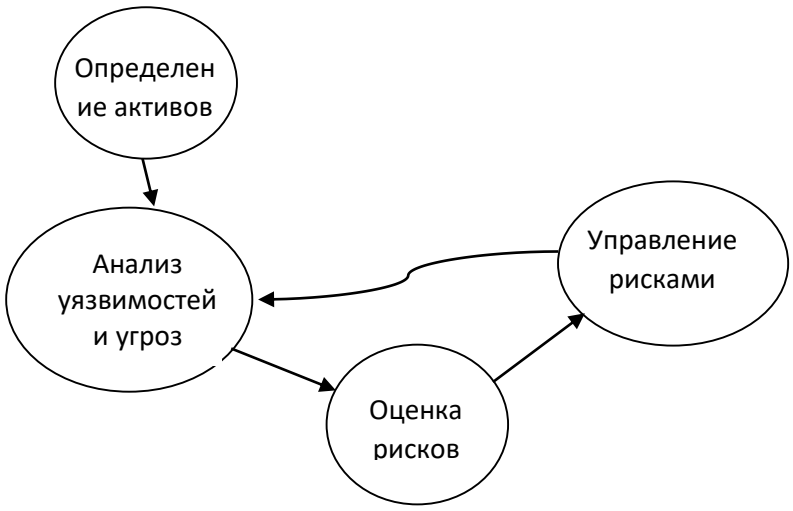
Анализ уязвимостей – объективное исследование реально существующих активов предприятия, являющихся объектом защиты: оборудования, сети, ПО, документации, баз данных. Выявление уязвимостей и возможных угроз.

Оценка рисков – ранжирование возможных угроз по степени опасности. Вычисление рисков (чем выше ущерб и вероятность, тем больше риск).

Риск-менеджмент – принятие конкретных мер (разработка и внедрение политики безопасности предприятия)

По каждому риску предпринимаются меры из списка:

- Принятие риска. Касается неизбежных атак, наносящих приемлемый ущерб.
- Устранение риска. Существующий риск сводится на нет либо устранением уязвимости (исправить ошибку), либо угрозы (установить антивирус).
- Снижение риска. Если риск невозможно ни принять, ни устранить, то предпринимаются действия по его снижению (более строгие требования к паролям).
- Перенаправление риска. Если невозможно вышеизложенное, то риск может быть перенаправлен страховой компании.



2.2 Комплексный подход к обеспечению безопасности

Уровни информационной безопасности

Для обеспечения информационной безопасности необходим комплексный системный подход.

Все методы защиты информации по характеру проводимых действий можно разделить на уровни:

1) **Законодательные меры обеспечения информационной безопасности** – это законы, постановления, указы, нормативные акты и стандарты.

2) **Административные меры** – это действия, предпринимаемые руководством предприятия или организации для обеспечения информационной безопасности.

3) **Процедурные меры** – это меры безопасности, ориентированные на людей.

4) **Физические средства защиты**. Сюда относится экранирование помещений, проверка поставляемой аппаратуры, средства наружного наблюдения, замки, сейфы и т.д..

5) **Программно-технические меры** направлены на контроль компьютерных сущностей, реализуются программным и аппаратным обеспечением узлов и сети.

Наибольший положительный эффект достигается в том случае, когда все перечисленные способы применяются комплексно.

Законодательный уровень является важнейшим для обеспечения ИБ.

В 1983 году **Международная организация экономического сотрудничества и развития** определила термин **«компьютерная преступность»** (или «связанная с компьютерами преступность») как любые незаконные, неэтичные или неправомерные действия, связанные с автоматической обработкой данных и/или их передачей.

В настоящее время компьютерные преступления - это самостоятельный вид преступности.

2001 г. – принята Европейская конвенция о компьютерной преступности.

Основные правовые акты, регламентирующие защиту информации в Республике Беларусь:

1) Конституция РБ.

2) Гражданский кодекс РБ (28 декабря 2009г. №114-З)

Административно-правовые санкции, связанные с информацией:

Статья 9.6 Отказ госоргана в предоставлении гражданину информации.

Статья 22.6 Несанкционированный доступ к компьютерной информации

Статья 22.7 Нарушение правил защиты информации

Статья 22.10 Незаконный отказ в доступе к архивному документу

Статья 22.13 Незаконное разглашение коммерческой или иной тайны

3) Уголовный кодекс РБ. Глава 31. Преступления против информационной безопасности. Статьи 349-355

Статья 349. Несанкционированный доступ к компьютерной информации

Статья 350. Модификация компьютерной информации

Статья 351. Компьютерный саботаж

Статья 352. Неправомерное завладение компьютерной информацией

Статья 353. Изготовление либо сбыт специальных средств для получения неправомерного доступа к компьютерной системе или сети

Статья 354. Разработка, использование либо распространение вредоносных программ

Статья 355. Нарушение правил эксплуатации компьютерной системы или сети

4) Закон РБ «Об информации, информатизации и защите информации» 2008г., №455-З

5) Закон РБ «Об электронном документе», 2006г., №357-З

6) Закон РБ «Об электронном документе и ЭЦП», 2009г., №113-З

7) Закон РБ «О государственных секретах», 19.07.2010г., №170-З

8) О мерах по совершенствованию использования национального сегмента сети Интернет: Указ Президента РБ, 1.02.2010г., №60

9) О некоторых вопросах в сфере государственных секретов: Указ Президента РБ, 25.02.2011г., №68

10) Об утверждении положения о лицензировании деятельности по технической защите информации, в том числе криптографическими методами, включая применение электронной цифровой подписи: постановление Совета Министров РБ, 20.10.2003г., №1374; в ред. постановление Совета Министров РБ, 12.07.2008г., №1012

11) Об утверждении положения о порядке выполнения работ (оказания услуг) по технической защите информации: Приказ Государственного центра безопасности информации при Президенте РБ, 26 июля 2007г., №80

В современном мире глобальных сетей нормативно-правовая база должна быть согласована с международной практикой. Законы и стандарты страны должны соответствовать международному уровню.

Роль стандартов и спецификаций. Наиболее важные стандарты и спецификации в области информационной безопасности

Знание стандартов и спецификаций важно для специалистов в области информационной безопасности по целому ряду причин:

- необходимость следования некоторым стандартам (например, криптографическим) закреплена законодательно.

- стандарты и спецификации - одна из форм накопления знаний, прежде всего о процедурном и программно-техническом уровнях ИБ. В них

зафиксированы апробированные, высококачественные решения и методологии, разработанные наиболее квалифицированными специалистами.

- стандарты и спецификации являются основным средством обеспечения взаимной совместимости аппаратно-программных систем и их компонентов, причем в Internet-сообществе это средство действительно работает, и весьма эффективно.

Количество стандартов и спецификаций в области ИБ велико.

Международные стандарты являются основой разработки национальных стандартов. Это означает, что знание международных стандартов требуется для понимания направлений развития отечественной нормативной базы, что, в свою очередь, важно для выработки стратегии построения и совершенствования информационных систем.

Две существенно отличающиеся друг от друга ***группы стандартов и спецификаций***:

- оценочные стандарты, предназначенные для оценки и классификации информационных систем и средств защиты по требованиям безопасности;

- технические спецификации, регламентирующие различные аспекты реализации и использования средств защиты.

Обе группы дополняют друг друга. Оценочные стандарты выделяют важнейшие понятия и аспекты ИС, играя роль организационных и архитектурных спецификаций. Другие спецификации определяют, как строить ИС предписанной архитектуры.

Из числа оценочных необходимо выделить:

1) стандарт Министерства обороны США "Критерии оценки доверенных компьютерных систем" –«Оранжевая книга» (1983г.) и его интерпретацию для сетевых конфигураций – «Красная книга» (1987г.);

2) "Гармонизированные критерии Европейских стран" (1991г., Великобритания, Франция, Германия, Нидерланды);

3) Международный стандарт ISO "Критерии оценки безопасности информационных технологий" (точнее, его первоисточник - "**Общие критерии**" – 1.12.1999г.) и разработанные на его основе профили защиты*;*

4) Руководящие документы Гостехкомиссии России;

5) Федеральный стандарт США "Требования безопасности для криптографических модулей", регламентирующий конкретный, но очень важный и сложный аспект информационной безопасности.

Основным источником технических спецификаций, применимых к современным распределенным ИС, является Internet-сообщество, главным образом, "Тематическая группа по технологии Internet" (Internet Engineering Task Force, IETF) и ее подразделение - рабочая группа по безопасности.

Ядром рассматриваемых технических спецификаций служат документы по безопасности на IP-уровне (IPsec). Кроме этого, анализируется защита на транспортном уровне (Transport Layer Security, TLS), а также на уровне приложений (спецификации GSS-API, Kerberos).

Internet-сообщество уделяет внимание не только программно-техническому, но также административному и процедурному уровням информационной безопасности. ("Руководство по информационной безопасности предприятия", "Как выбирать поставщика Интернет-услуг", "Как реагировать на нарушения информационной безопасности"). Комплексность подхода Internet-сообщества к проблемам ИБ проявляется и в том, что в спецификациях рассматриваются как профилактические меры, направленные на предупреждение нарушений ИБ, так и меры реагирования на нарушения.

ГОСТ - межгосударственный стандарт, который действует в рамках стран СНГ.

Белорусские государственные криптографические стандарты

СТБ 34.101.27-2022 "Информационные технологии и безопасность. Средства криптографической защиты информации. Требования безопасности".

СТБ 34.101.31-2020 "Информационные технологии и безопасность. Алгоритмы шифрования и контроля целостности".

СТБ 34.101.45-2013 "Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых".

СТБ 34.101.47-2017 "Информационные технологии и безопасность. Алгоритмы генерации псевдослучайных чисел".

СТБ 34.101.60-2014 "Информационные технологии и безопасность. Алгоритмы разделения секрета".

СТБ 34.101.65-2014 "Информационные технологии и безопасность. Протокол защиты транспортного уровня (TLS)".

СТБ 34.101.66-2014 "Информационные технологии и безопасность. Протоколы формирования общего ключа на основе эллиптических кривых".

СТБ 34.101.77-2020 "Информационные технологии и безопасность. Криптографические алгоритмы на основе sponge-функции". (Стандарт хеширования)

СТБ 34.101.87-2022 "Информационные технологии и безопасность. Инфраструктуры аутентификации".

Политика информационной безопасности. Организационные и физические меры обеспечения ИБ

К административному уровню ИБ относятся действия общего характера, предпринимаемые руководством организации для обеспечения ИБ.

Главная цель мер административного уровня – сформировать Политику ИБ и обеспечить ее выполнение, выделяя необходимые ресурсы и

контролируя состояние дел. Политика безопасности отражает подход организации к защите своих информационных ресурсов. Политика безопасности строится на основе анализа рисков, которые признаются реальными для информационной системы организации.

Политика безопасности – это набор правил, используемых для принятия решений, касающихся вопросов безопасности.

Политика безопасности может быть определена на уровне предприятия, сети, компьютера, пользователя, приложения. На уровне компьютера или сети ПБ устанавливает администратор.

Политика безопасности (на уровне предприятия) – это совокупность документированных решений, принимаемых руководством организации и направленных на защиту информации от заданного множества угроз безопасности.

Политика безопасности устанавливает правила, которые определяют:

- источники угроз, виды атак;
- какой ущерб принесет предприятию потеря или раскрытие определенных данных (анализ рисков);
- какую информацию защищать и от кого;
- какие средства использовать для защиты каждого вида информации;
- настройку компьютерных систем и сетей;
- кто и когда имеет доступ к системе;
- действия служащих в обычных условиях и в случае непредвиденных обстоятельств и т.д...

Т.о., политика выполняет две основные функции:

- определяет безопасность внутри организации;
- определяет место каждого служащего в системе безопасности.

При выработке политики безопасности возможны два подхода:

- запрещать все, что не разрешено в явной форме (высокая степень безопасности, дорого, неудобства пользователей);
- разрешать все, что не запрещено в явной форме (менее защищенная сеть, дешевле, удобна для пользователей).

С практической точки зрения политику безопасности целесообразно рассматривать на трех уровнях детализации.

К верхнему уровню можно отнести решения, затрагивающие организацию в целом. Они носят весьма общий характер. На этом уровне цели организации в области ИБ формулируются в терминах доступности, целостности и конфиденциальности.

Как стартовая точка при разработке программ безопасности используется Британский стандарт BS 7799 (1995г., 2005г.) и разработанный на его основе стандарт ISO/IEC 17799 (2005г.). Стандарт одинаково применим ко всем организациям и всем структурным подразделениям внутри компании.

К среднему уровню политики безопасности можно отнести вопросы, касающиеся отдельных аспектов ИБ, но важные для различных эксплуатируемых организацией систем. Примеры таких вопросов – отношение

к передовым технологиям, доступ в Интернет, использование домашних компьютеров, применение пользователями неофициального ПО и т.д..

Политика безопасности нижнего уровня относится к конкретным информационным сервисам: политика идентификации и аутентификации, управления доступом, политика аудита, контроля сетевых соединений, политика использования Интернета, политика работы с электронной почтой, политика управления пользователями, политика резервного копирования (частота, место хранения, резервируемая информация), процедуры обработки инцидентов и т.д.

На процедурном уровне рассматриваются меры безопасности, ориентированные на людей, а не на технические средства.

На процедурном уровне можно выделить следующие классы мер:

- управление персоналом;
- поддержание работоспособности;
- реагирование на нарушения режима безопасности;
- планирование восстановительных работ.

Т.о. это меры, регламентирующие действия сотрудников, начиная с должностных инструкций.

Основной принцип физической защиты – непрерывность защиты в пространстве и времени. Для физической защиты окон опасности быть не должно.

Основные направления физической защиты:

- физическое управление доступом;
- противопожарные меры;
- защита поддерживающей инфраструктуры;
- защита от перехвата данных;
- защита мобильных систем.

К средствам физической защиты относятся: охрана, замки, сейфы, перегородки, телекамеры, датчики движения и многое другое. Для выбора оптимального средства целесообразно провести анализ рисков. Есть смысл периодически отслеживать появление технических новинок в данной области, стараясь максимально автоматизировать физическую защиту.

Основные программно-технические меры

Программно-технические меры - это меры, направленные на контроль компьютерных сущностей – оборудования, программ и данных, образуют последний и самый важный рубеж ИБ.

Программно-технические меры реализуются программным и аппаратным обеспечением узлов и сети.

Центральным для программно-технического уровня является понятие **сервиса безопасности**.

К основным сервисам (или функциям) безопасности относятся:

- идентификация и аутентификация;
- управление доступом;
- протоколирование и аудит;
- конфиденциальность;
- контроль целостности;
- экранирование;
- анализ защищенности;
- обеспечение отказоустойчивости;
- обеспечение безопасного восстановления;
- туннелирование;
- управление.

Программно-технические меры безопасности можно разделить на следующие **виды**:

- превентивные, препятствующие нарушениям ИБ;
- меры обнаружения нарушений;
- локализующие, сужающие зону воздействия нарушений;
- меры по выявлению нарушителя;
- меры восстановления режима безопасности.

С практической точки зрения наиболее важными являются следующие **базовые принципы архитектурной безопасности**:

- непрерывность защиты в пространстве и времени, невозможность миновать защитные средства;
- принцип единого контрольно-пропускного пункта;
- следование признанным стандартам, использование апробированных решений;
- усиление самого слабого звена;
- невозможность перехода в небезопасное состояние (использование средств, которые при отказе переходят в состояние максимальной защиты);
- минимизация привилегий;
- разделение обязанностей;
- эшелонированность обороны;
- разнообразие защитных средств или использование комплексного подхода к обеспечению безопасности;
- простота и управляемость ИС;
- минимизация объема защитных средств, выносимых на клиентские системы, так как конфигурацию клиентских систем трудно или невозможно контролировать;
- принцип баланса возможного ущерба от реализации угрозы и затрат на ее предотвращение.

2.3 Введение в криптографию

Криптография как наука

Криптография – это наука о шифрах.

Криптография – это наука о методах преобразования (шифрования) информации с целью ее защиты от несанкционированных пользователей.

Слово «криптография» (cryptography) происходит от греческих слов «kryptus» - тайный, «graphein» - писать, т.е. дословно «тайнопись».

Криптоанализ – это наука, изучающая методы взлома шифров.

Криптографию и криптоанализ иногда называют науками-двойниками. И действительно, на практике они взаимно дополняют друг друга: то, что одна наука создает, другая разрушает, и наоборот.

Криптология – наука, изучающая шифры и их стойкость.

Криптология = Криптография + Криптоанализ

Период развития криптологии с древних времен до 1948 г. принято называть эрой донаучной криптологии, поскольку достижения тех времен основаны на интуиции и не подкреплялись математическими доказательствами. Криптологией занимались тогда почти исключительно как искусством, а не как наукой.

После Второй мировой войны была разработана «теория информации». Предметом ее изучения являются математические законы, которым подчиняются системы передачи данных. Созданная для решения проблем телефонии и телеграфии, она оказалась применима практически ко всем устройствам, передающим информацию, включая компьютеры и нервную систему животных. Ее идеи оказались настолько плодотворными, что были взяты на вооружение другими науками – психологией, лингвистикой, молекулярной генетикой, историей, статистикой и нейрофизиологией. Создатель этой теории - американский ученый Клод Элвуд Шеннон (1916-2001). Основы теории информации были изложены им в статье «Теория связи в секретных системах».

Благодаря работам Клода Шеннона криптография была поставлена на научную основу. Его доклад «Математическая теория криптографии» был подготовлен в секретном варианте в 1945 году, рассекречен и опубликован в 1948 году, переведен на русский язык в 1963г.

Публикации Шеннона стали началом эры научной криптологии с секретными ключами. В этих блестящих работах Шеннон связал криптографию с теорией информации. Он изложил схему секретной системы связи с точки зрения обычного канала связи. Первая статья породила теорию информации, а вторая – теорию шифров.

В 70-х годах произошло два события, серьезно повлиявших на дальнейшее развитие криптографии. Во-первых, был принят (и опубликован!)

первый стандарт шифрования данных (DES). Во-вторых, после работы американских математиков У.Диффи и М.Хеллмана родилась «новая криптография» - криптография с открытым ключом. Оба этих события были рождены потребностями бурно развивающихся средств коммуникаций, в том числе локальных и глобальных компьютерных сетей, для защиты которых потребовались легко доступные и достаточно надежные криптографические средства.

Как передать нужную информацию нужному адресату в тайне от других?

Размышляя над **задачей Тайной Передачи**, можно прийти к выводу, что есть 3 возможности:

1. Создать абсолютно надежный канал связи между абонентами.
2. Использовать общедоступный канал связи, но скрыть сам факт передачи.
3. Использовать общедоступный канал связи, но передавать по нему нужную информацию в таком виде, чтобы восстановить ее мог только адресат, знающий секрет.

Разберем эти возможности.

1. При современном уровне развития науки и техники сделать такой канал связи между удаленными абонентами для неоднократной передачи больших объемов информации практически нереально.
2. Разработкой методов, предназначенных для сокрытия факта существования сообщения, занимается **стеганография**.

Первые следы стеганографических методов теряются в глубокой древности. (Голову раба брили, на коже головы писали сообщение, после отрастания волос раба отправляли к адресату.

На пример:

1. Расстановка слов или букв открытого текста так, чтобы он нес в себе скрытое сообщение.
2. Пометка символов.
3. Невидимые чернила.
4. Прокалывание бумаги.
5. Печать между строк с использованием корректирующей ленты.

Сами по себе подобные методы выглядят архаично, но у них есть современные эквиваленты. Например, сокрытие сообщений с помощью использования незначащих бит в видеокадрах компакт-дисков.

В настоящее время разработано множество программных пакетов, позволяющих осуществлять подобные операции. Преимущество стеганографии состоит в том, что она может скрывать сам факт передачи сообщений, а не только их содержимое. Ведь само шифрование вызывает подозрение – значит есть что скрывать.

3. Разработкой методов преобразования информации с целью ее защиты от несанкционированных пользователей занимается **криптография**.

Основные понятия криптографии

Хотелось бы обратить особое внимание на вопросы терминологии.

У каждой отрасли знания, науки есть свой словарь. Существует общепринятая криптографическая терминология. В основном это англоязычные термины.

Имеются не очень хорошие переводы их на русский язык. Десятки вариантов одного термина, в т.ч. неправильных («шифрация», «криптование», «дешифрирование», «кодирование»).

На сегодняшний день в России существует одна из самых сильных криптографических школ в мире – наследие СССР. Советские криптоаналитики еще долго будут считаться одними из самых сильных специалистов в этой области. Дэвид Кон в книге «Выдающиеся взломщики шифров» пишет: «Русские вознесли достижения своей страны в криптологии до высоты полета ее космических спутников».

Соответственно наработана и терминология. Существуют устоявшиеся русские эквиваленты английских терминов. Например, код аутентичности – имитовставка. Тем не менее вся эта информация долгое время была конфиденциальной и строжайше охранялась. Только в 90-е годы ситуация изменилась.

Криптографию можно рассматривать как способ сохранения больших секретов (которые неудобно хранить в тайне из-за их размеров) при помощи секретов малых (которые прятать проще и удобней). Под большими секретами имеется в виду открытый текст, а малые секреты обычно называют криптографическими ключами. («Криптография и безопасность в технологии .NET» Торстейнсон, Ганеш)

Краеугольный камень криптографии – шифрование.

Шифрование –обратимое преобразование информации с помощью одного из алгоритмов и ключа. По-русски - **зашифрование (encryption).**

Исходное сообщение называется **открытым текстом (plaintext).** Зашифрованное сообщение называется **шифром**, **шифротекстом (ciphertext)**.

Процесс шифрования включает две составляющие:

1. Алгоритм шифрования.
2. Ключ (или ключи) – значение, не зависящее от открытого текста.

Результат, достигаемый при выполнении алгоритма, зависит от применяемого ключа. Изменение ключа приводит к изменению шифрованного текста.

Пространство ключей – множество всех возможных ключей, доступных для использования в алгоритме.

Криптограмма - окончательно обработанное и отосланное секретное сообщение. Термин «шифртекст» обращает внимание на результат зашифрования, в то время как термин «криптограмма» подчеркивает сам факт передачи сообщения и является аналогом слова «телеграмма».

Любой алгоритм шифрования должен быть дополнен алгоритмом расшифрования, чтобы привести зашифрованный текст в исходный вид.

Расшифрование (decryption) - проведение обратных преобразований шифртекста лицами, владеющими на законном основании ключом и системой шифрования (кодирования), для получения открытого текста.

Пара процедур – зашифрование и расшифрование – называется **криптосистемой.**

Процесс расшифрования следует отличать от дешифрования или криптоанализа, который ставит своей целью получение открытого текста людьми, не имеющими в своем распоряжении ни ключа, ни алгоритма шифрования, то есть лицами, являющимися третьей стороной, «противником». Разница между ними огромная, хотя начиная с того времени, когда возникло слово «криптоанализ», термины «расшифровать» или «раскодировать» часто использовались и в смысле «дешифровать».

Дешифрование (deciphering) – (взлом, вскрытие шифра) восстановление исходного текста без знания ключа в процессе криптоанализа шифротекста.

Сложность алгоритма раскрытия является одной из важных характеристик криптосистемы и называется **криптостойкостью.**

Успешный криптоанализ шифра часто называют его **вскрытием или взломом.** Попытку раскрытия шифра с применением методов криптоанализа называют **криптографической атакой**.

Надежность шифрования определяет нескольких факторов:

1. Сложность алгоритма шифрования (должен быть достаточно сложным, чтобы невозможно было расшифровать сообщение при наличии только шифрованного текста).
2. Секретность ключа – основной фактор надежности традиционного шифрования. Сам алгоритм может быть несекретным.

В традиционной (классической) криптографии принято фундаментальное правило, сформулированное в 19 веке - **правило Керкхоффа:**

Стойкость шифра должна определяться только секретностью ключа.

На первый взгляд кажется, что секретный алгоритм мог бы сделать шифр более стойким. Но это не так. Это наивное предположение. Идею использования секретного алгоритма называют «секретностью через скрытность». Вы прячете ценный предмет в небезопасное, но скрытое место, например, под матрац. Секретность, основанная на хорошо известном алгоритме, дает более высокую степень безопасности. Вы прячете ценный предмет в сейф, который всем виден, но надежно защищен от взлома.

Эта особенность традиционного шифрования обусловливает его широкую популярность и признание. Т.к. нет необходимости хранить в секрете алгоритм, то производители могут реализовать алгоритмы шифрования в виде дешевых общедоступных микросхем, которыми оснащены многие современные системы.

Типы криптосистем

Классификация криптографических систем строится на основе следующих трех характеристик:

1) Число применяемых ключей.
2) Тип операций по преобразованию открытого текста в шифрованный.
3) Метод обработки открытого текста.

По числу применяемых ключей.

Различают:

1) Симметричные криптосистемы;
2) Асимметричные криптосистемы.

Если отправитель и получатель используют один и тот же ключ, система шифрования называется **симметричной, системой с одним ключом, системой с секретным ключом, схемой традиционного шифрования.** Ключ, который используется, называется **секретным ключом. (**Например, DES, CAST, RC5, IDEA, Blowfish, классические шифры);

Если отправитель и получатель используют разные ключи, система называется **асимметричной, системой с двумя ключами, схемой шифрования с открытым ключом.** Ключи при этом называются **открытый и личный.** (RSA, Эль-Гамаля, Диффи-Хеллмана).

По типу операций по преобразованию открытого текста в шифрованный.

- **Подстановочные шифры (замены)** - шифрование основано на замещении каждого элемента открытого текста (бита, буквы, группы битов или букв) другим элементом. (Цезаря, Плейфейера, Хилла).

Различают **моноалфавитные замены** – шифры, в которых символы алфавита открытого текста заменяются на символы того же алфавита по определенному правилу, зависящему от ключа.

Полиалфавитные замены – символы алфавита открытого текста заменяются на символы того же либо другого алфавита

- **Перестановочные шифры** – шифрование основано на изменении порядка следования элементов открытого текста. (Лесенка, перестановка столбцов, «Магический квадрат»). Элементы переставляются по определенному алгоритму, зависящему от ключа. Перестановочные шифры являются простейшими и самыми древними.
- **Композиционные шифры** – шифрование основано на комбинации нескольких операций замены и перестановки. Композиционные шифры применяются в большинстве реальных современных систем шифрования. (DES).

По методу обработки открытого текста.

- **Блочные шифры** – Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после преобразования которого получается блок шифрованного текста такой же длины.

На пример: DES, шифр Файстеля.

- **Поточные шифры** – подразумевают шифрование всех элементов открытого текста последовательно, одного за другим (бит за битом, байт за байтом).

Примерами классических поточных шифров являются шифры Виженера (с автоматическим выбором ключа) и Вернама.

Блочные шифры изучены гораздо лучше. Считается, что они обладают более широкой областью применения, чем поточные. Большинство сетевых приложений, в которых применяется схема традиционного шифрования, используют блочные шифры.

В большинстве современных поточных систем результат шифрования текущего блока открытого текста зависит от его номера и не зависит от самих предыдущих блоков. Такие симметричные криптосистемы называют **шифрами гаммирования**. В этом случае стойкость системы определяется исключительно свойствами гаммы, являющейся обычно псевдослучайной числовой последовательностью.

Модель традиционного шифрования. Классические методы шифрования

По числу применяемых ключей различают:

3) Симметричные (традиционные) криптосистемы;
4) Асимметричные криптосистемы.

Рассмотрим **модель традиционной криптосистемы** более подробно.

Теоретические основы традиционной модели симметричной криптосистемы впервые были изложены в 1949 году в работе Клода Шеннона «Теория связи в секретных системах».

Источник создает сообщение в форме открытого текста:

$$P=[p_1, p_2, \ldots, p_m]$$

Текст – упорядоченный набор элементов.

Элементами p_i открытого текста P являются символы некоторого конечного алфавита A, состоящего из n символов:

$$p_i \in A$$

Традиционно использовался алфавит из 26 букв английского языка, но сегодня чаще применяется двоичный алфавит {0,1}.

Для шифрования генерируется ключ в форме:

$$K=[к_1, к_2, \ldots, к_j]$$

Если ключ генерируется там же, где и само сообщение, то его необходимо переправлять получателю по секретным каналам. Либо ключ создается третьей стороной, которая должна защищенным способом обеспечить доставку ключа отправителю и получателю сообщения.

Имея P и K с помощью алгоритма шифрования фомируется шифрованный текст C

$$C=[c_1, c_2, \ldots, c_n]$$

Это можно записать в виде формулы:

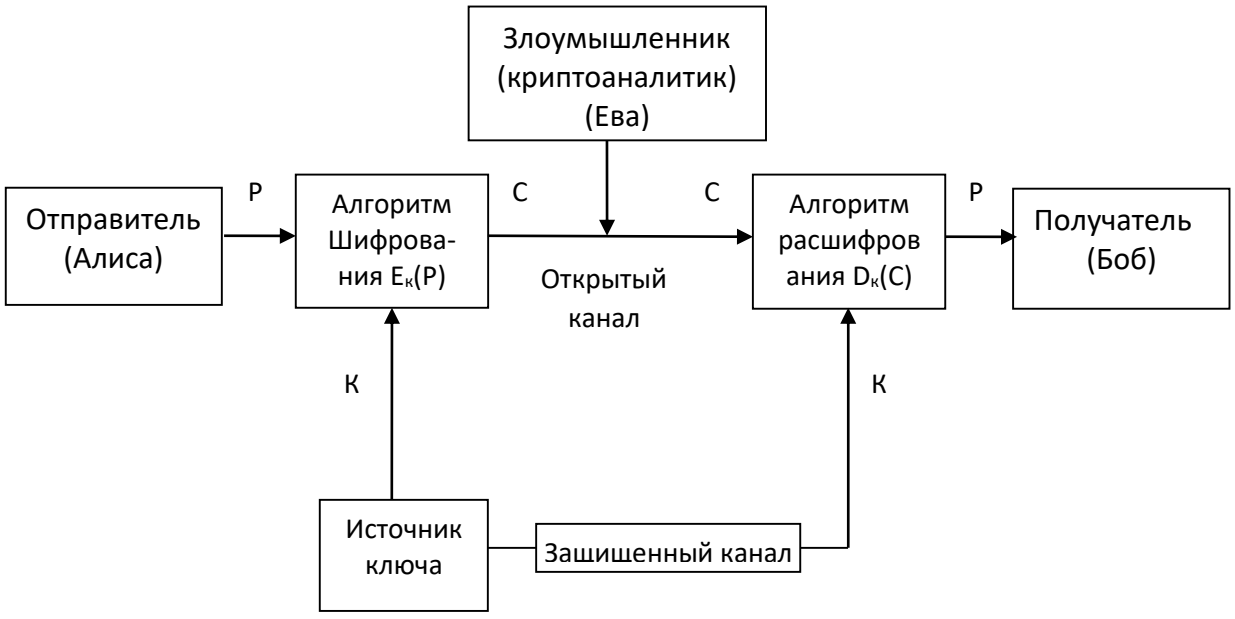
$$C = E_k(P)$$

У получается путем применения алгоритма шифрования E к открытому тексту P при использовании ключа K.

Обратное преобразование:

$$P = D_k (C)$$

Это математические обозначения, принятые в литературе по криптографии. Прелесть математических обозначений в их точности и компактности. Единственная проблема – для некоторых людей математика ничем не лучше головной боли.



Для того, чтобы сделать этот сценарий более жизненным и наглядным, в криптографической литературе отправителя и получателя часто называют Алисой и Бобом, а атакующую сторону представляет злой персонаж Ева. В более сложных сценариях вводятся дополнительные персонажи.

Шифры, которые использовались в докомпьютерную эпоху, называют **классическими.**

Знакомство с криптографией мы начнем с изучения классических шифров. Когда-то они были одними из самых надежных средств сокрытия информации, а в настоящее время представляют только академический интерес. Тем не менее они представляют интерес с точки зрения образовательного процесса.

Наша задача – освоить базовые понятия криптографии, что проще сделать на примере простых классических алгоритмов. Это поможет пониманию более сложных криптографических систем, которые мы будем рассматривать далее. Многие современные методы построения одноключевых шифросистем построены на основе методов, известных сотни лет. Да и некоторые методы современного криптоанализа имеют много общего с исторически известными способами взлома старинных шифров.

Самым древним и самым простым из известных подстановочных шифров является **шифр Юлия Цезаря**. В шифре Цезаря каждая буква алфавита заменяется буквой, которая находится на 3 позиции дальше в этом же алфавите.

В общем случае сдвиг может быть любым (от 1 до 25). Если каждой букве назначить числовой эквивалент, то:

$$C = E_k(P);\ \ c_i = (p_i + k)\ \mathrm{mod}26$$

Получаем обобщенный алгоритм Цезаря.

Шифр Цезаря со сдвигом на 13 букв вправо обозначается rot13.

Расшифрование:

$$p_i = (c_i - k)\ \mathrm{mod}26.$$

Раскрыть такой шифр можно путем простого последовательного перебора вариантов. Это **атака методом «грубой силы»** или **лобовая атака** - способ раскрытия шифра, при котором поиск ведется по всему возможному пространству ключей.

Применение простого перебора оправдано следующим:

1) известны алгоритмы шифрования и расшифрования;
2) всего 25 вариантов;
3) известен язык открытого текста.

В большинстве случаев при защите сетей выполняется 1), но не 2). Важная характеристика – 3). Если исходный текст предварительно сжат, то это

сильно затрудняет распознавание. (Например, протокол SSL)

Для криптоаналитика существует и другая возможность раскрытия шифра (кроме перебора вариантов). Если криптоаналитик имеет представление о природе открытого текста, можно использовать информацию о характерных признаках, присущих текстам на соответствующем языке (статистические методы). Например, анализ частот появления букв алфавита – частотный анализ.

Моноалфавитные шифры легко раскрываются, т.к. они наследуют частотность употребления букв оригинального алфавита. Чтобы в тексте, шифрованном с помощью методов подстановок, структура исходного текста проявлялась менее заметно, можно использовать:

- многобуквенное шифрование (полиграммные шифры), т.е. замещение не отдельных символов открытого текста, а комбинаций нескольких символов;
- несколько алфавитов.

Один из наиболее известных шифров, базирующихся на методе **многобуквенного шифрования**, - **шифр Плейфейера (Playfair).** В нем биграммы (комбинации из двух букв) открытого текста рассматриваются как самостоятельные единицы, преобразуемые в заданные биграммы шифрованного текста. Анализ частот биграмм сложнее. Долго считалось, что шифр Плейфейера взломать нельзя. Он служил стандартом шифрования в британской армии во время 1-й мировой войны и даже использовался в годы 2-й мировой войны в США.

Примером **многобуквенного шифра** является также **шифр Хилла** (1929г.). В его основе лежит алгоритм, который заменяет каждые m последовательных букв открытого текста m буквами шифрованного текста.

Одна из возможностей усовершенствования простого моноалфавитного шифра заключается в использовании нескольких моноалфавитных подстановок в зависимости от определенных условий. Семейство шифров, основанных на применении таких методов шифрования, называется **полиалфавитными шифрами.** Они обладают следующими свойствами:

- используется набор связанных моноалфавитных подстановок;
- имеется ключ, по которому определяется, какое конкретное преобразование должно применяться для шифрования на данном этапе.

Полиалфавитные подстановочные шифры были придуманы в 15-м веке. Леоном Баттистой-Альберти. Они пользовались популярностью до 19 века, когда их начали взламывать. Самым известным и простым алгоритмом такого рода является **шифр Виженера** (поточный).

Лучшей защитой от статистических методов криптоанализа (т. е. по частоте появления букв) является выбор ключевого слова, по длине равного длине открытого текста, но отличающегося от открытого текста по статистическим показателям. Такая система была предложена инженером компании AT&T Гилбертом Вернамом в 1918 году. В **алгоритме Вернама** шифрованный текст генерируется путем побитового выполнения операции XOR для открытого текста и ключа.

$$c_i = p_i \oplus k_i$$

$\oplus$ - «исключающее ИЛИ», XOR.
$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$
Операция XOR обратима относительно себя самой

$$p_i = ((p_i \oplus k_i) \oplus k_i)$$

Типы криптоатак и стойкость алгоритмов

Может ли существовать идеальный шифр, который невозможно взломать? Хотите верьте, хотите нет, но абсолютный метод шифрования существует.

Джозеф Моборн придумал улучшение схемы Вернама, которое сделало ее исключительно надежной. Он предложил 3 усовершенствования.

1) Длина ключа равна длине сообщения.
2) Ключ генерируется случайным образом.
3) Каждый ключ можно использовать только один раз.

На выходе получается случайная последовательность, не имеющая статистической связи с открытым текстом. Схема получила название **лента однократного использования** или **одноразовый блокнот.** Взлому она не поддается, т.к. шифртекст не содержит никакой информации об открытом тексте. Однако использование одноразового блокнота затруднено, потому что отправитель и получатель должны располагать одним случайным ключом, что трудно реализуемо на практике.

Понятие идеального шифра ввел в своих работах Шеннон. Он ставил задачу воспрепятствовать попыткам криптоанализа, основанным на статистических методах.

Идеальный шифр – это шифр, который полностью скрывает в шифрованном тексте все статистические закономерности открытого текста.

Схема шифрования называется **абсолютно стойкой (безусловно защищенной)**, если шифрованный текст не содержит информации, достаточной для однозначного восстановления открытого текста.

Это означает, независимо от того, сколько времени потратит противник на расшифровку текста, ему не удастся расшифровать его просто потому, что в шифрованном тексте нет информации, требуемой для восстановления открытого текста.

К. Шеннон ввел понятие **расстояние уникальности** – это необходимое количество зашифрованного текста, позволяющее однозначно воспроизвести открытый текст.

Шифры, не являющиеся абсолютно стойкими, называют **несовершенными**.

Максимум, что может дать алгоритм шифрования, это выполнение хотя бы одного из следующих условий:

a. Стоимость взлома шифра превышает стоимость расшифрованной информации;

b. Время, которое требуется для взлома шифра, превышает время, в течение которого информация актуальна.

Если схема шифрования соответствует обоим указанным критериям, она называется **защищенной по вычислениям.**

Желательным свойством любого алгоритма шифрования должна быть высокая чувствительность результата к изменению начальных данных — любые малые изменения открытого текста или ключа должны приводить к значительным изменениям в шифрованном тексте (**лавинный эффект**).

Например, алгоритм DES демонстрирует достаточно сильный лавинный эффект. (Изменение одного бита открытого текста приводит к изменению 34 битов шифрованного текста).

Попытку криптоанализа системы шифрования называют атакой.

Под взломом (т.е. успешном криптоанализом) системы шифрования принято понимать нахождение «уязвимости» шифра, позволяющей проводить атаку со сложностью меньшей, чем при переборе всех ключей.

При анализе новых шифров и доказательстве их криптографической стойкости используют различные виды криптоатак.

Основные типы криптоатак:

1) Лобовое вскрытие (лобовая атака, метод «грубой силы» – путем перебора всех возможных ключей). Если множество ключей достаточно большое, то подобрать ключ нереально. При длине ключа n бит количество возможных ключей равно 2^n. Таким образом, чем длиннее ключ, тем более стойким считается алгоритм для лобовой атаки.

В 2000г. хорошей длиной ключа считалось 128 бит, в 2015г. – уже 256 бит. В 1999г. распределенные вычисления для взлома DES проверяли 250 млрд. ключей в секунду.

2) Атака на основе шифртекста.

Криптоаналитик располагает шифртекстами нескольких сообщений, зашифрованных одним и тем же алгоритмом шифрования и возможно ключом.

3) Атака на основе открытого текста.

Криптоаналитик располагает доступом не только к шифртекстам нескольких сообщений, но и открытому тексту этих сообщений. Его задача состоит в определении ключа или ключей, чтобы он мог расшифровать другие сообщения.

4) Атака на основе подобранного открытого текста (криптоаналитик может выбирать открытый текст для шифрования).

Во многих сообщениях используются стандартные начала и окончания, которые могут быть известны криптоаналитику (атаки 2, 3).

В этом отношении особенно уязвимы шифрованные исходные коды программ из-за частого использования ключевых слов (define, struct, else, return и т.д.). Те же проблемы и у шифрованного исполняемого кода: функции, циклические структуры и т.д..

5) Бандитский криптоанализ.

Для получения ключа “криптоаналитик” прибегает к угрозам, шантажу или (не приведи, Господи!) пыткам. Возможно также взяточничество, которое называется вскрытием с покупкой ключа. Это очень мощные и, зачастую, самые эффективные методы взлома алгоритма.

Все формы криптоанализа для схем традиционного шифрования разрабатываются на основе того, что некоторые характерные особенности структуры открытого текста могут сохраняться при шифровании, проявляясь в шифрованном тексте.

Криптоанализ для схем с открытым ключом базируется на совершенно другом – на том, что математические зависимости, связывающие пары ключей, дают возможность с помощью логических рассуждений, зная один из ключей, найти второй.

2.4 Основы криптографии с секретным ключом

Сети Файстеля

Блочный шифр предполагает преобразование n-битового блока открытого текста в блок шифрованного текста такого же размера. Число вариантов различных блоков при этом равно 2^n. Чтобы шифрование было обратимым (т.е. чтобы обеспечивалась возможность расшифрования), каждый из таких блоков должен преобразовываться в свой уникальный блок шифрованного текста.

Преобразование, при котором каждый блок открытого текста преобразуется в уникальный блок шифрованного текста, называются **обратимыми**, **несингулярными** или **мономорфизмом**. Вот примеры несингулярного и сингулярного преобразований для $n = 2$.

Обратимое (несингулярное) отображение			Необратимое (сингулярное) отображение		
Открытый текст		Шифрованный текст	Открытый текст		Шифрованный текст
00	⇔	11	00	⇔	11
01	⇔	10	01	⇔	10
10	⇔	00	10	⇨	01
11	⇔	01	11	⇨	01

Шифр, в котором попеременно используются подстановки и перестановки называется продукционным (комбинационным).

Блочный шифр, который использует последовательность перестановок и подстановок, называют **сетью перестановок-подстановок** или **SP-сетью**. Большинство современных блочных алгоритмов, используемых в наши дни, относятся к так называемым **сетям Файстеля** (Feistel's network). Это алгоритмы DES, Lucifer, FEAL, Khufu, Khafre, LOKI, ГОСТ, CAST, Blowfish и др..

Шеннон предложил два важных свойства (метода), которыми должны обладать все криптографическистойкие системы **диффузию (рассеивание)** и **конфузию (перемешивание)**. Эти методы затрудняют криптоанализ:

Рассеивание (диффузия) – влияние любого элемента открытого текста или ключа на элементы шифртекста. Это свойство делает восстановление неизвестного ключа по частям трудной, а в идеале неразрешимой задачей.

Перемешивание (конфузия) – использование таких преобразований, которые усложняют поиск связи между открытым текстом и шифртекстом методами статистического анализа или делают его трудноосуществимым.

В блочных шифрах, имеющих дело с двоичными данными, диффузии можно достичь путем нескольких последовательных перестановок данных с последующим применением к результату перестановки некоторой функции — в итоге в формировании каждого бита шифрованного текста будет участвовать множество битов открытого текста.

Конфузия достигается использованием сложных подстановочных алгоритмов.

В итоге понятия диффузии и конфузии оказались настолько удачными с точки зрения описания сути желаемых характеристик блочных шифров, что эти термины стали базовыми для всех разработчиков современных шифров этого типа.

Большинство симметричных алгоритмов основано на **блочном шифре Файстеля"** (Feistel block cipher).

В начале 70-х г.г. для защиты электронной информации в сетях ЭВМ компания IBM финансировала научные исследования в области криптографии. Команду разработчиков возглавил Хорст Файстель, до того работавший с Клодом Шенноном.

Структура шифра Файстеля.

Шифрование состоит из этапов, называемых **раундами**.

P, P_i , K, K_i, n, L_i, R_i , $F(R_i , K_i)$

На вход раунда подается блок открытого текста P_i длиной n битов и подключ k_i. Блок открытого текста разделяется на две части L_0 и R_0 .

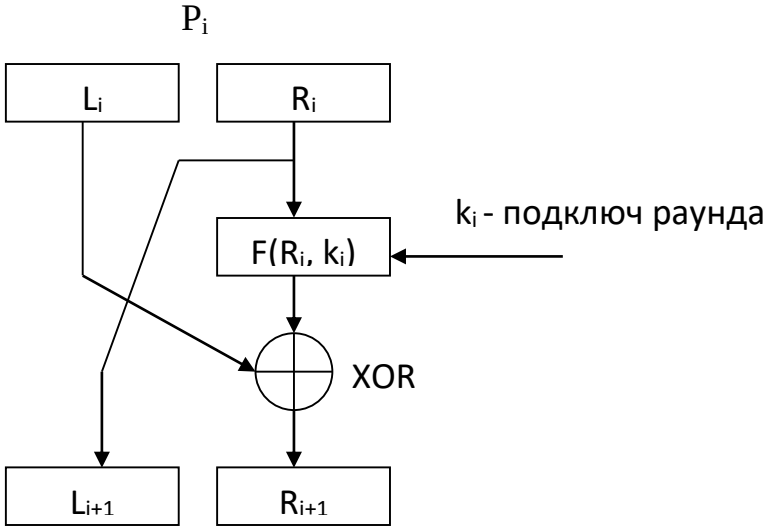


Рис. 1 i-й раунд шифрования

Все раунды обработки проходят по одной и той же схеме. Сначала выполняется операция подстановки.

К правой половине блока данных применяется функции раунда F и выполняется сложение полученного результата с левой половиной блока данных с помощью операции XOR (исключающего "ИЛИ"). Для всех раундов функция раунда имеет одну и ту же структуру, но зависит от параметра — подключа раунда (k_i).

Подключи раундов k_i отличаются от общего ключа K и друг от друга и вычисляются на основе общего ключа K.

После подстановки выполняется перестановка, представляющая собой обмен местами двух половин блока данных.

Т.о. результат (i+1)-го раунда определяется по результату i-го раунда:

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus F(R_i, K_i)$

Поскольку за один раунд обрабатывается только одна половина блока, желательно, чтобы число раундов было кратно двум.

Шифр, использующий такую конструкцию, гарантированно обратим. Можно спроектировать сколь угодно сложную функцию F, но 2 разных алгоритма для шифрования и расшифрования не нужны. Об этом автоматически позаботится структура сети Файстеля.

При расшифровании шифра Файстеля применяется тот же алгоритм, но на вход подается шифрованный текст, а подключи K_i используются в обратной последовательности.

Если размер левой части равен размеру правой, такую архитектуру называют **классической** или **сбалансированной сетью Файстеля**. Если левая и правая части не равны, то алгоритм называют **разбалансированной сетью Файстеля**.

Если параметры функции F изменяются после зашифрования каждого следующего символа, то сеть Файстеля называют **гетерогенной** (например, Khufru). Если параметры функции F не изменяются, то сеть называют **гомогенной**. Применение гетерогенных и разбалансированных сетей Файстеля может значительно улучшить характеристики шифра.

Криптоаналитическая стойкость сети Файстеля зависит от следующих параметров:

- **Размер блока.** Чем больше размер блока, тем выше надежность шифра (при прочих равных условиях), но тем ниже скорость выполнения операций шифрования и расшифрования. Разумным компромиссом является блок размером 64 бита, который является сегодня практически универсальным для всех блочных шифров (половинки по 32 бита – 32-битные процессоры).
- **Размер ключа.** Чем длиннее ключ, тем выше надежность шифра, но большая длина ключа тоже может быть причиной слишком медленного выполнения операций шифрования и расшифрования. На сегодня ключи длиной 64 бита и меньше считаются недостаточно надежными — обычно используются 128, 256 и 512-битовые ключи.
- **Число раундов обработки.** Чем больше число раундов шифрования, тем более затрудняется криптоанализ шифра. В общем случае число раундов должно выбираться так, чтобы все известные методы криптоанализа требовали больше усилий, чем анализ с помощью перебора всех ключей. Этот критерий использовался при разработке DES. Если бы DES использовал 15 раундов шифрования или меньше, то для дифференциального криптоанализа требовалось бы меньше усилий, чем для анализа с помощью перебора всех ключей. Обычно 16 раундов.
- **Алгоритм вычисления подключей.** Чем сложнее этот алгоритм, тем в большей степени затрудняется криптоанализ шифра.
- **Функция раунда.**

Спроектировать блочный шифр нетрудно. Трудно спроектировать такой блочный шифр, который не только стоек, но может быть легко описан и реализован. Настоящий фокус – разработать алгоритм с возможно наименьшим ключом, требованиями к памяти и скорости работы. (Б. Шнайер)

Т.о., целью построения блочных шифров является не только создание

стойкого алгоритма, но и такого, чтобы его реализация была дешевой, а время работы наименьшим. Поэтому легко реализуемые шифры на базе гомогенных сбалансированных сетей Файстеля применяются гораздо чаще.

Стандарт шифрования данных DES

Самым распространенным и наиболее известным алгоритмом симметричного шифрования является ***DES*** (Data Encryption Standard – Стандарт шифрования данных). Алгоритм был разработан в 1977 году, в 1980 году был принят NIST (National Institute of Standards and Technology США) в качестве стандарта Federal Information Processing Standard 46 (FIPS PUB 46).

DES является классической (сбалансированной), гомогенной *сетью Фейштеля* с двумя ветвями. Данные шифруются 64-битными блоками, используя 56-битный ключ (ключ 64 бита: 56 бит значащие, 8 бит проверки контроля четности).

Основные достоинства алгоритма DES:

- используется один ключ длиной 56 бит (для лобового вскрытия на тот момент – 1000 лет);
- зашифровав сообщение с помощью одного пакета программ, для расшифрования можно использовать любой пакет программ, соответствующий стандарту DES;
- высокая скорость работы;
- достаточно высокая криптостойкость алгоритма;
- регулярность, удобная для аппаратной реализации.

После того как DES был утвержден в качестве федерального стандарта США, постоянно возникали дискуссии по поводу его надежности и того уровня безопасности, который им обеспечивается.

Уязвимость DES была продемонстрирована только спустя 20 лет в 1997 году в ходе объявленного RSA Laboratories конкурса на поиск секретного ключа. Суть конкурса состояла в том, чтобы найти ключ DES, имея в распоряжении шифрованное сообщение для неизвестного открытого сообщения, начинающегося тремя известными блоками текста (24-символьной фразой "the unknown message is:"). Конкурс был объявлен 29 января 1997 года. Решив принять участие в конкурсе, независимый консультант Рок Версер (Rocke Verser) разработал программу перебора вариантов ключа и распространил ее в Internet. В общей сложности в выполнение проекта было вовлечено более 70000 систем, на каждой из которой добровольцы подбирали ключ в выделенных им областях пространства возможных ключей DES. Проект начался 18 февраля 1997 года и успешно закончился спустя 96 дней, когда после перебора примерно четверти всех возможных комбинаций был найден правильный ключ. Этот

конкурс показал возможность решения сложных криптографических задач с помощью распределенных систем персональных компьютеров.

С 1997 года компания RSA DSI проводит конкурсы DES Challenge по криптоанализу DES «грубой силой».

Рекордные достижения:

1) Старт 13.03.97 г., распределенные вычисления, группа пользователей Internet (координатор Роки Версер) – 96 суток.
2) Старт 13.01.98 г., распределенные вычисления, компания Distributed.net – 39 суток.
   Старт 13.07.98 г., специализированный компьютер, компания Electronic Frontier Foundation – 56 часов.
3) Старт 18.01.99 г., спец. компьютер + распределенные вычисления – 22 часа 15 мин..

Т.о. уязвимость DES была продемонстрирована, но он остался базовым строительным блоком для современных симметричных блочных шифров.

В 1998 г. NIST объявил конкурс на принятие нового стандарта со 128-битным ключом (к тому времени DES был вскрыт «в лоб» с использованием Интернет). В 2001 г. был принят AES (Advanced Encryption Standard).

Алгоритм DES.

(Рисунок)

Процесс шифрования состоит из четырех этапов.

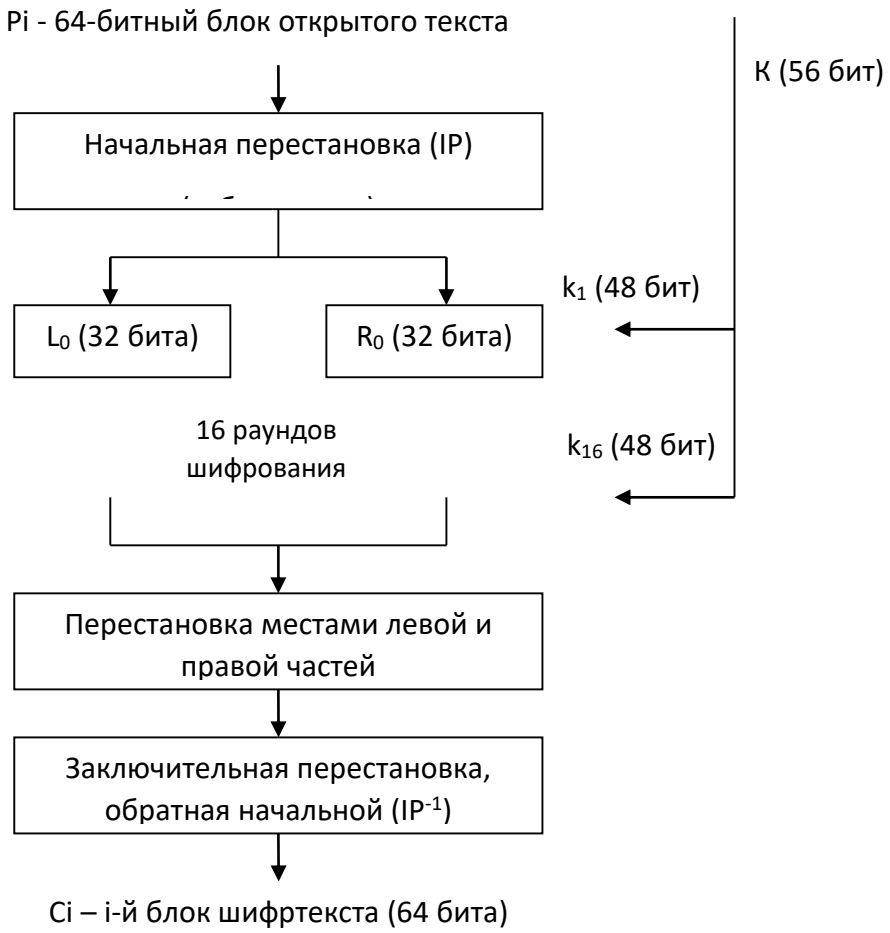
1) Начальная перестановка (IP) 64-битного блока исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей.

2) 16 *раундов* одной и той же функции, которая использует операции сдвига и подстановки.

3) Левая и правая половины выхода последней (16-й) итерации меняются местами.

4) Перестановка IP^{-1} результата, полученного на третьем этапе. Перестановка IP^{-1} инверсна начальной перестановке.

Подключи k_i для каждого *раунда* используются разные, полученные на основе исходного ключа К.



64-битный входной блок проходит через 16 раундов. Рассмотрим последовательность преобразований, используемую в каждом *раунде* DES.

Левая и правая части каждого промежуточного значения трактуются как отдельные 32-битные значения, обозначенные L и R. Каждую итерацию можно описать следующим образом:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, где $\oplus$ - операция XOR.

Выход левой половины L_i равен входу правой половины R_{i-1}.

Выход правой половины R_i является результатом применения операции XOR к L_{i-1} и функции F, зависящей от R_{i-1} и K_i.

Функция F.

1) R_i расширяется до 48 битов, используя таблицу, которая определяет *перестановку плюс расширение* на 16 битов. Цель расширяющей перестановки – увеличить зависимость битов результата от битов исходных данных (лавинный эффект).

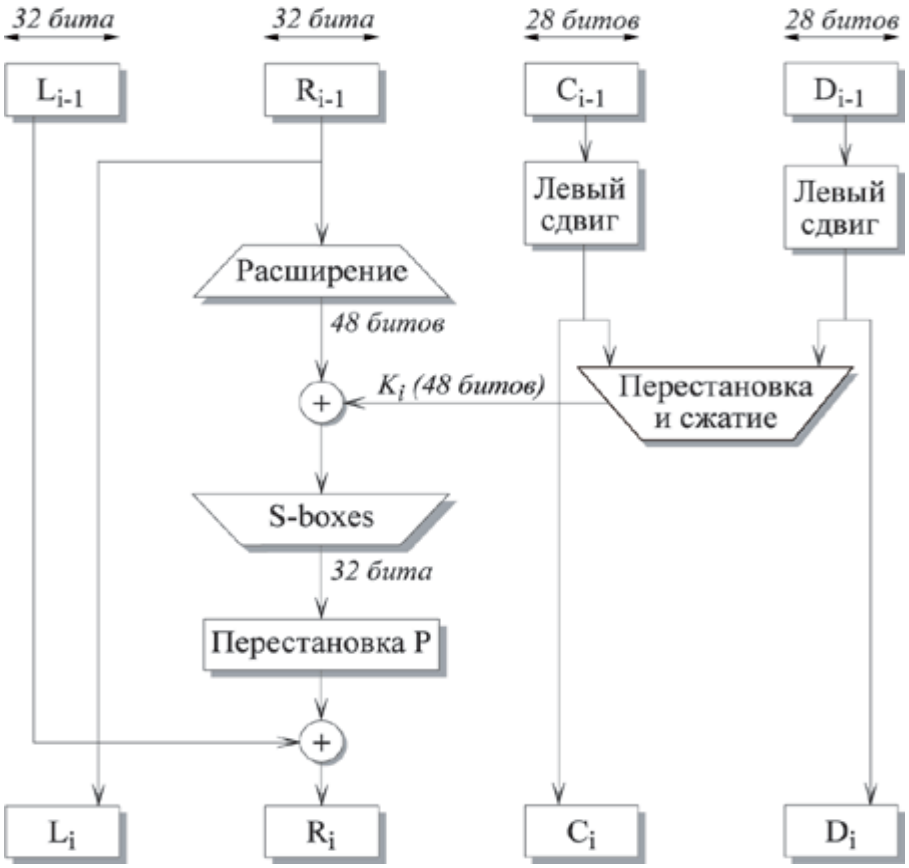
2) Выполняется операция XOR с 48-битным *подключом раунда* K_i.

3) Полученное 48-битное значение проходит через 8 S-блоков, образуя новое 32-битное значение (Это ключевой шаг алгоритма DES, именно S-блоки обеспечивают стойкость DES).

Каждый S-блок (матрица из 4 строк и 16 столбцов, элементы – 4-битовые числа) на входе получает 6 бит, а на выходе создает 4 бита. Эти преобразования определяются специальными таблицами.

4) Перестановка P, целью которой является максимальное переупорядочивание битов, чтобы в следующем *раунде* шифрования с большой вероятностью каждый бит обрабатывался другим *S-box*.

Структура i-го раунда.



Создание подключей

Первоначально 64-битовый ключ сокращается до 56-битового ключа путем отбрасывания каждого 8-го бита. Эти биты используются только для контроля четности, позволяя проверить отсутствие в ключе ошибок.

Для каждого из 16 раундов DES генерируется новый 48-битный подключ k_i.

56-битный ключ разделяется на две половинки по 28 бит, обозначаемые как C_0 и D_0 соответственно. Затем выполняется циклический сдвиг влево на 1 или 2 бита в зависимости от раунда. После этого выбирается 48 бит и изменяется их порядок (Сжимающая перестановка). Получается подключ k_i.

Благодаря сдвигу и сжимающей перестановке в каждом подключе используется отличное от других подмножество битов ключа. Каждый бит используется примерно в 14 из 16 подключей, хотя не все биты используются одинаковое число раз.

Расшифрование выполняется по тому же алгоритму, только ключи подаются в обратном порядке.

Объединение блочных шифров

Один из возможных подходов предполагает использование уже имеющегося ПО и оборудования для многократного шифрования с помощью DES с применением нескольких ключей **(«двойной» и «тройной» DES)**- дешево и сердито.

Известно множество путей объединения блочных алгоритмов для получения новых. Создание подобных схем стимулируется желанием повысить безопасность, избежав трудностей проектирования нового алгоритма.

Для объединения блочных алгоритмов применяется **многократное шифрование**, т.е. использование блочного алгоритма несколько раз с разными ключами для шифрования одного и того же блока открытого текста.

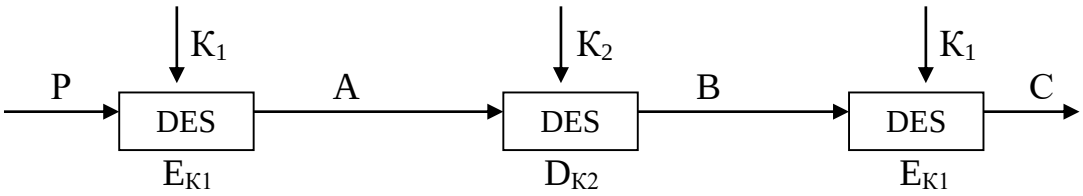
Двукратное шифрование блока открытого текста одним и тем же ключом не приводит к положительному результату. При использовании одного и того же алгоритма такое шифрование не влияет на сложность криптоаналитической атаки полного перебора.

У.Тачмен предложил «Тройной DES с 2-мя ключами».

Блок открытого текста P шифруется 3 раза с помощью 2-х ключей $К_1$ и $К_2$:

$$C = E_{K1}\,(D_{K2}\,(E_{K1}(P)))$$

Т.е. P сначала шифруется с помощью $К_1$, затем расшифровывается $К_2$ и окончательно шифруется $К_1$. Если блочный алгоритм использует n-битовый ключ, то длина ключа данной схемы 2n бит.



Этот режим называется режимом EDE (encrypt-decrypt-encrypt). Расшифрование:

$$P = D_{K1}\,(E_{K2}\,(D_{K1}(C)))$$

Данная схема разработана в корпорации IBM и приводится в стандартах X9.17, ISO8732 в качестве средств улучшения характеристик DES.

При трехкратном шифровании можно применить 3 различных ключа:

$$C = E_{K3}\,(D_{K2}\,(E_{K1}(P))),$$
$$P = D_{K1}\,(E_{K2}\,(D_{K3}(C))).$$

Трехключевой вариант имеет еще большую стойкость. Т.к. длина ключа – 168 бит.

Если требуется повысить безопасность большого парка оборудования, использующего DES, то гораздо дешевле переключиться на схемы трехкратных DES, чем переходить на другой тип криптосистем.

Режимы работы блочных шифров

Для применения DES в различных приложениях NIST были определены четыре режима работы. Они описаны в стандарте FIPS 81. Предполагалось, что этих четырех режимов должно быть достаточно для того, чтобы использовать DES практически в любой области. Эти режимы применимы и для других блочных шифров симметричной схемы.

Режим шифрования - это алгоритм применения блочного шифра, который позволяет преобразовывать открытый текст произвольной длины в шифротекст. Как видно из определения сам блочный шифр теперь является лишь частью другого алгоритма – алгоритма режима шифрования. Это обусловлено тем, что блочный шифр работает только с отдельным *блоком* данных, в то время как алгоритм *режима шифрования* имеет дело уже с целым *сообщением*, которое может состоять из некоторого числа *n* блоков.

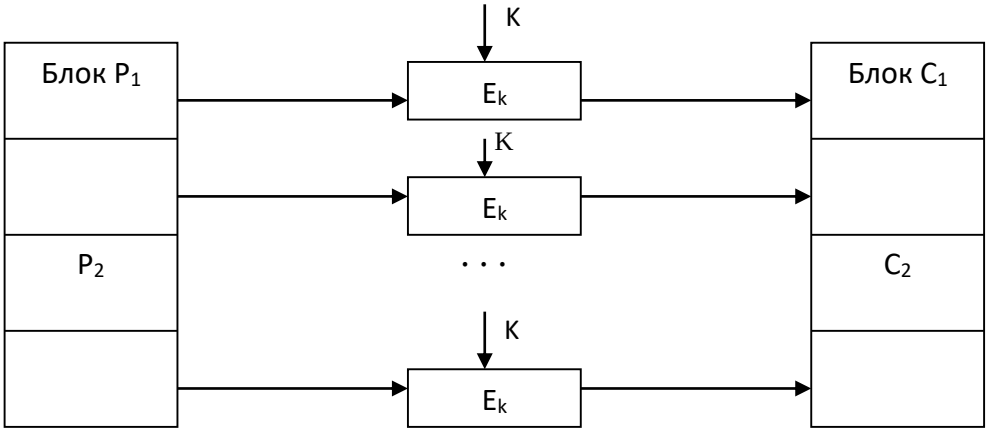
Позже NIST опубликовал еще несколько документов, касающихся режимов шифрования. Основной из них - «Рекомендации для режимов шифрования с блочным шифром». Его рекомендации относятся к любому, одобренному NIST блочному шифру.

Режимы шифрования бывают блочными и поточными. Режимы, в которых функция шифрования применяется ***до*** суммирования с блоком открытого текста называют ***поточными*** (далее будет видно почему), если же функция шифрования вызывается в алгоритме режима ***после*** наложения блока открытого текста, то такой режим шифрования будет ***блочным*.**

1) **Режим Простой Замены** (РПЗ) - Electronic Code Book (ECB) – электронная кодовая книга;
2) **Режим Выработки Имитовставки** (ВИВ) - Cipher Block Chaining (CBC) – сцепление шифрованных блоков;
3) **Режим Гаммирования с обратной связью** (ГОС) - Cipher FeedBack (CFB) –шифрованная обратная связь
4) **Режим Обратной Связи по Выходу** (ОСВ) - Output FeedBack (OFB) – обратная связь по выходу алгоритма шифрования
5) **Режим Гаммирования** (РГ) - CTR или CM (Counter Mode) – шифрование со счётчиком

Режим Простой Замены (РПЗ) или Режим электронной шифровальной книги (ECB)

Это простейший режим шифрования (**ECB- Electronic Codebook**). Открытый текст обрабатывается блоками по ***в*** бит. Каждый блок текста шифруется независимо от других с одним и тем же ключом. Т.о., блоки шифрованного текста не зависят друг от друга. Расшифрование тоже выполняется поблочно на основе одного и того же ключа.



$C_i = E_k(P_i)$

$P_i = D_k(C_i)$

Если длина сообщения превышает ***в*** бит, оно разделяется на ***в*** -битовые блоки с добавлением при необходимости заполнителей к последнему блоку.

Достоинства ECB: 1) простота, 2) параллельное исполнение, 3) возможность непосредственного за- и расшифрования любого блока сообщения по отдельности и независимо от других блоков.

Недостатки ECB: 1) необходимость дополнения последнего блока открытого текста не несущими информацию данными.

2) одинаковые *в*-битовые блоки открытого текста будут преобразовываться в одинаковые блоки шифртекста.

Самой важной особенностью режима ECB является то, что одинаковые *в*-битовые блоки открытого текста, если они встречаются в исходном сообщении, в шифрованном тексте тоже будут представляться одинаковыми блоками. Поэтому при передаче достаточно длинных сообщений режим ECB может не обеспечивать необходимый уровень защиты. Если сообщение имеет явно выраженную структуру, его структура может проявиться в шифртексте (особенно, если элементы повторяются с периодом в *в* бит).

Применение: метод ECB идеален для небольших объемов данных, например для ключей шифрования.

Заполнение последнего блока

Конфиденциальные данные могут иметь произвольный размер, а блочные средства шифрования используют блоки довольно больших размеров (64, 128,

256 и более битов).

Одно из тонких мест в проектировании средств криптографической защиты – вопрос о дополнении данных до размера, кратного размеру блока используемого шифра.

1) Можно дополнять последний блок нулями.

2) Единичным битом и остальными нулевыми.

Однако, криптоаналитик может воспользоваться этой информацией для криптоанализа.

3) Возможный способ решения проблемы – заполнение недостающих битов значением, которое является инверсией значения последнего бита данных.

4) Вычисление свертки или хэша от части или всего объема данных и заполнение недостающих битов в конце блока битами хэша. Это наиболее эффективный вариант с точки зрения стойкости, но наиболее медленный с точки зрения скорости обработки информации.

5) Чаще всего недостающие биты заполняют случайными или псевдослучайными значениями, а последние несколько битов (3 или 4) являются значением реальной длины данных в этом последнем блоке.

6) Для решения проблемы компанией RSA Security было разработано дополнение к стандарту PKCS#5 (Public Key Cryptography Standard # 5 padding). Дополнение заключается в следующем:

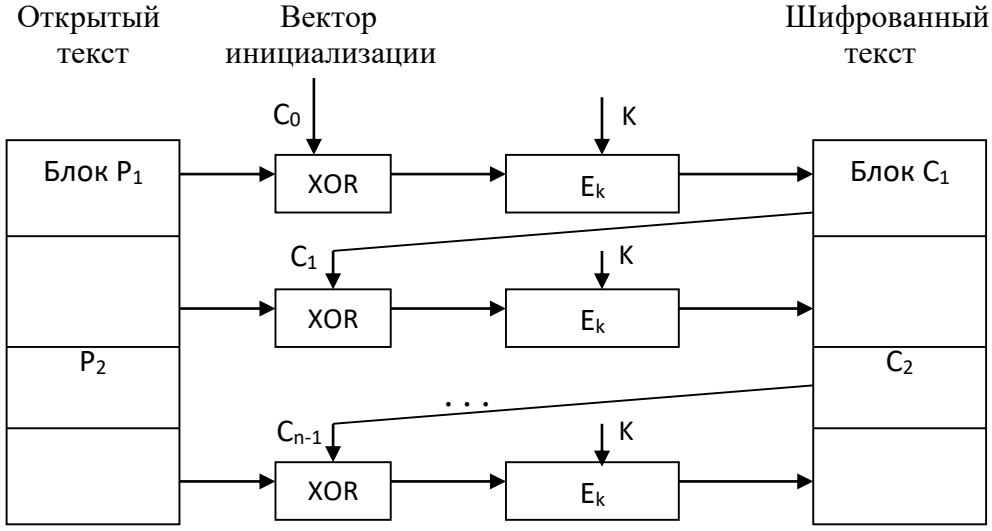
- Если m – число байтов, которыми нужно дополнить блок открытого текста, то значение каждого из дополняющих байтов устанавливается равным m.
- Если не требуется ни одного дополняющего байта, то добавляется 8-байтовый блок (64 бита), в котором значение каждого из дополняющих байтов устанавливаются равным 8.

Число байтов в последнем блоке открытого текста	Дополняющая последовательность байтов
1	7777777
2	666666
3	55555
4	4444
5	333
6	22
7	1
8	88888888

Т.о., последний байт открытого текста, полученного после расшифровки, указывает на то, сколько дополняющих байтов нужно из него удалить.

Режим Выработки Имитовставки (ВИВ) или Режим сцепления шифрованных блоков (CBC-Cipher Block Chaining)

В этом режиме исходный текст также делится на *в*-битовые блоки. Шифрование любого блока выполняется с одним и тем же ключом. Входное значение алгоритма шифрования задается равным операции XOR текущего блока открытого текста и полученного на предыдущем шаге блока шифрованного текста. На первом шаге используется **синхропосылка** (рус.) или **вектор инициализации** – псевдослучайная последовательность.



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

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

$C_0 = IV$

Чтобы получить i-й блок шифрованного текста, необходимо i-й блок открытого текста объединить с помощью XOR с блоком зашифрованного текста, полученным в результате шифрования (i-1)-го блока открытого текста. А затем применить алгоритм шифрования с ключом К.

При расшифровании текст тоже проходит через алгоритм расшифрования поблочно. При этом соответствующий блок открытого текста получается как XOR выходного блока алгоритма расшифрования и предыдущего блока шифрованного текста.

Недостаток CBC: зашифрование сообщения не поддаётся распараллеливанию.

Достоинства CBC:

1) При расшифровании функции $D_k(C_i)$ можно выполнять параллельно и независимо для всех блоков сообщения. Это дает значительный выигрыш во времени.

2) В процессе шифрования CBC все блоки открытого текста оказываются связанными, а входные данные, поступающие на вход функции шифрования, уже не жестко связаны с блоками открытого текста. Поэтому повторяющиеся

в-битовые (по размеру блока) последовательности в шифрованном тексте не проявляются.

3) Последний блок шифротекста, который получается на выходе алгоритма режима CBC зависит как от ключа блочного шифра и вектора инициализации, так и (что важнее в данном случае) от *всех* бит отрытого текста сообщения. А это означает, что этот последний блок шифротекста можно использовать как своего рода идентификатор сообщения, хэш-значение. Более того подделать этот идентификатор без знания ключа шифрования *K* так же трудно, как и правильно угадать сам ключ. Этот идентификатор широко используется для аутентификации сообщений и отправителей и носит название **MAC** (Message Authentication Code), в русскоязычной литературе – **КАС (код аутентификации сообщения** или **имитовставка**).

Применение:

1) Для обеспечения конфиденциальности сообщений, длина которых превышает длину блока шифра.

2) Для аутентификации или получения криптографической контрольной суммы.

Поточные режимы шифрования

Симметричный блочный шифр можно преобразовать и в поточный, используя режим шифрованной обратной связи (CFB), режим обратной связи по выходу (OFB), либо шифрование со счетчиком.

Преимущества при использовании поточного шифра:

1) нет необходимости дополнять сообщение до целого числа блоков (длина всего сообщения должна быть кратна не *b* (длине блока) а *j (*размеру обрабатываемой порции), что означает меньшее количество бит необходимое для выравнивания сообщения);

2) можно работать в режиме реального времени. Каждый символ можно шифровать и сразу же передавать адресату, не дожидаясь окончания шифрования остальной части сообщения;

3) можно подобрать такое значение *j* (размер обрабатываемой порции), что пропускная способность канала будет использоваться наиболее эффективно (либо текст сообщения будет обрабатываться с оптимальной скоростью);

Недостатки: 1) поскольку за одну итерацию алгоритма шифрования преобразуется меньшее число бит, то таких итераций потребуется больше. Отношение *b/j* характеризует количество "холостой" работы блочного шифра (*b* – длина блока в битах).

2) невозможно параллельное выполнение итераций алгоритма. Однако, в режиме расшифрования CFB при условии наличия полного шифротекста все порции открытого текста можно получить одновременно.

Режим Гаммирования с обратной связью (ГОС) или Режим шифрованной обратной связи (CFB-Cipher Feedback)

Предполагается, что единицей передачи данных являются j битов (обычно j = 8). Как и в режиме CBC, происходит сцепление элементов открытого текста, поэтому шифрованный текст, соответствующий любому элементу открытого текста, будет зависеть от всех предыдущих элементов открытого текста.

Шаги алгоритма:

1) Задается начальное значение *b*-битового вектора инициализации.

2) Вектор инициализации шифруется с ключом К и записывается в сдвиговый регистр.

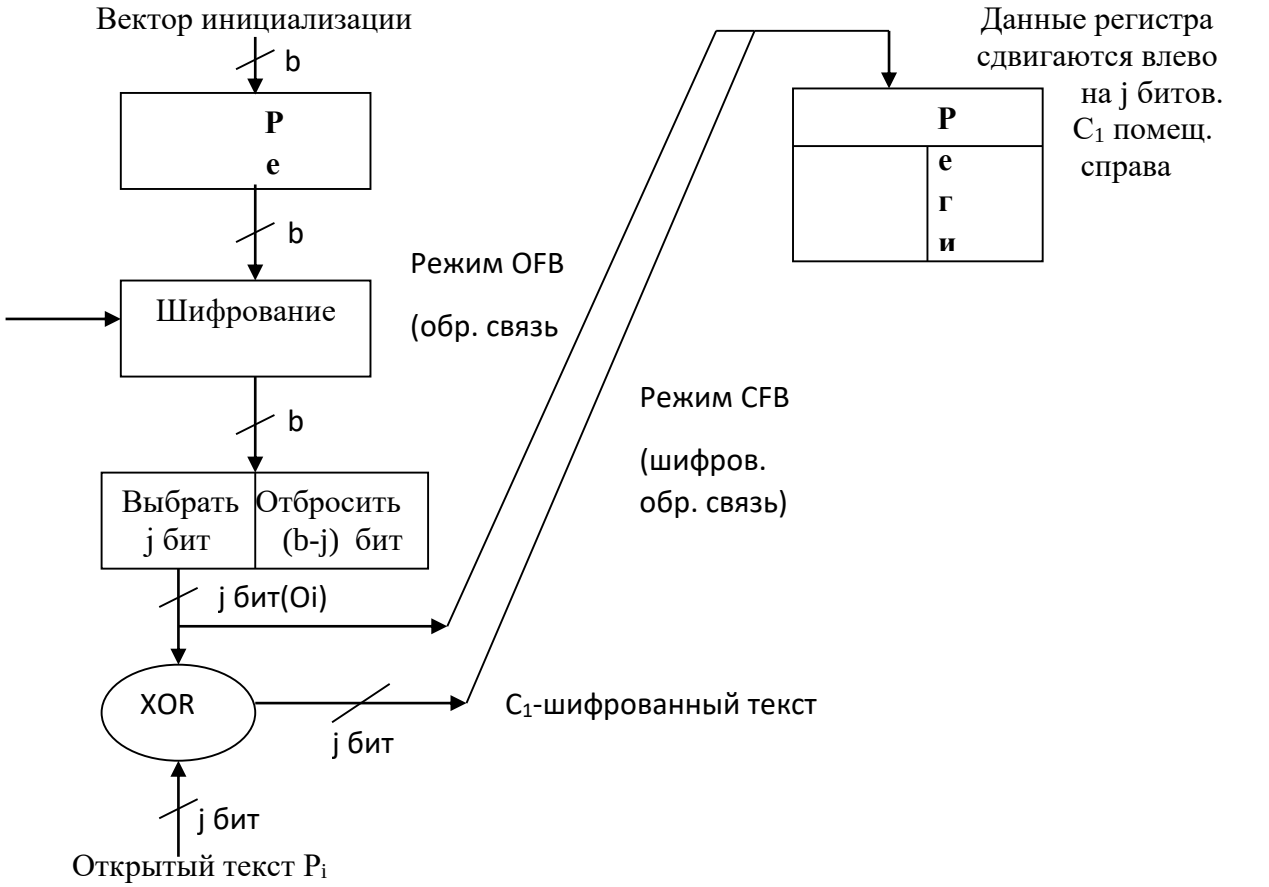
3) Первые j бит открытого текста с помощью операции XOR объединяются с j крайними левыми битами из сдвигового регистра. Получаются j битов зашифрованного текста.

4) Данные в регистре сдвигаются на j битов влево. j битов зашифрованного текста, полученных последними, переносятся в крайние правые разряды регистра.

$\mathbf{C_i = E_k\,(C_{i-1}) \oplus P_i}$

$\mathbf{P_i = E_k\,(C_{i-1}) \oplus C_i}$

$\mathbf{C_0 = IV,}$ где IV - вектор инициализации



Расшифрование CFB:

1. В регистр сдвига справа помещается j бит.
2. Шифрование.
3. j левых бит XOR с C_i, получается P_i.

Применение CFB.

1) Метод CFB используется в поточных приложениях, которые не приспособлены для работы с *в*-битовыми блоками данных (например, telnet работает побайтно - поточно).

2) Для аутентификации или получения криптографической контрольной суммы.

Режим обратной связи по выходу алгоритма шифрования (OFB-Output Feedback)

Режим OFB во многом подобен режиму CFB (шифрованной обратной связи). В режиме OFB в регистр сдвига подается значение, получаемое на выходе функции шифрования, а в режиме CFB в этот регистр подается порция шифрованного текста.

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

O_i – состояние, не зависящее ни от открытого текста, ни от шифртекста.

Такую обратную связь называют внутренней, поскольку механизм обратной связи не зависит ни от потока открытого текста, ни от потока шифртекста.

Расшифрование OFB: 1. В регистр сдвига справа помещается j бит выхода E_k. 2. Аналогично CFB.

Преимущества: 1) нет необходимости дополнять сообщение до целого числа блоков. Для последнего, возможно неполного, блока сообщения используется ровно столько бит выходных данных функции шифрования, сколько бит в этом блоке. Таким образом, в этом режиме, в отличие от предыдущих, длина сообщения остаётся неизменной в процессе шифрования и, главное, при передаче.;

2) Влияние возможных искажений битов при передаче данных не распространяется на последующие порции данных. Это преимущество для потоковых аудио и видео, в спутниковых каналах связи, т.к. здесь ошибки минимально ухудшают расшифрованный поток информации. (Потоковая передача данных по каналам с помехами).

Недостаток: плохая диффузия, режим OFB обеспечивает меньшую, чем CFB, надежность в отношении нарушений модификации потока данных. Например, единичная ошибка в зашифрованном тексте приводит к появлению единичной ошибки в расшифрованном открытом тексте. Нет лавинного

эффекта. Это дает возможность осуществления контролируемых изменений в получаемом адресатом открытом тексте. При передаче текстовых сообщений это существенный недостаток, т.к. в зашифрованный текст могут быть умышленно внесены локальные изменения, которые вызовут локальные изменения в расшифрованном тексте.

Шифрование со счетчиком (CTR или CM- Counter Mode)

Потоковый режим.

На каждой итерации алгоритма на вход функции шифрования подаётся некое случайное значение T_i . Эти входные данные должны быть различны для всех итераций алгоритма, в которых блочный шифр использует один и тот же ключ шифрования. Поэтому генератор таких значений иногда называют счётчиком (что даёт наиболее простой способ генерации уникальных значений T_i). Можно использовать ГПЧ (генератора псевдослучайных чисел), но тогда необходим начальный вектор инициализации для ГПЧ со стороны отправителя и получателя сообщений.

$C_i = P_i \oplus O_i$,

где $O_i = E_k(T_i)$

T_i – значение счетчика (синхропосылка)

$P_i = C_i \oplus O_i$

Таким образом шифртекст в режиме CTR получается суммированием по модулю 2 очередного блока открытого текста с выходными данными функции шифрования. На вход функции шифрования подаётся очередное значение T_j счётчика блоков сообщения. Расшифрование происходит также путём суммирования по модулю 2 очередного блока шифротекста и результата преобразования функцией шифрования очередного значения счётчика T_j.

Преимущества: 1) операции за- и расшифрования в режиме CTR можно производить параллельно и независимо для всех блоков (как в ECB);

2) одинаковые блоки открытого текста не будут преобразованы в одинаковые блоки шифртекста;

3) отсутствует проблема последнего блока.

Недостаток:

Необходимость генерирования счетчиков и их передачи.

Выбор режима шифрования

Симметричные блочные шифры используются для:

- интерактивного шифрования;
- шифрования криптографических ключей при автоматизированном распределении ключей;
- шифрования файлов;
- шифрования почтовых отправлений;
- защиты сообщений электронной системы платежей;

- аутентификации;
- др. практических задач.

Первоначально симметричные блочные шифры применялись только для шифрования данных. Затем их применение было обобщено и на аутентификацию.

Банковские стандарты ANSI определяют для шифрования режимы ECB и CBC, а для аутентификации – режимы CBC и n-битовый CFB.

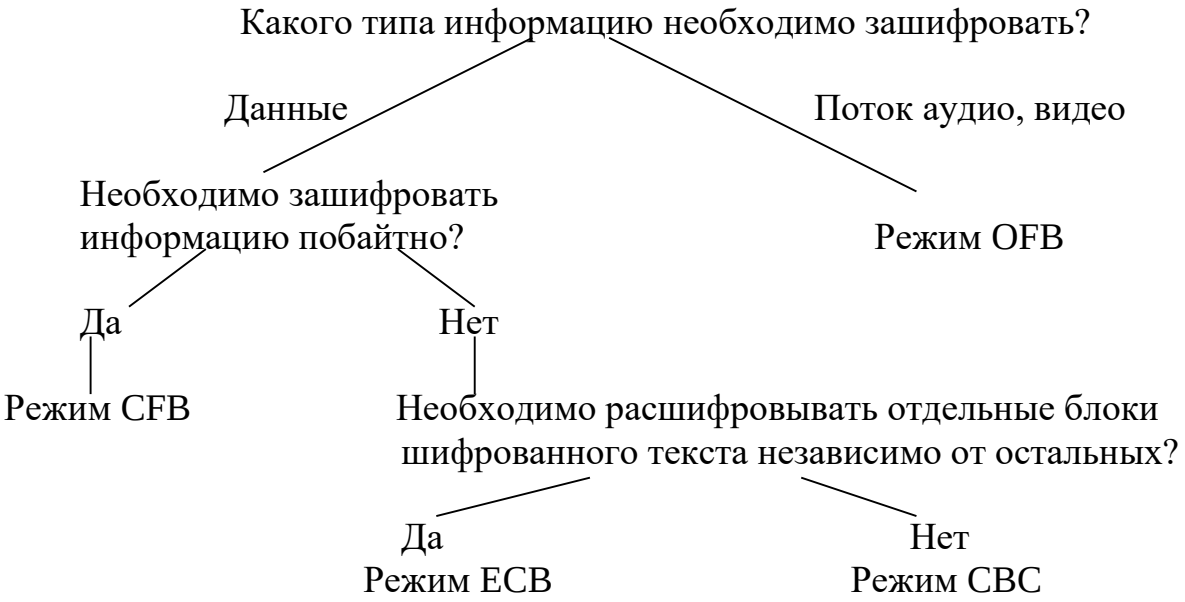
В системах автоматической обработки данных человек не в состоянии просмотреть данные, чтобы установить, не внесены ли в них изменения и ошибки. Поэтому желательно иметь автоматическое средство обнаружения преднамеренной и непреднамеренной модификации данных. Обыкновенные средства обнаружения ошибок непригодны, т.к. если алгоритм преобразования кода известен, противник может выработать правильную контрольную сумму после внесения изменений в данные.

С помощью симметричного алгоритма в режиме CBC можно образовать криптографическую контрольную сумму, которая способна защитить как от случайных, так и от преднамеренных изменений данных.

Этот процесс описывается стандартом аутентификации данных ЭВМ (FIPS 113). Суть стандарта состоит в том, что данные зашифровываются в режиме шифрованной обратной связи (CFB) или в режиме сцепления шифрованных блоков (CBC), в результате чего получается окончательный блок шифра, представляющий собой функцию всех разрядов открытого текста.

После этого сообщение может быть передано по сети с использованием присоединенного к нему последнего блока шифра, который служит криптографической контрольной суммой.

Выбор режима шифрования:



2.5 Криптография с открытым ключом

Принципы построения криптосистем с открытым ключом

Уитфилд Диффи и Мартин Хеллман были первыми, кто публично изложил принципы криптографии с открытым ключом в 1975 году. (Агенство Нац. Безопасности США утверждало, что использовало из еще в 1966 г., но доказательств не представило.)

Модель криптосистемы с открытым ключом.

1. Алгоритмы шифрования и расшифрования являются открытыми.
2. Каждый участник генерирует пару ключей (открытый и личный).
3. Один из ключей публикуется (размещается в открытом каталоге или файле). Поэтому его называют **открытым ключом**. Все участники имеют доступ к открытым ключам друг друга. Открытый ключ может быть отправлен по незащищенным каналам.
4. Второй ключ из пары называется **личным ключом**, т.к. он остается в личном владении. Личный ключ генерируются участником для себя и никогда никуда не передается. До тех пор, пока системе удается сохранять свой личный ключ в секрете, поступающие сообщения оказываются защищенными.
5. С точки зрения вычислений нереально определить ключ расшифрования, зная только используемый криптографический алгоритм и ключ шифрования.
6. Некоторые алгоритмы с открытым ключом (например, RSA) имеют следующее свойство: любой ключ из пары может служить для шифрования, и тогда другой может применяться для расшифрования.
7. В любой момент система может изменить свой личный ключ и опубликовать соответствующий ему открытый ключ, заменяющий старый открытый ключ.

Рассмотрим основные элементы схемы шифрования с открытым ключом более подробно.

Источник - Алиса - создает открытый текст

$X = [x_1, x_2, \ldots, x_M]$, $x_i \in A$ (конечный алфавит).

Сообщение адресовано получателю - Бобу.

Адресат генерирует связанную пару ключей:

- открытый ключ KU_b;
- личный ключ KR_b.

Отправитель формирует шифрованный текст

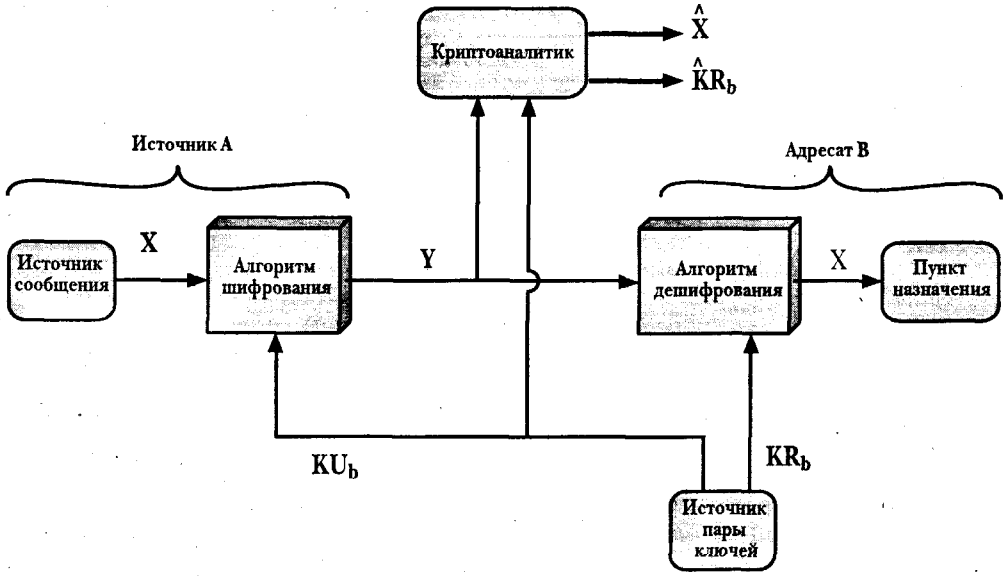
$Y = [y_1, y_2, \ldots, y_N]$ следующим образом:

$$Y = E_{KUb}(X).$$

Получатель выполняет обратное преобразование:

$$X = D_{KRb}(Y).$$

Противник, наблюдая Y и имея доступ к KU_b, но не к KR_b, или X, должен будет пытаться восстановить X и/или KR_b.



Данная схема обеспечивает конфиденциальность. Но КсОК можно использовать также для аутентификации.

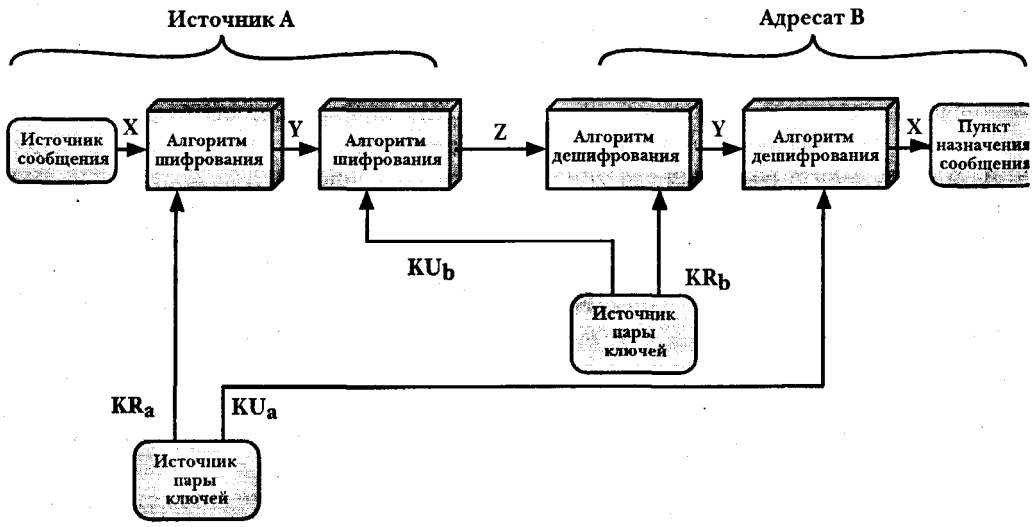
В этом случае отправитель А готовит сообщение адресату В и перед отправлением шифрует это сообщение с помощью личного ключа пользователя А. Получатель В может расшифровать это сообщение, используя открытый ключ А. Т.к. сообщение было зашифровано личным ключом отправителя А, только он и мог это сделать. Поэтому в данном случае все шифрованное сообщение выступает в качестве *цифровой подписи.*

$$Y=E_{KRa}(X),$$
$$X=D_{KUa}(Y).$$

Для обеспечения конфиденциальности и аутентификации необходимо двойное шифрование.

$$Y=E_{KUb}[E_{KRa}(X)],$$
$$X = E_{KUa}[E_{KRb}(Y)].$$

Алгоритм шифрования с открытым ключом, который оказывается весьма сложным, должен при каждой передаче данных применяться четыре раза, а не два.



Применение криптосистем с открытым ключом.

1) **Шифрование/расшифрование.** Отправитель шифрует сообщение с использованием открытого ключа получателя.

2) Цифровая подпись. Отправитель "подписывает" сообщение с помощью своего личного ключа. Подпись получается в результате применения криптографического алгоритма к сообщению или к небольшому блоку данных, являющемуся функцией сообщения.

3) Обмен ключами (Управление ключами). Две стороны взаимодействуют, чтобы обменяться сеансовым ключом. При этом возможно несколько различных подходов, предполагающих применение личных ключей одной или обеих сторон.

Одни алгоритмы шифрования с открытым ключом подходят для всех трех типов применения, тогда как другие предназначены только для одной или двух из этих категорий.

Например: RSA – 1,2,3. DSS – 2. Диффи-Хеллмана – 3.

Недостатки КсОК.

1) Алгоритмы с открытым ключом медленные.

Симметричные алгоритмы примерно в 100 раз (программные реализации), 1000-10000 раз (аппаратные реализации) быстрее. Конечно и компьютеры становятся все быстрее. Но и требования к пропускной способности тоже возрастают, и ключи становятся длиннее.

Поэтому всегда будет необходимость шифровать данные быстрее, чем это может обеспечить криптография с открытым ключом.

2) КсОК уязвимы к атакам на основе подобранного открытого текста.

Эта форма атаки применима только к системам с открытым ключом.

Если известен Y=E(X), где X – открытый текст и существует n возможных открытых текстов, криптоаналитику достаточно зашифровать все n возможных открытых текстов открытым ключом и сравнить результаты с Y.

Он не сможет таким путем восстановить ключ расшифрования, но сумеет

определить X (исходный текст).

Атака на основе подобранного открытого текста особенно эффективна, если число возможных шифрованных сообщений относительно невелико. Например, если X- это денежная сумма (N счета), про которую известно, что она < 1 000 000, такое вскрытие сработает. Криптоаналитик испытает весь миллион значений.

Криптосистема RSA

Со времени открытия криптографии с открытым ключом только один такой алгоритм получил широкое признание – схема Райвеста-Шамира-Адлемана (RSA). Его разработали: Рон Райвест (Rivest), Ади Шамир (Shamir), Лен Адлеман (Adleman) в 1978 г.

RSA – блочный шифр, в котором открытый текст *M*, шифртекст *C*, открытый ключ KU_b и личный ключ KR_b принадлежат множеству целых чисел $Z_N = \{0,1,\ldots,N-1\}$

$$Z_N = \{0,1,\ldots,N-1\}, \text{ где}$$

N – модуль

$N = P \cdot Q$

P и *Q* – случайные большие простые числа. Для обеспечения максимальной безопасности выбирают *P* и *Q* примерно равной длины.

Открытый текст M шифруется блоками длиной k бит: $k \le log_2 N$ (двоичное значение каждого блока <N), $0 \le M_i \le N-1$.

На практике длина блока выбирается из расчёта:

$2^k < N \le 2^{k+1}$

Отправитель должен знать *N* и *e*, т.е.

***KU*={*e*,*N*} – открытый ключ.**

Получатель должен знать *P*,*Q* и *d*:

***KR*={*d*,*P*,*Q*} – личный ключ** (храниться в секрете).

У. Диффи и М. Хеллман сформулировали требования, выполнение которых обеспечивает безопасность асимметричной криптосистемы (в применении к RSA):

1) Должны существовать такие *e*,*d* и *n*, что $M^{e \cdot d} \equiv M \bmod N$ для всех *M*<*N* $(a = k \cdot n + b, a \equiv b \bmod n \Rightarrow b = a \bmod n)$ $M = M^{e \cdot d} \bmod n \Rightarrow M^{e \cdot d} \equiv M \bmod n$.

2) Должны относительно легко вычисляться M^e и C^d для всех *M*<*N*

3) Должно быть практически невозможно определить *d* по имеющимся *e*, *n*.

Из требования (1) с применением следствия теоремы Эйлера $\varphi(n) = \varphi(P \cdot Q) = (p-1)(q-1)$, т.е. *e* и *d* – взаимно обратные по mod *n*, e^{-1} – мультипликативное обратное *d*.

Это возможно, если *d* и *e* являются взаимно простыми с *φ(n)*, т.е.

НОД(φ(n), e) = 1, $1 < e \le \varphi(n)$.

φ(*n*) – функция Эйлера – количество положительных целых, меньших *n*,

которые взаимно просты с n.

Если бы противник смог разложить N на множители P и Q, то он узнал бы «потайной ход» - тройку чисел (P,Q,e), вычислил значение функции Эйлера

$$\varphi(n) = (P-1)(Q-1)$$

и определил значение секретного ключа $d \equiv e^{-1} \bmod \varphi(N)$.

Процедуры шифрования и расшифрования

Пользователь А хочет передать пользователю В сообщение в зашифрованном виде с использованием RSA. Криптосистему RSA должен сформировать получатель сообщения, т.е. пользователь В.

Последовательность действий:

На стороне пользователя В.

1. Пользователь В выбирает 2 произвольных больших простых числа P и Q.
2. Пользователь В вычисляет значение модуля N = P*Q
3. Пользователь В вычисляет функцию Эйлера

$$\varphi(N) = (P-1)(Q-1)$$

(количество положительных целых <N, которые взаимно просты с N)

2. Пользователь В выбирает случайным образом значение открытого ключа e с учётом выполнения условий:

$$НОД(\varphi(n), e) = 1, \quad 1 < e \le \varphi(n),$$

т.е. e и $\varphi(N)$ – взаимнопростые.

3. Пользователь В вычисляет значение секретного ключа d, используя расширенный алгоритм Евклида при решении сравнения

$$d \equiv e^{-1} \bmod \varphi(N)$$

(e и d являются взаимно простыми по модулю $\varphi(N)$)

или

$$ed \equiv 1 \bmod \varphi(N),\ d < \varphi(N)$$

6. Пользователь В пересылает пользователю А открытый ключ - пару чисел KU_b=(N,e) по незащищённому каналу.

На стороне пользователя А.

7. Пользователь А разбивает исходный открытый текст М на блоки, каждый из которых может быть представлен в виде числа

$$M_i = 0,1,2,\dots,N-1 \qquad 0 \le M_i \le N-1$$

8. Пользователь А шифрует текст, представленный в виде последовательности чисел M_i по формуле

$$C_i = M_i^e (mod\ N)$$

и отправляет криптограмму пользователю В

$$C_1, C_2, \dots, C_i, \dots$$

На стороне В:

9. Пользователь В расшифровывает принятую криптограмму, используя секретный ключ по формуле

$$M_i = C_i^d (mod\ N)$$

Пример:

Шифрование сообщения CAB.

Для простоты вычислений используем небольшие числа.

Пользователь B

1. Выбирает P=3 и Q=11
2. Вычисляет модуль $N=P\cdot Q=3\cdot 11=33$
3. Вычисляет значение функции Эйлера

$\varphi(N)=\varphi(33)=(P-1)(Q-1)=2\cdot 10=20$

3. Выбирает в качестве открытого ключа e - произвольное число взаимно простое с $\varphi(N)$ и $1 < e \le \varphi(n)$.

   $1<e\le 20$, $НОД(e,20)=1$

Пусть e=7

4. Вычисляет значение секретного ключа d, используя расширенный алгоритм Евклида при решении сравнения

   $ed\equiv 1 \bmod \varphi(N)$ $(a\equiv b \bmod n)$

   это значит $ed= k*\varphi(N)+1$ $(a=k*n+b)$

Возьмем k=1, подставим e и φ(N), получим

$7*d=1*20+1=21$

d=3

$3<20$ $(d<\varphi(N))$

6. Пересылает пользователю A открытый ключ - пару чисел (N=33, e=7)

Пользователь A

7. Представляет шифруемое сообщение как последовательных чисел в диапазоне 0..32 (n-1=32).

Пусть A-1, B-2, C-3

CAB: $M_1=3$, $M_2=1$, $M_3=2$.

8. Шифрует текст $M_1M_2M_3$, используя e=7 и N=33, по формуле

$C_i=M_i^e (mod\ N)=M_i^7 (mod\ 33)$

Получает $C_1=3^7\ mod\ 33=2187\ mod\ 33=9$

$C_2=1^7\ mod\ 33=1$

$C_3=2^7\ mod\ 33=128\ mod\ 33=29$

Отправляет B криптограмму C_1,C_2,C_3=9,1,29.

Пользователь B

9. Расшифровывает криптограмму, используя d=3 по формуле

$Mi=Ci^3\ mod 33$

$M_1=9^3\ mod\ 33=729\ mod\ 33=3$

$M_2=1^3\ mod\ 33=1$

$M_3=29^3\ mod\ 33=24389\ mod\ 33=2$

Таким образом: 3,1,2=CAB

Вычислительные аспекты и защищённость RSA

Вычислительные аспекты.

I) При вычислении ключей необходимо выбрать P и Q – 2 больших ($10^{75} \div 10^{100}$) простых числа. Чаще всего процедура заключается в выборе случайного нечётного числа приблизительно желаемой величины и выяснения, является ли это число простым. Для проверки того, что числа простые, существует целый ряд тестов.

Почти все такие тесты носят вероятностный характер. Это значит, что тест определит только, что число вероятно простое. Например, тест Миллера-Рабина или Лемана.

Затем подбирается 2-ое простое число из условий:

1) P и Q должны различаться по длине всего на несколько разрядов (например, P и Q должны попадать в диапазон от 10^{75} до 10^{100}.

2) $НОД(P\text{-}1, Q\text{-}1)$ должен быть достаточно малым.

II) После определения P и Q выбирается значение открытого ключа. Процедура заключается в генерировании случайных чисел и сравнении с $\varphi(N)$, пока не будет найдено число, взаимно простое с $\varphi(N)$.

$НОД(e, \varphi(N))=1$.

Шифрование и расшифрование в RSA предполагает использование операции возведения целого числа в целую степень по *mod n.*

Если возведение в степень выполнять непосредственно с целыми числами, а потом проводить сравнение по модулю n, то промежуточные значения окажутся огромными.

Поэтому, используя свойства арифметики в классах вычетов:

$$((a \bmod n)\cdot(b \bmod n)) \bmod n=(a\cdot b) \bmod n$$

можно приводить промежуточные результаты по *mod n.* Это делает вычисления практически возможными.

Вторая проблема – эффективная реализация операции возведения в степень.

В RSA используются очень большие показатели. Один из возможных подходов:

возведение числа в квадрат и повторное возведение в квадрат промежуточных результатов: $x^{16}=x\cdot xxxxxxxxxxxxxxx$.

$$(((x^2)^2)^2)^2=x^{16}$$

Тот же результат при всего четырёх умножениях.

Защищённость алгоритма RSA.

Тремя возможными подходами к криптоанализу алгоритма RSA являются следующие.

1) **Простой перебор.** Предполагает проверку всех возможных личных

ключей.

2) **Математический анализ.** Существует несколько подходов такого рода, но все они по сути эквивалентны нахождению множителей p и q, $p \cdot q = n$.

3) **Анализ временных затрат.** Опирается на анализ времени выполнения алгоритма расшифрования.

2.6 Хеширование и электронная цифровая подпись

Хеш-функции

Хеширование — преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины.

В качестве хеш-функции используются математические или иные функции, которые принимают на входе сообщение произвольной длины (его называют **прообраз**), и преобразуют его в значение фиксированной длины (обычно меньшей), называемое **значением хеш-функции, сверткой, хеш-кодом, профилем сообщения, дайджестом, функцией сжатия.**

$$h = H(M),$$

где M – прообраз,

H(M) – значение хеш-функции.

Целью использования функции хеширования является получение «дактилоскопической» характеристики файла или любого блока данных.

Хеш-функции используются без ключа. Алгоритм вычисления хеш-функции является открытым. Проверить хеш-значение может кто-угодно.

Существует множество алгоритмов хеширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкость и т.п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи.

Применение функций хеширования:

1. Сравнение данных: если у двух массивов хеш-функции разные, массивы гарантированно различаются; если одинаковые — массивы, скорее всего, одинаковы.
2. Контроль ошибок: изменение любого числа битов сообщения выливается в изменение хэш-кода. Значение хэш-кода присоединяется к сообщению отправителем. Получатель устанавливает аутентичность путем повторного вычисления значения функции хеширования. Пример - контрольная сумма или CRC.
3. Хеширование паролей.
4. Генерация ключей шифрования фиксированной длины на основе строки, введенной пользователем.

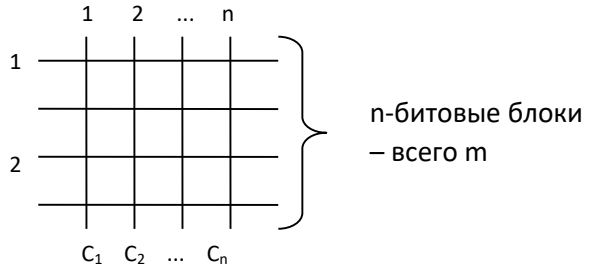
Пример простой функции хеширования.

Все функции хеширования построены на том, что вводимое значение рассматривается как последовательность n-битовых блоков. Одной из

простейших функций хеширования является связывание всех блоков операций поразрядного исключающего «Или»(XOR)

$$c_i = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{im} \ ,$$

где c_i – i-й бит хэш-кода, 1<=i<=n,
m – число n-битовых блоков в сообщении,
b_{ij} – i-й бит в j-м блоке



Усовершенствовать эту функцию можно с помощью выполнения однобитового циклического сдвига:

1. Начальная инициализация n-битового значения функции хеширования нулевым значением.
2. Последовательная обработка n-битовых блоков данных по следующему правилу:
   – Выполнение циклического сдвига текущего значения функции хеширования влево на 1 бит.
   – Добавление текущего блока к значению функции хеширования с помощью операции XOR.

В общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше чем вариантов входного массива. Поэтому возможно возникновение коллизий.

Коллизии – массивы данных, имеющие одинаковые значения хеш-кодов.

Вероятность возникновения коллизий играет важную роль в оценке качества хеш-функций.

Хеш-функции – один из основных элементов многих криптографических протоколов. В криптографии применяются только криптографическистойкие хеш-функции.

Для возможности ее применения в криптографии, функция хеширования H должна иметь следующие свойства:

1. Значение H(x) должно вычисляться относительно легко для любого заданного x, а алгоритм вычисления должен быть практичным с точки зрения аппаратной и программной реализации.
2. Изменение в прообразе даже одного бита должно приводить к изменению приблизительно половины битов в значении хеш-функции (диффузия или лавинный эффект).

3. Для любого данного кода h должно быть практически невозможно вычислить x, для которого H(x)=h. Это свойство однонаправленности или необратимости.

Т.о. в криптографии используют однонаправленные хеш-функции: вычислительно невозможно найти прообраз, соответствующий данному значению хеш-функции.

4. Для любого данного блока x должно быть практически невозможным вычислить y≠x, для которого H(x)=H(y). Такое свойство называется слабой сопротивляемостью коллизиям или стойкость к коллизиям первого рода.

Это свойство гарантирует, что не удастся найти другое сообщение, дающее в результате хеширования то же самое значение. Что предотвращает возможность фальсификации сообщения.

5. Должно быть практически невозможно подобрать пару сообщений (M, M'), имеющих одинаковый хеш-код. Это свойство называется сильной сопротивляемости коллизиям или стойкость к коллизиям второго рода.

Должно быть очень трудно сгенерировать два прообраза, имеющих одинаковое значение хеш-функции.

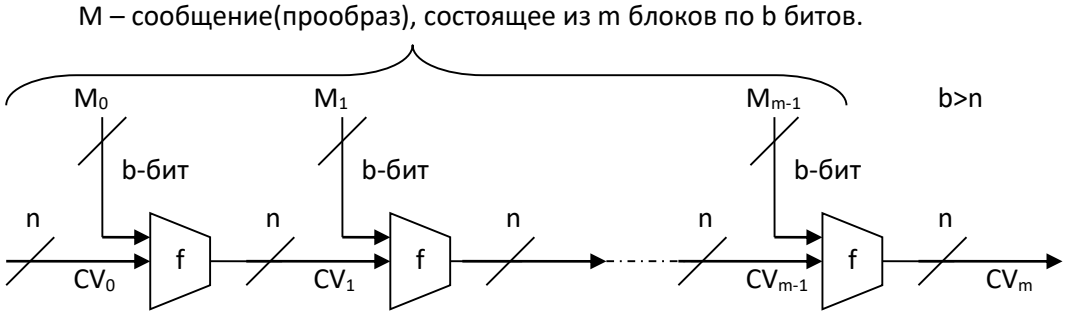
Это свойство определяет стойкость функции хеширования к классу атак, известных под названием атаки, построенные на парадоксе задачи о днях рождения.

Согласно парадоксу о днях рождения, нахождение коллизии для хеш-функции с длиной значений n бит требует в среднем перебора около $2^{n/2}$ операций. Поэтому n-битная хеш-функция считается криптостойкой, если вычислительная сложность нахождения коллизий для неё близка к $2^{n/2}$.

Обобщенная структура функции хеширования

Рассмотрим структуру типичной функции хеширования. Эту структуру предложил Меркл и называется она **итерированной функцией хеширования**.

Именно такую структуру имеет сегодня большинство функций хеширования, используемых в криптографии.



CV_0 - начальное значение переменной сцепления (n бит)

CV_i - переменная сцепления

M_i – i-й вводимый блок сообщения M

f – функция сжатия

m – число вводимых блоков

Функция хеширования получает на вход сообщение M и делит его на m блоков равной фиксированной длины по b битов каждый. Если необходимо, то последний блок дополняется до b битов.

Алгоритм хеширования предполагает многократное применение функции сжатия f, получающей на вход два значения:

1. n –битовое значение, полученное на предыдущем этапе и называемое **переменной сцепления CV_i,**
2. b – битовый блок сообщения M_i

На выходе функция f порождает n –битовое выходное значение.

В начале хеширования переменная сцепления получает начальное значение CV_0, являющееся частью алгоритма. Конечное значение переменной сцепления и будет значением функции хеширования.

Обычно b (блина блока) > n (длина хэш-кода), поэтому и говорят о сжатии. Таким образом, функция хеширования может быть формально описана следующим образом:

$CV_0 = IV$ –начальное n –битовое значение,

$CV_i = f\ (CV_{i-1}, M_{i-1})$, $1 <= i <= m$,

$H(M) = CV_m$, где вводимыми данными функции хеширования является сообщение M, складывающееся из блоков $M_0, M_1, \ldots, M_{m-1}$.

Если функция сжатия f обладает сопротивляемостью коллизиям, то такой же будет и итерированная функция хеширования. Таким образом, проблема создания защищенной функция хеширования сводится к проблеме поиска обладающей сопротивляемостью коллизиям функции сжатия.

Алгоритмы хеширования

Структура наиболее важных из современных функций хеширования соответствует базовой структуре, изложенной выше.

На практике эта структура оказалась надежной, поэтому новые разработки заключаются в усовершенствовании базовой структуры и увеличении длины хэш-кода.

Современные алгоритмы хеширования:, MD, SHA, RIPEMD, HMAC.

Алгоритм MD4 (Message Digest) – лежит в основе всех современных хеш-алгоритмов, является предшественником MD5, MD6 и SHA. Разработан Рональдом Райвестом из Массачусетского Технологического Института в 1990г.. Последняя публикация – RFC-1320.

Общие свойства (требования):

1. Защищенность. (Вычислительно невозможно найти два сообщения с одинаковым хеш-значением. Никакой метод взлома не должен быть эффективнее лобовой атаки).
2. Скорость. (Подходит для программной реализации).
3. Простота и компактность.
4. Предпочтение архитектуры с прямым порядком следования байтов (процессоры Intel 80-x, Pentium – хранение наименее значимых байтов слов в младшей адресной позиции).

Алгоритм MD5 вычисления профиля сообщения разработан Роном Райвестом в 1991г.(опубликован в 1992г.) Описан в RFC 1321. Является улучшенной в плане безопасности версией MD4.

Алгоритм использует в качестве входных данных сообщение произвольной длины и выдает 128-битовое значение профиля сообщения. Данные обрабатываются 512-битовыми блоками.

MD5 один из наиболее распространенных защищенных алгоритмов хеширования.

В настоящее время MD5 считается уязвимым с точки зрения криптоанализа.

Уже В 1996 году было объявлено о коллизии в алгоритме (получение одинакового значения функции для разных сообщений) и предложено использовать другие алгоритмы хеширования, такие как Whirlpool, SHA-1 или RIPEMD-160.

В 2004 году был запущен проект MD5CRK, который обнаружил уязвимости алгоритма, используя birthday атаки.

MD5 позволяет получать относительно надёжный идентификатор для блока данных. Такое свойство алгоритма широко применяется в разных областях. Оно позволяет искать дублирующиеся файлы на компьютере, сравнивая MD5 файлов, а не их содержимое.

С помощью MD5 проверяют целостность скачанных файлов — так, некоторые программы идут вместе со значением хеша. Например, диски для инсталляции.

MD5 используется для хеширования паролей.

Хеш-функции также широко используются для генерации ключей фиксированной длины для алгоритмов шифрования на основе заданной ключевой строки.

Алгоритм MD6 — алгоритм хеширования, разработанный в 2008г. Предлагается на смену менее совершенному MD5. По заявлению авторов, алгоритм устойчив к дифференциальному криптоанализу.

Представлен на конференции Crypto 2008 в качестве кандидата на стандарт SHA-3.

Размер хеша переменный от 0 до 512 бит, $0<d\leq 512$.

Количество раундов переменное и составляет: $r = 40 + (d / 4)$, что нетрадиционно велико.

Помимо традиционного бесключевого режима, применяется хеширование с ключом 512-бит. Основывается на использовании ряда простейших операций: XOR, сложения и сдвига.

В алгоритме хеш-функции использованы весьма оригинальные идеи. За один проход обрабатывается 512 байт, затрудняя проведение ряда атак и облегчая распараллеливание, для чего также применяются древовидные структуры.

Американский стандарт функции хеширования (SHS)

Стандарт функции хеширования США - **SHS (Secure Hash Standard).** – В основе стандарта - семейство алгоритмов вычисления значения хеш-функции **SHA (Secure Hash Algorithm) - з**ащищенный алгоритм хеширования**.**

Рекомендован в качестве основного для государственных учреждений в США. Используется во многих криптографических приложениях и протоколах:

- S/MIME — дайджесты сообщений.
- SSL — дайджесты сообщений.
- IPSec — для алгоритма проверки целостности в соединении «точка-точка».
- SSH — для проверки целостности переданных данных.
- PGP — для создания электронной цифровой подписи.

Алгоритм SHA (сейчас известный как **SHA-0**) разработан в 1993 году NSA (Агенство национальной безопасности) совместно с NIST (Национальный институт стандартов и технологий). Опубликован в виде федерального стандарта обработки информации PUB FIPS 180. Структура SHA близка к MD4.

Новая версия **SHA-1**опубликована в документе FIPS PUB 180-1 в 1995г. Описан в RFC 3174.

Размер хеша – 160 бит.

Размер блока – 512 бит.

SHA-1 значительно более защищенный от атак с перебором всех вариантов, чем MD5. (Профиль SHA-1 на 32 бита длинее, чем MD5, следовательно, сложность задачи нахождения сообщения, имеющего заданный профиль, имеет порядок: 2^{128} операций для MD5, 2^{160} для SHA-1).

Семейство алгоритмов SHA-2 включает **SHA-224, SHA-256, SHA-384, SHA-512.** Введены в действие в 2002 г. Последняя версия - FIPS PUB 180-4.

Размер хеша соответственно 224, 256, 384, 512 бит.

Число раундов 64 или 80.

Размер блока 512 и 1024 бит.

В 2006 г. появился стандарт **RFC 4634** «Безопасные хеш-алгоритмы США (*SHA* и *HMAC-SHA*)», описывающий *SHA-1* и семейство *SHA-2*.

Хеш-функции семейства *SHA-2* построены на основе структуры Меркла — Дамгарда

SHA-3 (*Keccak* — произносится как «кечак») — алгоритм хеширования переменной разрядности, разработанный группой авторов во главе с Йоаном Дайменом, соавтором Rijndael в 2008г.

2 октября 2012 года Keccak стал победителем конкурса криптографических алгоритмов, проводимым Национальным институтом стандартов и технологий США.

В 2015 году алгоритм утверждён и опубликован в качестве стандарта FIPS 202.

Размер хеша: 224, 256, 384, 512 (переменный)

Число раундов: 24

Алгоритм HMAC.

В последние годы возрос интерес к методу вычислений MAC (кода аутентичности сообщения) на основе хэш-кодов. (Ранее использовались симметричные блочные цифры).

Это объясняется следующим:

1. Криптографические функции хеширования программно выполняются быстрее, чем симметричные алгоритмы.
2. Нет ограничений на экспорт из США и других стран.

Был предложен ряд вариантов добавления секретного ключа в уже существующий алгоритм хеширования.

Наибольшую популярность получил подход, называемый HMAC. Описан в документации RFC 2104. Принят как обязательный для реализации в протоколе безопасности IPSec и используется в других протоколах Internet, например SSL (Secure Socket Layer – защищенный обмен сообщениями протоколов прикладного уровня TCP/IP).

Преимущества:

1. Возможность использования существующих функций хеширования, для которых имеются широко доступные программные реализации.

2. Возможность легкой замены функции хеширования на более быструю и защищенную.

3. Высокая скорость работы.

4. Возможность применения ключей и простота обращения к ним.

Таким образом, в алгоритме HMAC функция хеширования интерпретируется как «черный ящик». Это имеет ряд преимуществ схемы HMAC перед другими схемами.

Российский стандарт функции хеширования

ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция хеширования» — действующий российский криптографический стандарт, определяющий алгоритм и процедуру вычисления хеш-функции. Название хеш-функции — «Стрибог», по имени славянского божества, — часто используется вместо официального названия стандарта, хотя в его тексте явно не упоминается.

Разработан Центром защиты информации и специальной связи ФСБ России с участием ОАО «ИнфоТеКС» и введенный в действие 1 января 2013 года.

Размер хеша — 256 или 512 бит; размер блока входных данных — 512 бит.

Стандарт определяет алгоритм и процедуру вычисления хеш-функции для последовательности символов. Этот стандарт разработан и введён в качестве замены устаревшему стандарту ГОСТ Р 34.11-94:

Требования к хеш-функции[1]:

- у хеш-функции не должно быть свойств, которые позволяли бы применить известные атаки;
- в хеш-функции должны использоваться изученные конструкции и преобразования;
- вычисление хеш-функции должно быть эффективным, занимать мало времени;
- не должно быть лишних преобразований, усложняющих конструкцию хеш-функции. Причем каждое используемое в хеш-функции преобразование должно отвечать за определенные криптографические свойства.

В основу хеш-функции положена итерационная конструкция Меркла-Дамгарда с использованием MD-усиления. Под MD-усилением понимается дополнение неполного блока при вычислении хеш-функции до полного путём добавления вектора (0 … 01) такой длины, чтобы получился полный блок. Из дополнительных элементов нужно отметить следующие:

- завершающее преобразование, которое заключается в том, что функция сжатия применяется к контрольной сумме всех блоков сообщения по модулю 2^{512};
- при вычислении хеш-кода на каждой итерации применяются разные функции сжатия. Можно сказать, что функция сжатия зависит от номера итерации.

Описанные выше решения позволяют противостоять многим известным атакам.

Кратко описание хеш-функции ГОСТ Р 34.11-2012 можно представить следующим образом. На вход хеш-функции подается сообщение произвольного размера. Далее сообщение разбивается на блоки по 512 бит, если размер сообщения не кратен 512, то оно дополняется необходимым

количеством бит. Потом итерационно используется функции сжатия, в результате действия которой обновляется внутреннее состояние хеш-функции. Также вычисляется контрольная сумма блоков и число обработанных бит. Когда обработаны все блоки исходного сообщения, производятся еще два вычисления, которые завершают вычисление хеш-функции:

- обработка функцией сжатия блока с общей длиной сообщения.
- обработка функцией сжатия блока с контрольной суммой.

Функция хэширования СТБ 34.101.77-2016

Государственный стандарт РБ СТБ 34.101.77-2016 «Информационные технологии и безопасность. Алгоритмы хеширования.»

Стандарт определяет семейство криптографических алгоритмов хеширования, предназначенных для контроля целостности и необратимого сжатия данных.

Алгоритм хеширования по сообщению произвольной длины строит хеш-значение — слово фиксированной длины. Стороны могут организовать контроль целостности сообщений путем сравнения их хеш-значений с достоверными контрольными хеш-значениями.

Изменение сообщения с высокой вероятностью приводит к изменению соответствующего хеш-значения, и поэтому хеш-значения могут использоваться вместо самих сообщений, например в системах ЭЦП.

Алгоритмы хеширования могут дополнительно использоваться при построении систем имитозащиты, генераторов случайных и псевдослучайных чисел, протоколов аутентификации, доказательств вычислительной работы и др.

Шаговая функция

Алгоритмы хеширования построены по схеме sponge (губка), описанной в [1]. Ядром схемы является шаговая функция, которая определяет сложное биективное преобразование слов большой длины.

В настоящем стандарте шаговая функция действует на слова длины 1536. Действие задается алгоритмом bash-f, определенным в 6.2.

Шаговая функция bash-f имеет самостоятельное значение и может использоваться за пределами настоящего стандарта для построения других криптографических алгоритмов.

Уровень стойкости

Алгоритмы хеширования настоящего стандарта отличаются уровнем стойкости l. Это натуральное число, кратное 16 и не превосходящее 256. ***Алгоритм уровня l вычисляет хеш-значения длины $2l$***, обрабатывая входные слова блоками длины $1536 - 4l$. Уровни $l = 128$, $l = 192$ и $l = 256$ являются стандартными, им следует отдавать предпочтение.

При выборе l следует учитывать, что для определения сообщения с заданным хеш-значением требуется выполнить порядка 2^{2l} операций, а для определения двух различных сообщений с одинаковыми хеш-значениями требуется выполнить порядка 2^{l} операций.

Следует учитывать также, что с ростом l, кроме повышения стойкости, снижается быстродействие алгоритмов. В частности, хеширование на уровне l = 256 выполняется примерно в 2 раза медленнее, чем на уровне l = 128.

Хеш-значение

Длина хэш-значения регулируется уровнем стойкости l. Если при фиксированном l требуются не все, а $n < 2l$ символов хеш-значения, то должны использоваться первые n символов.

Защита функций хеширования и кодов аутентификации.

Атаки на функции хеширования и коды аутентичности можно разделить на две категории:

1. Перебор всех вариантов.
2. Криптоанализ.

I.Атаки с перебором всех вариантов.

1. Функции хеширования.

Способность функции хеширования противостоять атакам с перебором всех вариантов зависит исключительно от длины хэш-кода.

В настоящее время два самых популярных алгоритма вычисления хэш-кодов SHA-1 и RIPEMD-160 обеспечивают хеш-код 160-битовой длины.

2. Коды аутентичности сообщений.

Свойство алгоритма вычисления MAC, которое требуется для защищенности этого алгоритма:

Вычислительная стойкость – при наличии одной или большего числа пар «текст - MAC», (Xi,Ck(Xi)), практически невозможно вычислить любую новую пару (X,Ck(X)) для любого нового значения X=Xi. Другими словами, когда противник хочет получить правильный код аутентичности MAC для некоторого сообщения X, он может использовать две линии атаки:

а) атаку пространства ключей;

б) атаку значения MAC.

Если противник определит ключ, он сможет генерировать MAC для любого X. Противник может также попытаться выяснить MAC, не пытаясь найти ключ. Целью в таком случае является либо поиск правильного MAC для данного сообщения, либо поиск сообщения, соответствующего данному MAC.

II.Криптоанализ.

При этом пытаются обнаружить свойства алгоритма, позволяющие уменьшить объем вычислений в сравнении со скоростью вычислений при полном переборе вариантов.

Идеальная функция хеширования или алгоритм вычисления MAC должны требовать для криптоанализа усилий больше, чем при переборе всех вариантов.

Атаки, основанные на парадоксе задачи о днях рождения.

1. Отправитель А готовится «подписать» сообщение с помощью присоединения к сообщению m-битового значения MAC и шифрование этого значения MAC с личным ключом А.

2. Противник генерирует $2^{m/2}$ вариантов сообщения. Кроме того, противник готовит и столько же сообщений, являющихся вариантами ложного сообщения, которым должно будет заменено истинное.

3. Эти два набора сообщений сравниваются, чтобы найти пару сообщений, порождающих один и тот же хеш-код. Вероятность успеха в соответствии с парадоксом задачи о днях рождения оказывается > 0,5. Если совпадение не найдено, генерируются новые истинные и ложные сообщения, пока не обнаружит совпадение.

4. Противник предлагает вариант истинного сообщения на подпись пользователю А. Эта подпись затем может быть добавлена к ложному сообщению для передачи получателю. Так как оба варианта имеют один хэш-код, они породят одинаковые подписи и противник будет уверен в успехе, не зная ключа шифрования.

Парадокс задачи о днях рождения.

Проблема:

Чему равно минимальное значение k, при котором вероятность того, что по крайней мере у двоих из группы k человек дни рождения совпадают, оказывается равной 0,5?

Оказывается k=23 (P=0,5077). При k=100 P=0,999.

Методы аутентификации сообщений. Коды аутентификации

Аутентификация сообщения - это процедура проверки того, что полученное сообщение пришло от указанного источника и не было изменено в пути следования.

В аутентификации сообщений (как в аутентификации в целом) могут быть выделены два основных уровня:

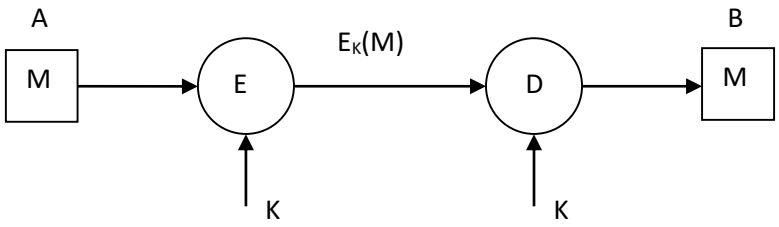
1. создание аутентификатора - на низшем уровне должна выполняться некоторая функция, порождающая аутентификатор (удостоверение, используемое для подтверждения подлинности субъекта).
2. использование аутентификатора в протоколе аутентификации высшего уровня, дающем получателю сообщения возможность проверить достоверность сообщения.

Способы создания аутентификатора:

1. Шифрование сообщения. В качестве аутентификатора используется шифрованный текст всего сообщения.
2. Вычисление кода аутентификации сообщения.
3. Вычисление хеш-кода сообщения.

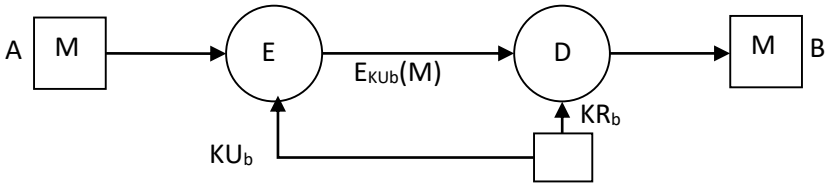
Шифрование сообщения.

Традиционное шифрование (конфиденциальность, аутентификация).

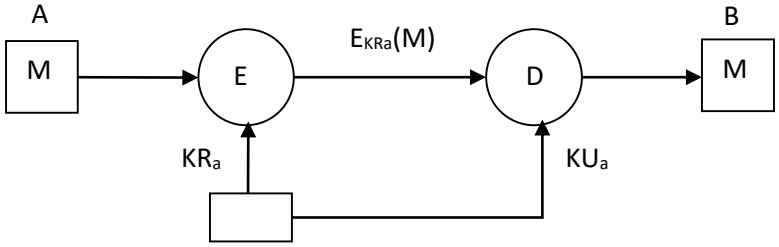


Шифрование с открытым ключом.

А) Простое шифрование с открытым ключом обеспечивает конфиденциальность, но не аутентификацию, так как противник тоже может использовать открытый ключ KUb, чтобы зашифровать сообщение и отправить его от имени А.



Б) Аутентификация и цифровая подпись.



Код аутентификации - небольшой блок данных фиксированного размера, созданный с помощью открытой функции сообщения и секретного ключа.

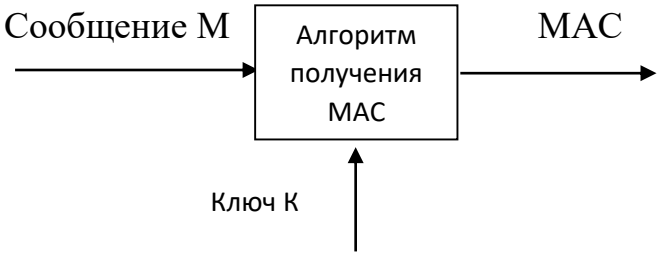
Этот блок данных называется еще:

- криптографическая контрольная сумма;

- имитовставка (русский термин);

MAC (Message Authentication Code) или **DAC** (Date Authentication Code).

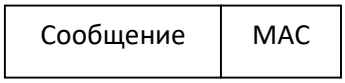
При этом предполагается, что две, участвующие в обмене данными стороны А и В, используют общий секретный ключ К.



Чтобы послать сообщение М адресату В, отправитель А вычисляет код аутентификации сообщения как функцию сообщения и ключа:

$$MAC = C_K(M)$$

Сообщение, с добавленным к нему значением MAC пересылается адресату. Получатель выполняет аналогичные вычисления и получает новое MAC, затем сравнивает его с полученным.



Если ключ известен только получателю и отправителю и значения MAC совпадают, то можно утверждать следующее:

1. Получатель может быть уверен, что сообщение не было изменено.
2. Получатель может быть уверен, что сообщение пришло от указанного отправителя (так как никому больше не известен К).

Разница между получением MAC и шифрованием:

1) размер;

2) для алгоритма вычисления MAC не требуется свойство обратимости.

Код аутентичности сообщения не обеспечивает цифровую подпись, так как отправитель и получатель используют один ключ.

Для получения кода аутентификации могут быть использованы:

1) современные симметричные алгоритмы в режимах CBC и CFB;
2) алгоритмы хэширования с добавлением секретного ключа (алгоритм HMAC).

Один из наиболее известных алгоритмов вычисления MAC – **алгоритм аутентификации данных (DAA** – Data Authentication Algorithm). Он описан в стандарте ANSI (X9.17). Это алгоритм шифрования DES в режиме сцепления шифрованных блоков (CBC) с нулевым вектором инициализации.

Понятие, свойства и виды ЭЦП

Аутентификация сообщений защищает две обменивающиеся сообщениями стороны от любой третьей, но не обеспечивает защиту каждой из сторон от другой. Здесь имеется целый ряд возможностей (атак) для возникновения конфликтов (отрицание авторства сообщения, фальсификация сообщения).

Поэтому в ситуациях, когда нет полного доверия между отправителем и получателем, требуется нечто большее, чем аутентификация. Наиболее привлекательное решение проблемы – использование цифровой подписи.

Цифровая подпись обеспечивает целый комплекс защиты, который было бы сложно осуществить любым другим способом.

Электронная цифровая подпись (ЭЦП) - характеристика электронного документа (или программы), полученная в результате криптографического преобразования документа и позволяющая проверить сохранение целостности документа с момента формирования подписи, авторство и подтвердить факт подписания электронного документа (неотказуемость).

Для ПО – доказательство **легальности** приобретения.

Цифровая подпись аналогична подписи сделанной от руки.

Исходя из определения, ЭЦП должна обладать следующими **свойствами**:

1. Достоверность (целостность). (ЭЦП должна давать возможность установить автора, дату и время подписи, а также отсутствие искажения информации в электронном документе с момента формирования подписи.)
2. Неподдельность (авторство). (Никто не может выдать себя за автора. ЭЦП доказывает, что именно подписавший и никто другой сознательно подписал документ.)
3. Невозможность использовать подпись повторно (т.к. она часть документа, 2 сообщения – 2 разных подписи).
4. Невозможность изменить подписанный документ. (Изменение документа – изменение подписи.)
5. Невозможность отрицания авторства (неотказуемость). (Человек не сможет утверждать, что документ подписан не им.)

Из свойств следуют **требования к цифровой подписи:**

1. ЭЦП должна быть функцией подписываемого сообщения. (Свойства 3, 4).
2. ЭЦП должна использовать информацию, уникальную для отправителя, чтобы предотвратить возможность фальсификации, и отрицания авторства. (Свойство 2, 5).
3. С точки зрения вычислений должно быть нереально фальсифицировать цифровую подпись (ни с помощью создания нового сообщения для имеющейся цифровой подписи, ни с помощью создания фальшивой цифровой подписи для имеющегося сообщения).
4. ЭЦП должна легко генерироваться.
5. ЭЦП должно быть легко распознать и проверить.
6. Алгоритм ЭЦП должен быть удобным для реализации.

Существует несколько схем построения цифровой подписи:

- На основе алгоритмов симметричного шифрования. Обычно данная схема предусматривает наличие в системе третьего лица — арбитра, пользующегося доверием обеих сторон. Авторизацией документа является сам факт шифрования его секретным ключом и передача его арбитру. Симметричные схемы ЭЦП менее распространены.
- На основе алгоритмов асимметричного шифрования. На данный момент такие схемы ЭЦП наиболее распространены и находят широкое применение. Обычно кроме асимметричного алгоритма они используют еще алгоритм хэширования.

На основе асимметричных схем созданы модификации цифровой подписи, отвечающие различным требованиям:

- Групповая цифровая подпись
- Неоспоримая цифровая подпись
- «Слепая» цифровая подпись

- Конфиденциальная цифровая подпись
- Цифровая подпись с доказуемостью подделки
- Доверенная цифровая подпись
- Одноразовая цифровая подпись

Существуют **2 категории ЭЦП**:

- непосредственная цифровая подпись;
- арбитражная цифровая подпись.

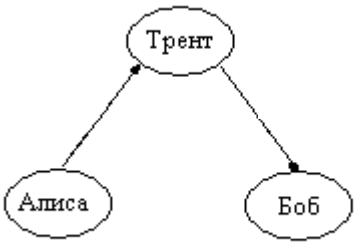
Непосредственная цифровая подпись подразумевает участие только обменивающихся данными сторон (источник, адресат). Предполагается, что адресат знает открытый ключ источника.

Все до сих пор предлагавшиеся схемы непосредственного применения цифровой подписи имеют общее слабое место: пригодность всей схемы зависит от защищенности личного ключа отправителя (утерян, похищен, либо якобы утерян и кто-то воспользовался).

Арбитражная цифровая подпись.

Проблемы, возникающие при использовании непосредственной цифровой подписи, могут решаться с помощью **арбитра** (третьей стороны). Имеется множество протоколов применения арбитражных цифровых подписей. Все они, в общем, строятся следующим образом:

1. Каждое подписанное сообщение отправителя А к адресату В сначала попадает к арбитру (Тренту).
2. Арбитр подвергает сообщение и цифровую подпись тестированию по ряду критериев, чтобы проверить достоверность источника и содержимого сообщения.
3. Арбитр добавляет метку даты-времени.
4. Арбитр подписывает сообщение.
5. Арбитр отправляет сообщение Бобу.



Метки времени необходимы для того, чтобы невозможно было повторно использовать подписанный документ.

Алиса послала Бобу подписанный чек на 100$. Боб отнес его в банк и получил деньги (банк проверил подлинность подписи). Боб, будучи парнем не промах, на следующей неделе снова несет чек в банк и т. д.

Поэтому в цифровой подписи часто вставляют метки времени. В документ включают дату и время подписания. После чего подписывают. Банк сохраняет метку времени в своей базе данных. Если Боб попытается еще раз

получить деньги по чеку Алисы, то это будет замечено и Бобу придется лет 15 изучать криптографические протоколы в тюрьме.

Протокол ЭЦП, использующий традиционное шифрование, арбитраж, обеспечивающий конфиденциальность сообщения

До начала взаимодействия есть общие секретные ключи у всех участников.

(1) $A \to T : ID_A, Ek_{AB}[M], Ek_{TA}[ID_A, H(Ek_{AB}[M])]$

(2) $\mathrm{T} \to \mathrm{B} : \mathrm{Ek_{TB}[ID_A \parallel Ek_{AB}[M] \parallel Ek_{TA}[ID_A \parallel H(Ek_{AB}[M])] \parallel t]}$

k_{AB} – секретный ключ A и B,

k_{TA} – секретный ключ T и A,

k_{TB} – секретный ключ T и B

M – открытое сообщение (прообраз),

H(M) – хэш-код M,

t - метка даты / времени.

ID_A – идентификатор A доказывает, что сообщение не было изменено в пути следования,

$Ek_{AB}[M]$ – сообщение передается в зашифрованном виде, Трент не может прочитать его, но может повторно сосчитать хэш-код, т.е. проверить цифровую подпись.

Трент добавляет метку t (даты/времени) для того, чтобы невозможно было повторно использовать подписанный документ.

Боб не может сам проверить подпись.

Проблемы в традиционных схемах:

1. Сговор арбитра с отправителем с целью отрицания факта отправки;
2. Сговор T и B с целью фальсификации подписи отправителя.

Эти сложности преодолеваются с помощью схемы с открытым ключом.

Протокол ЭЦП, использующий шифрование с открытым ключом, арбитраж, арбитр не видит сообщения.

Протокол предполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: личный и открытый. Личный ключ хранится абонентом в тайне и используется им для формирования электронной цифровой подписи. Открытый ключ известен всем другим пользователям и предназначен для проверки электронной цифровой подписи получателем подписанного документа.

(1) $A \to T : ID_A, Ekr_A[ID_A, Eku_B[Ekr_A[M]])]$

(2) $T \to B : Ekr_T[ID_A, Eku_B[Ekr_A[M]), t]$,

Трент расшифрует подпись открытым ключом Алисы. Поставит метку даты-времени.

Внутренне дважды шифрованное сообщение защищено от арбитра.

Арбитр подписывает сообщение своим личным ключом.

Преимущества:

1. До начала обмена нет никакой информации в совместном распоряжении сторон, что предотвращает возможность сговора с целью обмана (никаких совместных ключей);
2. Некорректно датированное сообщение не может быть передано, т. к. Трент добавляет метку даты-времени t;
3. Содержание M является секретным для Трента (зашифровано открытым ключом Боба)

Создание и верификация ЭЦП на основе асимметричного алгоритма

В отличие от асимметричных алгоритмов шифрования, в которых шифрование производится с помощью открытого ключа, а расшифрование — с помощью личного, в асимметричных схемах цифровой подписи подписание производится с применением личного ключа, а проверка подписи — с применением открытого.

При использовании асимметричной схемы ЭЦП может быть создана:

- с помощью шифрования всего документа личным ключом отправителя;
- с помощью шифрования хэш-кода документа личным ключом отправителя.

Использование хэш-функций даёт следующие преимущества:

1) Меньший объем ЭЦП.

Поскольку подписываемые документы переменного (и как правило достаточно большого) объёма, в схемах ЭЦП чаще подпись ставится не на сам документ, а на его хэш-код.

2) Большая скорость вычисления ЭЦП.

Алгоритмы вычисления хэш-кода являются быстрыми. Поэтому формировать хэш документа и подписывать его получается намного быстрее, чем подписывать сам документ.

3) Возможность легкой замены одного алгоритма хэширования другим.

Хэш-функции не являются частью алгоритма ЭЦП, поэтому в схеме может быть использована любая криптографическистойкая хэш-функция.

Стоит подчеркнуть, что т.к. подписывается не сообщение (произвольной длины), а его хэш-код, поэтому возможны коллизии и одна подпись может быть действительна для нескольких сообщений с одинаковым хэшем. Поэтому выбор стойкой хэш-функции очень важен для всей системы в целом.

Технология применения электронной цифровой подписи предполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: открытый и личный. Личный ключ хранится абонентом в тайне и используется для формирования ЭЦП. Открытый ключ предназначен для проверки ЭЦП получателем документа. Для распространения открытых ключей может

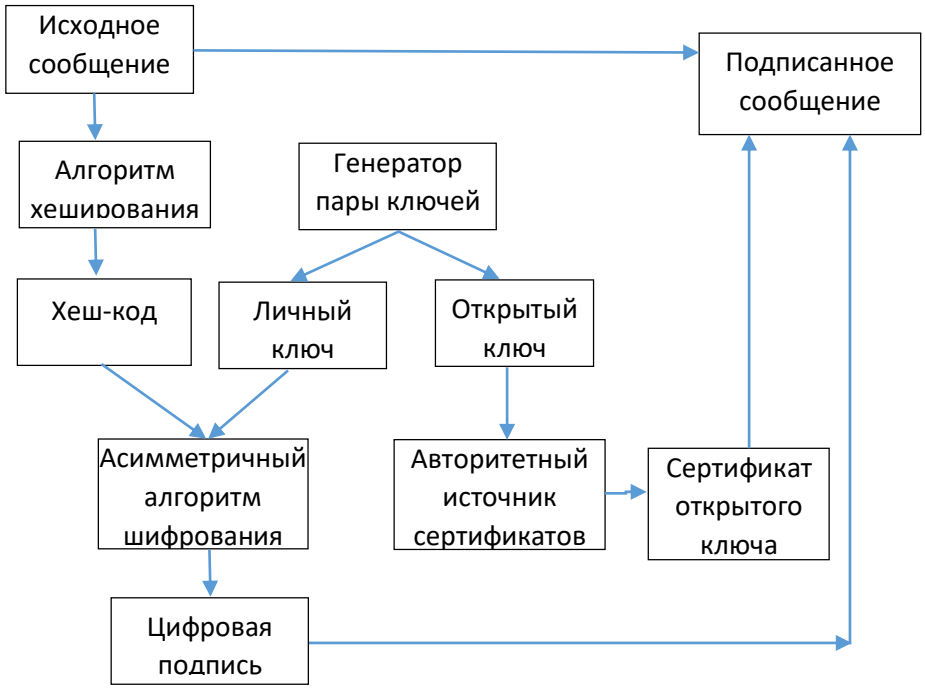
использоваться один из возможных способов, например, сертификаты открытых ключей.

Общая схема применения электронной цифровой подписи изображена на рисунке.

Берется исходное сообщение и создается его хеш-код при помощи одного из алгоритмов хеширования. Затем хеш-код шифруется при помощи личного ключа отправителя сообщения. Результат шифрования и называют электронной цифровой подписью

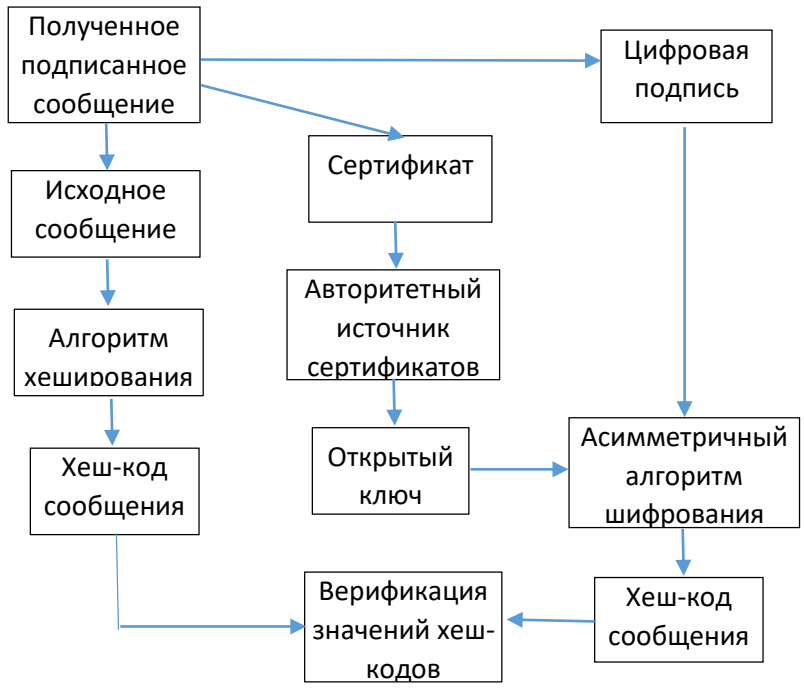
Никто, кроме владельца личного ключа, не сможет создать такую подпись, даже располагая оригиналом исходного сообщения. Никто не сможет изменить сообщение или создать поддельное сообщение так, чтобы обман не раскрылся.

Подписанное сообщение формируется объединением исходного сообщения, его цифровой подписи и сертификата открытого ключа, соответствующего тому личному ключу, которым шифровался хеш.



При получении подписанное сообщение верифицируется. Получатель хочет убедиться в том, что сообщение действительно отправлено отправителем, а не кем-либо еще. Также получатель хочет убедиться, что сообщение не было изменено в пути следования.

Схема верификации ЭЦП:



Полученное сообщение разбивается на три компоненты – исходное сообщение, открытый ключ отправителя, цифровая подпись.

Получатель извлекает сертификат открытого ключа отправителя и проверяет его с помощью центра выдачи сертификатов. С помощью открытого ключа, содержащегося в сертификате, получатель может расшифровать хеш-код. После этого заново вычисляется хеш-код сообщения и сравнивается с полученным значением. Если хеши совпали, сообщение аутентично и не изменено.

Асимметричные схемы ЭЦП:

- Схемы, основанные на алгоритме RSA: FDH (Full Domain Hash), вероятностная схема RSA-PSS (Probabilistic Signature Scheme), схемы стандарта PKCS#1 и другие
- Схема Эль-Гамаля
- Схема Шнорра
- Американские стандарты электронной цифровой подписи DSS: DSA, ECDSA
- Российские стандарты электронной цифровой подписи: ГОСТ Р 34.10-94 (в настоящее время не действует), ГОСТ Р 34.10-2001 (не рекомендован к использованию после 31 декабря 2017 года), ГОСТ Р 34.10-2012
- Стандарт Евразийского союза: ГОСТ 34.310-2004 полностью идентичен российскому стандарту ГОСТ Р 34.10-2001
- Белорусский стандарт электронной цифровой подписи СТБ 1176.2-99 (в настоящее время не действует), СТБ 34.101.45-2013 и т.д.

Алгоритм цифровой подписи RSA

Первой и наиболее известной во всем мире системой ЭЦП стала система RSA. Сначала необходимо вычислить пару ключей (личный и открытый). Для этого отправитель (автор) электронного письма:

1. Выбирает 2 больших простых числа P и Q;
2. Находит их произведение N = P*Q;
3. Находит значение функции Эйлера φ(N) = (P-1)(Q-1);
4. Вычисляет открытый ключ e из условий: $e \le \varphi(N),\ \text{НОД}(e, \varphi(N)) = 1$
5. Вычисляет личный ключ d из условий: $d < N,\ \ e * d \equiv 1 (\mathrm{mod}\, \varphi(N))$

Пара чисел (e, N) является открытым ключом KUa. Эту пару чисел автор передает партнерам по переписке. Числа (d, P,Q) сохраняются как личный ключ KRa.

Обобщенная схема формирования и проверки ЭЦП по алгоритму RSA:

Отправитель:

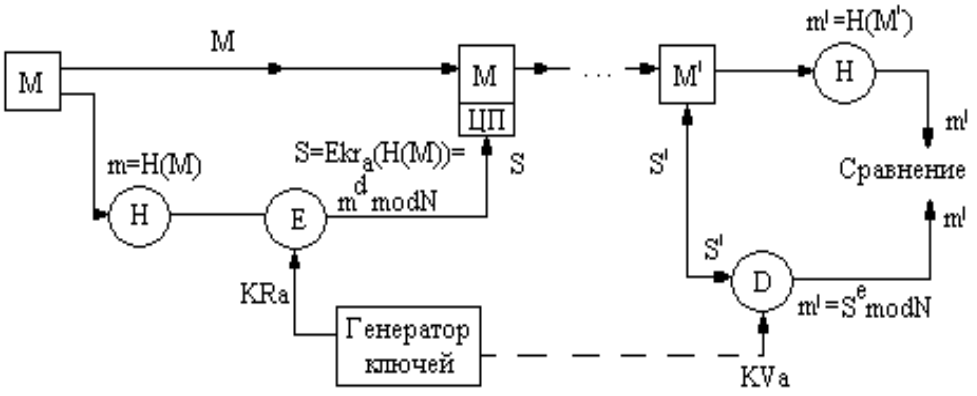
1. Сообщение M поступает на вход функции хэширования H, вычисляется хэш-код m=H(M).
2. Хеш-код шифруется личным ключом отправителя KRa, таким образом получается $\text{ЦП}: S = m^d \bmod N = (H(M))^d \bmod N$.
3. Параметры (M,S) передаются получателю (S добавляется к M).

Получатель:

1. Принимает сообщение M' и повторно вычисляет его хеш-код: m'=H(M')
2. Расшифровывает цифровую подпись, используя открытый ключ отправителя KUa:

$$m' = S^e \bmod N.$$

Если вычисленный хеш-код совпадает с расшифрованным, то цифровая подпись считается подлинной. Т. к. только отправитель знает свой личный ключ, то подпись мог поставить только он.



Недостатки алгоритма цифровой подписи RSA:

1. При вычислении модуля N, ключей e и d необходимо проверять большое количество дополнительных условий, что сделать практически

трудно. Невыполнение любого из этих условий делает возможным фальсификацию цифровой подписи.

2. Для обеспечения криптостойкости цифровой подписи RSA необходимо использовать при вычислениях целые числа не менее 2^{512} (~10^{154}) каждое, что требует больших вычислительных затрат.

3. ЭЦП RSA уязвима к так называемой мультипликативной атаке.

Алгоритм RSA позволяет злоумышленнику без знания личного ключа d сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования, уже подписанных документов.

Допустим, злоумышленник может сконструировать 3 сообщения M_1, M_2, M_3, у которых хэш-значения

$$m_1 = h(M_1),\ m_2 = h(M_2),\ m_3 = h(M_3),$$

причем, $m_3 = (m_1 * m_2)(\mathrm{mod} N)$.

Допустим, что для 2-ух сообщений M_1 и M_2 получены законные подписи:

$$S_1 = m_1^{\,d} \bmod N,\ S_2 = m_2^{\,d} \bmod N.$$

Тогда злоумышленник может легко вычислить подпись S_3 для документа M_3, даже не зная секретного ключа d:

$$S_3 = (S_1 * S_2) \bmod N.$$

Более надежный и удобный для реализации на ПК алгоритм цифровой подписи был разработан в 1984 году Тахером Эль Гамалем. В 1991 году этот алгоритм был выбран за основу для национального стандарта США DSS.

Стандарты цифровой подписи (DSS, ГОСТ, СТБ)

DSS (Digital Signature Standard) — американский стандарт ЭЦП. Впервые опубликован в 1994 в документе FIPS-186 (Federal Information Processing Standards). Стандарт несколько раз обновлялся, последняя версия FIPS-186-4 (июль 2013).

В основе стандарта DSS лежит алгоритм цифровой подписи **DSA (Digital Signature Algorithm) –** асимметричный криптографический алгоритм. Этот алгоритм может быть использован только для генерации цифровой подписи и не предназначен для других целей (шифрование, распределение ключей). Алгоритм основан на вычислительной сложности взятия логарифмов в конечных полях.

Алгоритм разработан NIST - Национальным институтом стандартов и технологий (США) в 1991 году, опубликован в 1993 году.

Цифровая подпись служит для установления целостности данных и для установления подлинности подписавшейся стороны. Получатель подписанных данных может использовать цифровую подпись для

доказательства третьей стороне факта, что подпись действительно сделана отправляющей стороной.

DSA фактически включает в себя два алгоритма: для создания подписи и для ее верификации.

Оба алгоритма в начале вычисляют хэш-код сообщения, используя криптографическистойкую хеш-функцию. На стороне отправителя используется хэш-код и личный ключ для создания подписи. На стороне получателя используется хэш-код сообщения, подпись и открытый ключ для проверки подписи.

Используемая хэш-функция описана в американском стандарте функции хеширования **SHS (Secure Hash Standard)**. Применяется алгоритм хеширования **SHA**.

В первой версии стандарта использовалась SHA-1, в последней версии можно использовать любой алгоритм семейства SHA-2. В августе 2015 был опубликован стандарт FIPS-202, описывающий новую хэш-функцию SHA-3. Возможно в скором времени в DSS также включат SHA-3.

Размер личного ключа: 160-256 бит.

Размер открытого ключа: 1024-3072 бит.

Размер подписи: два числа по 160-256 бит.

Параметры DSA

1. p – простое число p, где $2^{L-1} < p < 2^{L}$, $512 =< L =< 1024$ и L кратно 64
2. q – простой делитель (p-1), причем $2^{159} < q < 2^{160}$, т.е длиной 160 бит
3. $g = h^{(p-1)/q} \bmod p$, где h любое целое число $1 < h < p - 1$ такое, что $h^{(p-1)/q} \bmod p > 1$
4. x – случайное или псевдослучайное целое число, где $0 < x < q$ (160-битовое)
5. $y = g^{x} \bmod p$

6. k – случайное или псевдослучайное целое число, где 0 < k < q.

Целые p, q и g могут быть открытыми и могут быть общими для группы пользователей. Их называют компонентами глобального открытого ключа.

x , y - личный и открытый ключи пользователя. Пользователь выбирает личный ключ x и вычисляет открытый y.

k - секретный номер сообщения пользователя, уникальный для каждого выполнения подписи.

Генерация подписи

Подписью сообщения M является пара чисел r и S, где

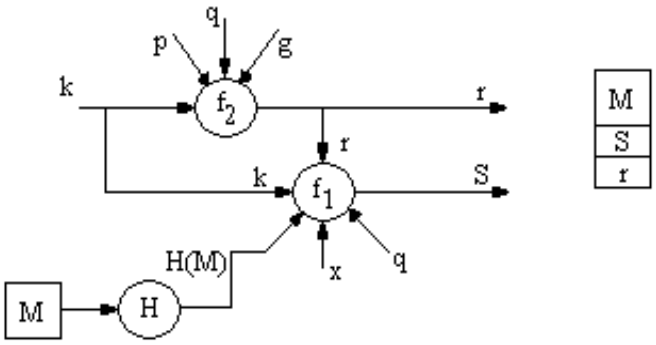
$$\begin{cases} r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q \\ S = f_1(H(M), k, x, r, q) = (k^{-1}(SHA(M) + xr)) \bmod q \end{cases}$$

SHA(M) — 160-битная бинарная строка.

k^{-1} – мультипликативное обратное.

Если r = 0 или S = 0, должно быть сгенерировано новое k и вычислена новая подпись. Если подпись вычислялась правильно, вероятность того, что r = 0 или S = 0 очень мала.

Подпись (r, S) вместе с сообщением M пересылается получателю.



Проверка подписи

Числа p, q, g (глобальный открытый ключ) и y (открытый ключ) находятся в открытом доступе.

Пусть M', r' и S' — полученные версии M, r и S соответственно. При проверке подписи сначала нужно посмотреть, выполняются ли следующие неравенства:

0 < r' < q

0 < s' < q.

Если хотя бы одно неравенство не выполнено, подпись должна быть отвергнута. Если условия неравенств выполнены, производятся следующие вычисления:

$w = f_3(s', q) = (s')^{-1} \bmod q$

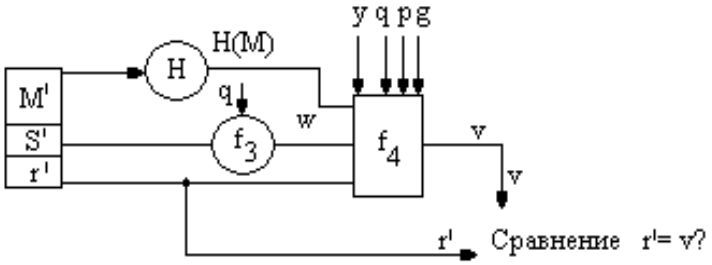
u1 = ((SHA(M')w) mod q

u2 = ((r')w) mod q

$v = f_4(y, q, g, p, SHA(M'), w, r') = (((g)^{ul} (y)^{u2}) \bmod p) \bmod q$.

Если v = r', то подлинность подписи подтверждена.

Если v ≠ r', то сообщение могло быть изменено, сообщение могло быть неправильно подписано или сообщение могло быть подписано мошенником. В этом случае полученные данные следует рассматривать как поврежденные.



Интенсивные вычисления, при создании цифровой подписи потребуются только при возведении в степень $g^x \bmod p$. Т. к. это значение не зависит от M, то его можно вычислить заранее. Отправитель вообще может заранее вычислить целый ряд значений r, чтобы пользоваться ими по мере необходимости. Определить мультипликативное обратное (k^{-1}) тоже можно заранее.

DSA медленнее RSA. Но работу практических реализаций DSA часто можно ускорить с помощью предварительных вычислений.

В любой реализации DSA для безопасности системы очень важен хороший генератор случайных чисел.

Российский стандарт ЭЦП

В России правовые условия использования электронной цифровой подписи в электронных документах регламентирует Федеральный закон Российской Федерации от 6 апреля 2011 года № 63-ФЗ «Об электронной подписи».

В результате было введено определение трех видов электронных подписей:

- **Простой электронной подписью** является электронная подпись, которая посредством использования кодов, паролей или иных средств подтверждает факт формирования электронной подписи определенным лицом.
- **Усиленной неквалифицированной электронной подписью** является электронная подпись, которая:
  1. получена в результате криптографического преобразования информации с использованием ключа электронной подписи;
  2. позволяет определить лицо, подписавшее электронный документ;
  3. позволяет обнаружить факт внесения изменений в электронный документ после момента его подписания;
  4. создается с использованием средств электронной подписи.
- **Усиленной квалифицированной электронной подписью** является электронная подпись, которая соответствует всем признакам

неквалифицированной электронной подписи и следующим дополнительным признакам:

1. ключ проверки электронной подписи указан в квалифицированном сертификате;
2. для создания и проверки электронной подписи используются средства электронной подписи, получившие подтверждение соответствия требованиям, установленным в соответствии с 63-ФЗ

С 1 января 2013 года гражданам может быть выдана универсальная электронная карта, в которую встроена усиленная квалифицированная электронная подпись.

Российский стандарт ЭЦП ГОСТ Р 34.10-2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.» Введен 1 января 2013 года.

Электронная цифровая подпись СТБ 34.101.45-2013

В РБ стандарт цифровой подписи введен в 2013 году. СТБ 34.101.45-2013 «Информационные технологии и безопасность. Алгоритмы ЭЦП и транспорта ключа на основе эллиптических кривых.»

Стандарт устанавливает алгоритмы выработки и проверки электронной цифровой подписи (ЭЦП), алгоритмы транспорта ключа, а также сопровождающие их алгоритмы генерации и проверки параметров эллиптической кривой, генерации личных и открытых ключей, проверки открытых ключей. Стандарт применяется при разработке средств криптографической защиты информации, в том числе средств ЭЦП и шифрования.

В настоящем стандарте использованы ссылки на следующие технические нормативные правовые акты в области технического нормирования и стандартизации (далее — ТНПА):

СТБ 34.101.17-2012 Информационные технологии и безопасность. Синтаксис запроса на получение сертификата

СТБ 34.101.19-2012 Информационные технологии и безопасность. Форматы сертификатов и списков отозванных сертификатов инфраструктуры открытых ключей

СТБ 34.101.23-2012 Информационные технологии и безопасность. Синтаксис криптографических сообщений

СТБ 34.101.26-2012 Информационные технологии и безопасность. Онлайновый протокол проверки статуса сертификата (OCSP)

СТБ 34.101.31-2011 Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности

СТБ 34.101.47-2012 Информационные технологии и безопасность. Криптографические алгоритмы генерации псевдослучайных чисел

В настоящем стандарте применяют следующие термины с соответствующими определениями (приведены некоторые из них):

Электронная цифровая подпись (ЭЦП) - контрольная характеристика сообщения, которая вырабатывается с использованием личного ключа, проверяется с использованием открытого ключа, служит для контроля целостности и подлинности сообщения и обеспечивает невозможность отказа от авторства.

Токен ключа - сообщение, которое передается от одной стороны другой при транспорте ключа и представляет собой транспортируемый ключ в защищенной форме, а также данные, необходимые получателю для снятия защиты.

Транспорт ключа - конфиденциальная передача ключа от одной стороны другой.

Настоящий стандарт определяет алгоритмы ЭЦП, которые предназначены для контроля целостности и подлинности сообщений. Автор сообщения использует свой личный ключ для выработки ЭЦП, а связанный с личным ключом открытый ключ используется другими сторонами для проверки ЭЦП. При правильном управлении ключами корректность проверяемой подписи означает, что она была выработана владельцем личного ключа и после этого сообщение не изменялось.

Только владелец личного ключа может выработать корректную ЭЦП, что не позволяет ему отказаться от авторства сообщения и может быть использовано другими сторонами для доказательства такого авторства.

Алгоритмы выработки и проверки ЭЦП построены по схеме Шнорра [3]. При выполнении алгоритмов используются вычисления в группе точек эллиптической кривой над конечным простым полем. В стандарте определяются алгоритмы генерации и проверки параметров, описывающих искомую группу. Определены также алгоритм генерации пары ключей (личного и открытого) и алгоритм проверки открытого ключа.

Проверка знания личного ключа, удостоверение принадлежности открытого ключа и проверка такой принадлежности реализуются с помощью дополнительных методов и средств, в совокупности называемых инфраструктурой открытых ключей. Например, в СТБ 34.101.19 определяются элементы инфраструктуры на основе сертификатов открытых ключей.

Параметры эллиптической кривой, личный и открытый ключи могут быть использованы не только для контроля целостности и подлинности, но и для обеспечения конфиденциальности защищаемой информации. В стандарте определяются алгоритмы транспорта ключа, предназначенные для защищенной передачи ключей и других секретных данных между двумя сторонами. С помощью транспортируемого ключа стороны могут выполнять шифрование или другие криптографические операции.

Для реализации транспорта отправитель выполняет алгоритм создания токена ключа и передает полученный токен получателю. Получатель выполняет алгоритм разбора токена и восстанавливает транспортируемый ключ. При создании токена отправитель использует открытый ключ получателя. При разборе токена получатель использует свой личный ключ.

Алгоритмы ЭЦП построены так, что злоумышленнику вычислительно трудно решить задачу подделки ЭЦП. В этой задаче злоумышленник получает параметры эллиптической кривой и открытый ключ ЭЦП. Злоумышленник не знает личный ключ, но может передавать для подписи на нем произвольные сообщения, получать и анализировать результаты.

Ему требуется построить корректную ЭЦП к любому сообщению, отличному от ранее подписанных.

Стойкость алгоритмов ЭЦП определяется уровнем $l \in \{128, 192, 256\}$. На уровне l для подделки ЭЦП злоумышленнику требуется выполнить порядка $2l$ операций. Стойкость основывается на сложности дискретного логарифмирования в группе точек эллиптической кривой и на стойкости используемых функций хеширования.

Уровень l определяет длины параметров, ключей, подписей, а также быстродействие алгоритмов ЭЦП. Следует учитывать, что с ростом l, кроме повышения стойкости, снижается быстродействие алгоритмов.

Для алгоритмов транспорта ключа вводятся аналогичные уровни стойкости $l \in \{128, 192, 256\}$. На уровне l для определения транспортируемого ключа по токену и открытому ключу получателя злоумышленнику требуется выполнить порядка $2l$ операций.

На уровне стойкости l должна использоваться функция h, значениями которой являются двоичные слова длины $2l$. Например, при $l = 128$ в качестве h можно выбрать функцию, заданную алгоритмом belt-hash.

Транспорт ключа

В алгоритмах транспорта ключа используется заголовок ключа. Заголовок представляет собой слово $I \in \{0, 1\}128$, которое описывает открытые атрибуты транспортируемого ключа, например данные об отправителе, получателе, назначении ключа. Формат заголовка фиксируется в конкретной информационной системе. Заголовок может передаваться вместе с токеном ключа. Если необходимости в передаче атрибутов ключей нет, то могут использоваться постоянные заголовки, которые не требуется передавать. По умолчанию $I = 0128$.

Один и тот же ключ может транспортироваться одновременно нескольким сторонам. В этом случае отправитель должен создать токены ключа для каждой из сторон. Если стороны-получатели используют одинаковые параметры эллиптической кривой, то при создании токенов может использоваться один и тот же одноразовый личный ключ k.

Используется алгоритм хэширования belt-hash, описанный в стандарте СТБ

Используется алгоритм зашифрования belt-block, определенный в СТБ 34.101.31. Входными данными алгоритма являются слово $X \in \{0, 1\}128$ и ключ $\theta \in \{0, 1\}256$, выходными — зашифрованное слово $Y \in \{0, 1\}128$.

3 Лабораторный практикум

3.1 Лабораторная работа № 1

КЛАССИЧЕСКИЕ МЕТОДЫ ШИФРОВАНИЯ

Цель работы

Познакомиться с основными криптографическими терминами и моделью традиционного шифрования. Изучить типы криптосистем. Изучить особенности классических методов шифрования на примере конкретных алгоритмов.

Теоретические сведения

Методы шифрования применяются не одну тысячу лет.

Классическими шифрами называются шифры, которые использовались в докомпьютерную эпоху. В настоящее время они не применяются на практике, так как легко взламываются с помощью современных вычислительных средств. Однако классические шифры представляют исторический и учебный интерес, поскольку основаны на тех же принципах, что и современные шифры. Все они относятся к более широкой группе симметричных шифров.

Симметричные шифры - это шифры, в которых для шифрования и расшифрования используется один и тот же ключ. Их называют еще **традиционными**, **криптосистемами с одним ключом** или **криптосистемами с секретным ключом**. Современные симметричные шифры широко применяются на практике. Модель шифрования с секретным ключом была предложена Клодом Шенноном, основателем теории информации.

Ниже рассмотрены примеры классических шифров.

1. Обобщенный алгоритм Цезаря.

Самым древним и самым простым из известных подстановочных шифров является шифр, использовавшийся Юлием Цезарем. В шифре Цезаря каждая буква алфавита заменяется буквой, которая находится на 3 позиции дальше в этом же алфавите. Алфавит считается циклическим, то есть после Z идет A.

Если каждой букве назначить числовой эквивалент (a=1, b=2, и т.д.), то алгоритм можно выразить следующими формулами. Каждая буква открытого текста P (Plaintext) заменяется буквой шифрованного текста C (Ciphertext):

$$C = E_3(P) = (P + 3) \bmod(26),$$

где 26 – число букв в алфавите; 3 – ключ.

В общем случае сдвиг может быть любым, поэтому **обобщенный алгоритм Цезаря** записывается формулой:

$$C = E_k(P) = (P + K) \bmod(N),$$

где К – ключ, который может принимать значения в диапазоне от 1 до 25; N - размерность алфавита.

Алгоритм расшифрования:

$$P = D_k(C) = (C - K) \bmod(N)$$

Шифр Цезаря легко вскрывается с помощью лобовой атаки или на основе анализа частот появления букв в шифртексте.

2. Система Цезаря с ключевым словом.

Особенностью этой системы является использование **ключевого слова** для смещения и изменения порядка символов в алфавите подстановки.

Выберем некоторое число k, 0 <= k < N-1 (N- размерность алфавита), и слово или короткую фразу в качестве ключевого слова. Желательно, чтобы все буквы ключевого слова были различными. В противном случае надо удалить повторяющиеся символы до шифрования.

Пусть выбран ключ:

- DIPLOMAT - ключевое слово;
- k = 5 - ключевое число.

Ключевое слово записывается под буквами алфавита, начиная с буквы, числовой код которой совпадает с выбранным числом k:

```
 0  1  2  3  4  5
 A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
             D  I  P  L  O  M  A  T
```

Оставшиеся буквы алфавита подстановки записываются после ключевого слова в алфавитном порядке:

```
 0  1  2  3  4  5
 A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
 V  W  X  Y  Z  D  I  P  L  O  M  A  T  B  C  E  F  G  H  J  K  N  Q  R  S  U
```

Достоинством системы Цезаря с ключевым словом является то, что количество возможных ключевых слов практически неисчерпаемо. Недостатком этой системы является возможность взлома шифра на основе анализа частот появления букв.

3. Шифр, использующий линейное преобразование

Алгоритм шифрования:

1) Определяем порядковый номер буквы в открытом тексте - n;
2) Определяем код буквы в алфавите: X;
3) Вычисляем смещение S по формуле:

$$S = (a*n + b) \bmod N$$

Пара чисел (a, b) – ключ шифрования;
N – количество символов в алфавите.

4) Определяем код буквы шифртекста в алфавите: $Y = (X+S) \bmod N$

Алгоритм расшифрования:

1) Определяем порядковый номер буквы в шифртексте - n;
2) Определяем код буквы шифртекста в алфавите - Y;
3) Вычисляем смещение - S;
4) Определяем код буквы открытого текста: $X = (Y-S) \bmod N$

Пример:

Возьмем русский алфавит – 33 буквы + «пробел».

N = 34

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		17	18	19	20	21	22
А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О		П	Р	С	Т	У	Ф

23	24	25	26	27	28	29	30	31	32	33	34
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	

Зашифруем слово РЕКА с ключом (3, 1).

1. Определяем порядковый номер буквы Р в открытом тексте: n=1.
2. Определяем код буквы Р в алфавите: X=18.
3. Вычисляем смещение: $S = (3 * 1+ 1) \bmod 34 = 4$
4. Определяем код буквы шифртекста: $(18+4) \bmod 34 = 22$

Это буква Ф.

Продолжая шифрование согласно алгоритму, получим шифртекст: ФЛФМ

4. Шифр, использующий нелинейное преобразование

Отличается от предыдущего тем, что в ходе шифрования используется нелинейное преобразование:

$$S = (a*n^2 + b*n + c) \bmod N,$$

где (a,b,c) – ключ шифрования, N – размерность алфавита, n – порядковый номер буквы в открытом тексте.

Алгоритм шифрования: $C_i = (P_i + S_i) \bmod N$

Алгоритм расшифрования: $P_i = (C_i - S_i) \bmod N$

Пример:

Возьмем русский алфавит – 33 буквы + «пробел».

Зашифруем слово РЕСПУБЛИКА с ключом (8, 4, 9).

1. Определяем порядковый номер буквы Р в открытом тексте: n=1.
2. Определяем код буквы Р в алфавите: X=18.
3. Вычисляем смещение: $S = (8 * 1^2 + 4*1 + 9) \bmod 34 = 21$
4. Определяем код буквы шифртекста: $(18+21) \bmod 34 = 39 \bmod 34 = 5$

Это буква Д.

Продолжая шифрование, получим шифртекст: ДУИ КП СЧ

5. Шифр, использующий сложение с ключевым словом

Если длина ключевого слова меньше, чем длина текста, то ключевое слово повторяется.

Алгоритм шифрования:

1) Определяем порядковый номер буквы в открытом тексте - i;
2) Определяем код i-й буквы текста в алфавите: X_i;
3) Определяем код i-й буквы ключа в алфавите: $К_i$;
4) Вычисляем код буквы шифртекста в алфавите:

$$У_i = (X_i + К_i) \bmod N$$

N – количество элементов в алфавите.

Алгоритм расшифрования:

1) Определяем порядковый номер буквы в открытом тексте - i;
2) Определяем код i-й буквы шифртекста в алфавите: Y_i;
3) Определяем код i-й буквы ключа в алфавите: $К_i$;
4) Вычисляем код буквы шифртекста в алфавите:

$$X_i = (У_i - К_i) \bmod N$$

Пример:

Возьмем русский алфавит – 33 буквы + «пробел».

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф

23	24	25	26	27	28	29	30	31	32	33	34
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	

Зашифруем слово РЕСПУБЛИКА с ключом КРЫША.

1. Определяем порядковый номер буквы Р в открытом тексте: n=1.
2. Определяем код буквы Р в алфавите: $X_1 = 18$.
3. Определяем код i-й буквы ключа в алфавите: $К_1 = 12$
4. Вычисляем код буквы шифртекста в алфавите:

$У_i = (18 + 12) \bmod 34 = 30$

Это буква Ь.

Продолжая шифрование согласно алгоритму, получим шифртекст:
ЦМЗФМЭДГБ

6. Шифрующие таблицы Трисемуса

В 1508г. аббат из Германии Иоганн Трисемус впервые описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово, причем повторяющиеся буквы отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку.

Пример.

Для русского алфавита шифрующая таблица может иметь размер 4x8. Выберем в качестве ключа слово БАНДЕРОЛЬ. Шифрующая таблица с таким ключом будет выглядеть следующим образом:

Б	А	Н	Д	Е	Р	О	Л
Ь	В	Г	Ж	З	И	Й	К
М	П	С	Т	У	Ф	Х	Ц
Ч	Ш	Щ	Ъ	Ы	Э	Ю	Я

При шифровании находят в этой таблице очередную букву открытого текста и записывают в шифртекст букву, расположенную ниже ее в том же столбце. Если буква текста оказывается в нижней строке таблицы, то берут самую верхнюю букву из того же столбца.

Такие шифры называют **монограммными** и **моноалфавитными**, т.к. шифрование выполняется по одной букве и используется один алфавит.

7. Биграммный шифр Плейфейера.

Одним из наиболее известных шифров, базирующихся на методе многобуквенного шифрования, является шифр Плейфейера, в котором биграммы открытого текста рассматриваются как самостоятельные единицы, преобразованные в заданные биграммы шифрованного текста. Он применялся Великобританией во время Первой мировой войны

Алгоритм Плейфейера основан на использовании матрицы букв размерности 5x5, созданной на основе некоторого ключевого слова.

Например:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

В данном случае ключевым словом является monarchy(монархия). Матрица создается путем размещения букв, использованных в ключевом слове, слева направо и сверху вниз (повторяющиеся буквы отбрасываются). Затем оставшиеся буквы алфавита размещаются в естественном порядке в оставшихся строках и столбцах матрицы. Буквы I и J считаются одной и той

же буквой. Открытый текст шифруется порциями по две буквы в соответствии со следующими правилами.

1. Если оказывается, что повторяющиеся буквы открытого текста образуют пару для шифрования, то между этими буквами вставляется специальная буква-заполнитель, например x. В частности, такое слово как balloon будет преобразовано к виду ba lx lo on.

2. Если буквы открытого текста попадают в одну и ту же строку матрицы, каждая из них заменяется буквой, следующей за ней в той же строке справа – с тем условием, что для замены последнего элемента строки матрицы служит первый элемент той же строки. Например, ar шифруется как RM.

3. Если буквы открытого текста попадают в один и тот же столбец матрицы, каждая из них заменяется буквой, стоящей в том же столбце сразу под ней, с тем условием, что для замены самого нижнего элемента столбца матрицы берется самый верхний элемент того же столбца. Например, mu шифруется как CM.

4. Если не выполняется ни одно из приведенных выше условий, каждая буква из пары букв открытого текста заменяется буквой, находящейся на пересечении содержащей эту букву строки матрицы и столбца, в котором находится вторая буква открытого текста. Например, hs шифруется как BP, а ea – как IM (или JM, по желанию шифровальщика).

7. Шифр «двойной квадрат» Уитстона.

В 1854г. англичанин Чарльз Уитстон разработал новый метод шифрования биграммами. При этом используются сразу две таблицы, размещенные по одной горизонтали. Шифр оказался очень надежным и применялся Германией даже в годы Второй мировой войны.

Рассмотрим пример. Пусть имеются две таблицы со случайно расположенными в них русскими алфавитами. Перед шифрованием исходное сообщение разбивают на биграммы. Каждая биграмма шифруется отдельно. Первую букву биграммы находят в левой таблице, а вторую – в правой. Затем строят прямоугольник так, чтобы буквы биграммы лежали в его противоположных вершинах. Другие две вершины этого прямоугольника дают буквы биграммы шифртекста.

Ж	Щ	Н	Ю	Р
И	Т	Ь	Ц	Б
Я	М	Е	.	С
В	Ы	П	Ч	
:	Д	У	О	К
З	Э	Ф	Г	Ш
Х	А	,	Л	Ъ

И	Ч	Г	Я	Т
,	Ж	Ь	М	О
З	Ю	Р	В	Щ
Ц	:	П	Е	Л
Ъ	А	Н	.	Х
Э	К	Ц	Ш	Д
Б	Ф	У	Ы	

Предположим, что шифруется биграмма ИЛ. Буква И находится в столбце 1 и строке 2 левой таблицы. Буква Л в столбце 5 и строке 4 правой таблицы. Это означает, что прямоугольник образован строками 2 и 4, а также столбцами 1 и 5. Следовательно, в биграмму шифртекста входят буква О, расположенная в столбце 5 и строке 2 правой таблицы, и буква В, расположенная в столбце 1 и строке 4 левой таблицы.

Если обе буквы биграммы сообщения лежат в одной строке, то и буквы шифртекста берут из этой же строки. Первую букву биграммы шифртекста берут из левой таблицы в столбце, соответствующем второй букве биграммы сообщения. Вторая буква биграммы шифртекста берется из правой таблицы в столбце, соответствующем первой букве биграммы сообщения. Поэтому биграмма сообщения ТО превращается в биграмму шифртекста БЖ.

Для получения шифрующих таблиц можно использовать ключевые слова, аналогично шифру Трисемуса.

9. Шифр Виженера.

Для усовершенствования простого моноалфавитного шифра можно использовать несколько моноалфавитных подстановок, применяемых в ходе шифрования открытого текста в зависимости от определенных условий.

Семейство шифров, основанных на применении таких методов шифрования, называется **полиалфавитными шифрами**. Подобные методы шифрования обладают следующими общими свойствами:

1. Используется набор связанных моноалфавитных подстановок.

2. Имеется некоторый ключ, по которому определяется, какое конкретное преобразование должно применяться для шифрования на данном этапе.

Самым широко известным и одновременно простым алгоритмом такого рода является шифр Виженера (Vigenire). Этот шифр базируется на наборе правил моноалфавитной подстановки, представленных 26 шифрами Цезаря со сдвигом от 0 до 25. Каждый из таких шифров можно обозначить ключевой буквой текста. Например, шифр Цезаря, для которого смещение равно 3, обозначается ключевой буквой d.

Для облегчения понимания и применения этой схемы была предложена матрица, названная "табло Виженера". Все 26 шифров располагаются по горизонтали, и каждому из шифров соответствует своя ключевая буква, представленная в крайнем столбце слева. Алфавит, соответствующий буквам открытого текста, находится в первой сверху строке таблицы. Процесс шифрования прост - необходимо по ключевой букве *x* и букве открытого текста *y* найти букву шифрованного текста, которая находится на пересечении строки *x* и столбца *y*. В данном случае такой буквой является буква *v*.

Таблица 1. Современное табло Виженера

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Чтобы зашифровать сообщение, нужен ключ, имеющий ту же длину, что и само сообщение. Обычно ключ представляет собой повторяющееся нужное число раз ключевое слово, чтобы получить строку подходящей длины.

Например, если ключевым словом является deceptive, сообщение "we are discovered save yourself" шифруется следующим образом.

Ключ: deceptivedeceptivedeceptive

Открытый текст: wearediscoveredsaveyourself

Шифрованный текст: zicvtwqngrzgvtsavzhcqyglmgj

Расшифровать текст также просто - буква ключа определяет строку, буква шифрованного текста, находящаяся в этой строке, определяет столбец, и в этом столбце в первой строке таблицы будет находиться соответствующая буква открытого текста.

Преимущество этого шифра заключается в том, что для представления одной и той же буквы открытого текста в шифрованном тексте имеется много различных вариантов - по одному на каждую из неповторяющихся букв ключевого слова. Таким образом, скрывается информация, характеризующая частотность употребления букв. Но и с помощью данного метода все же не удается полностью скрыть влияние структуры открытого текста на структуру шифрованного.

10. Шифрование методом Вернама.

Метод предложен инженером компании AT&T Гилбертом Вернамом (Gilbert Vernam) в 1918г. Его система оперирует не буквами, а двоичными числами. Кратко ее можно выразить формулой

$$C_i = p_i \oplus k_i,$$

где

p_i - i-я двоичная цифра открытого текста,

k_i - i-я двоичная цифра ключа,

C_i - i-я двоичная цифра шифрованного текста.

$\oplus$ - операция XOR (исключающее "ИЛИ").

Таким образом, шифрованный текст генерируется путем побитового выполнения операции XOR для открытого текста и ключа. Благодаря свойствам этой операции для расшифровки достаточно выполнить подобную операцию:

$$p_i = C_i \oplus k_i.$$

Сутью этой технологии является способ выбора ключа. Вернам предложил использовать закольцованную ленту, что означает циклическое повторение ключевого слова, так что его система на самом деле предполагала работу хоть и с очень длинным, но все же повторяющимся ключом.

Несмотря на то, что такая схема в силу очень большой длины ключа значительно усложняет задачу криптоанализа, схему, тем не менее, можно взломать, имея в распоряжении достаточно длинный фрагмент шифрованного текста, известные или вероятно известные фрагменты открытого текста либо и то, и другое сразу.

11. Перестановочные шифры.

При шифровании с помощью перестановочных шифров элементы шифруемого текста переставляются по определенным правилам в пределах блока этого текста. В качестве ключа при этом могут использоваться:

- размер таблицы;
- слово или фраза, задающие перестановку:
- случайная последовательность натуральных чисел.

Одним из самых примитивных перестановочных табличных шифров является **простая перестановка**, для которой ключом служит размер таблицы. Исходное сообщение записывается в таблицу по столбцам, а затем считывается по строкам. При расшифровании действия выполняются в обратном порядке.

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый **одиночной перестановкой по ключу.** Он отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы. Если в качестве ключа используется слово, то каждая его буква заменяется числом, согласно порядку следования букв в алфавите. Если буквы повторяются, они

нумеруются слева направо. Например, ключевое слово «КОРОВА» заменяется на последовательность чисел 346521. После перестановки столбцов содержимое таблицы считывается по строкам.

Для обеспечения дополнительной стойкости можно повторно зашифровать сообщение, которое уже прошло шифрование. Такой метод шифрования называется **двойной перестановкой по ключу**. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы.

12. Шифр «Магический квадрат».

В средние века для шифрования перестановкой применялись магические квадраты.

Магическими квадратами называют квадратные таблицы с вписанными в их клетки последовательными натуральными числами, начиная от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число.

Примеры магических квадратов 4x4:

7	12	1	14
2	13	8	11
16	3	10	5
9	6	15	4

9	16	2	7
6	3	13	12
15	10	8	1
4	5	11	14

Шифруемый текст вписывали в магические квадраты в соответствии с нумерацией их клеток. Если затем выписать содержимое такой таблицы по строкам, то получится шифртекст, сформированный благодаря перестановке букв исходного сообщения. В те времена считалось, что созданные с помощью магических квадратов шифртексты охраняет не только ключ, но и магическая сила.

Пример магического квадрата и его заполнения сообщением: ПРИЛЕТАЮ ВОСЬМОГО

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

О	И	Р	М
Е	О	С	Ю
В	Т	А	Ь
Л	Г	О	П

Получаемый шифртекст: ОИРМ ЕОСЮ ВТАЬ ЛГОП

Задания на лабораторную работу

Разработать программные модули, реализующие алгоритм шифрования и алгоритм расшифрования в соответствии с вариантом задания, указанным преподавателем.

В программе задавать используемый алфавит.

При шифровании вводить с клавиатуры ключ и открытый текст, при расшифровании – ключ и шифртекст.

Варианты заданий:

1. Обобщенный алгоритм Цезаря.
2. Система Цезаря с ключевым словом.
3. Шифр, использующий линейное преобразование.
4. Шифр, использующий нелинейное преобразование.
5. Шифр, использующий сложение с ключевым словом.
6. Шифрующие таблицы Трисемуса.
7. Шифр Плейфейера.
8. Шифр «двойной квадрат» Уитстона.
9. Шифр Виженера.
10. Шифр Вернама (использовать для преобразования символов ASCII-таблицу).
11. Простая перестановка.
12. Одиночная перестановка по ключу.
13. Двойная перестановка по ключу.
14. Шифрование с использованием «магических квадратов».

Содержание отчета по лабораторной работе

1. Цель работы.
2. Структура алгоритмов шифрования и расшифрования.
3. Описание программы (входные и выходные значения, функции).
4. Текст программы.
5. Результаты работы программы.
6. Выводы.

Для защиты лабораторной работы необходимо подготовить отчет, продемонстрировать работу программы согласно варианту задания, устно ответить на контрольные вопросы.

Контрольные вопросы

1. Поясните понятия криптография, криптология, криптоанализ, стеганография.
2. Сформулируйте правило Керкхоффа.
3. Что такое шифрование, дешифрование, криптосистема, алфавит, открытый текст, пространство ключей?
4. Нарисуйте модель симметричной криптосистемы, предложенную К. Шенноном, и запишите формулы шифрования и расшифрования?

5. По каким признакам классифицируются криптосистемы?

6. Какие системы шифрования называют классическими? Используют ли их на практике? Что общего у классических шифров с современными симметричными криптосистемами?

7. Что такое криптостойкость? В каком случае схема шифрования называется абсолютно стойкой? Приведите пример абсолютно стойкого шифра.

8. Какие криптосистемы называют защищенными по вычислениям?

3.2 Лабораторная работа № 2

АЛГОРИТМ ШИФРОВАНИЯ ДАННЫХ DES

Цель работы

Познакомиться с основами симметричного шифрования. Изучить алгоритм шифрования DES на примере упрощенной версии S-DES.

Методические указания

1. Описание алгоритмов шифрования и расшифрования S-DES.

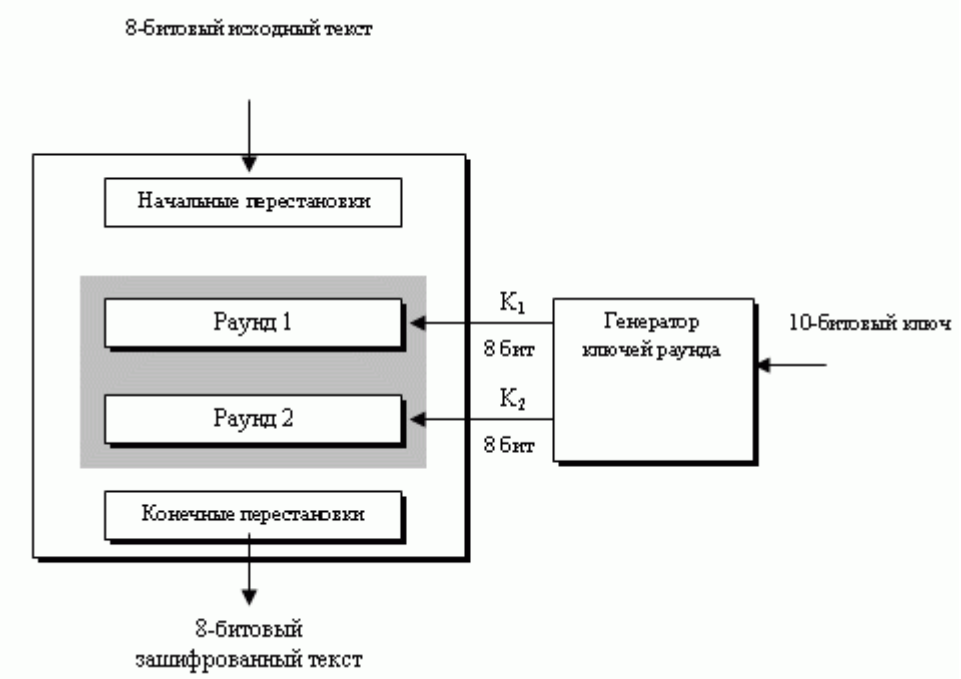
Среди современных методов традиционного шифрования долгое время самым распространённым являлся алгоритм DES (Data Encryption Standard). В 1977 году DES был утверждён и получил официальное имя: Federal Information Processing Standard 46 (FIPS PUB 46).

Алгоритм DES относится к группе симметричных алгоритмов, называемых сетями Файстеля.

Упрощенный S-DES – это алгоритм шифрования по структуре подобный DES, но имеющий меньше параметров. S-DES был разработан профессором Эдвардом Шейфером в учебных целях.

S-DES получает на входе 8-битовый блок открытого текста и 10-битовый ключ. В результате получается 8-битовый блок шифрованного текста. Используется 2 раунда шифрования.

Общая структура алгоритма шифрования S-DES представлена на рисунке.



Общая структура S-DES

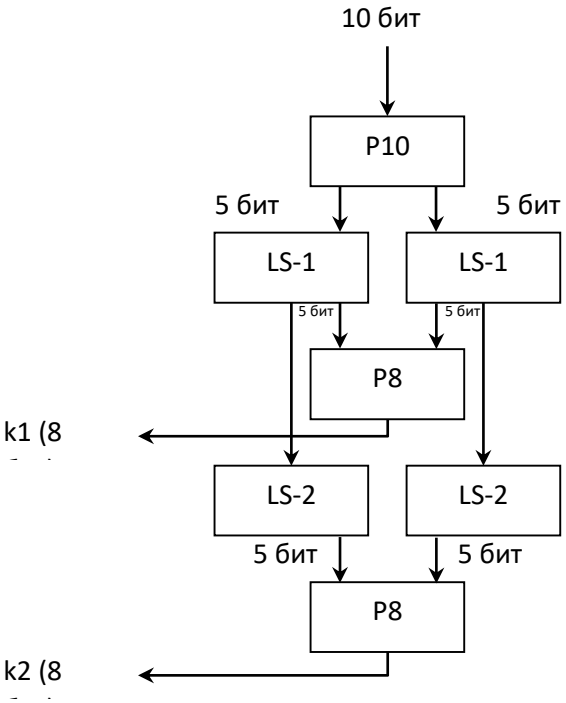
Алгоритм шифрования S-DES включает в себя последовательное выполнение 5-ти операций:

- начальная перестановка (IP);
- функция f_k — является композицией операций перестановки и подстановки, зависит от подключа раунда;
- перестановка SW ;
- f_k ;
- IP^{-1} - перестановка, обратная начальной.

Расшифрование производится по той же схеме, только подключи раундов подаются в обратном порядке.

2. Вычисление подключей S-DES.

В S-DES используется 10-битовый ключ, который должен быть как у отправителя, так и у получателя сообщения. Из этого ключа генерируются два 8-битовых подключа.



Вычисление подключей раундов

Пример. Пусть имеем на входе следующий ключ $K=642_{(10)}$:

$$K=\begin{Bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 & k_9 & k_{10} \end{Bmatrix}$$

1) Перестановка P10.

$$K=\begin{Bmatrix} 3 & 5 & 2 & 7 & 4 & 10 & 1 & 9 & 8 & 6 \\ k_3 & k_5 & k_2 & k_7 & k_4 & k_{10} & k_1 & k_9 & k_8 & k_6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{Bmatrix} \quad (P10)$$

2) Циклический сдвиг влево на одну позицию. Выполняется отдельно для первых 5-ти битов и отдельно для вторых 5-ти битов.

$$K=\begin{Bmatrix} 5 & 2 & 7 & 4 & 3 & 1 & 9 & 8 & 6 & 10 \\ k_5 & k_2 & k_7 & k_4 & k_3 & k_1 & k_9 & k_8 & k_6 & k_{10} \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{Bmatrix}$$

3) Перестановка P8.

$$K=\begin{Bmatrix} 6 & 3 & 7 & 4 & 8 & 5 & 10 & 9 & – & – \\ k_1 & k_7 & k_9 & k_4 & k_8 & k_3 & k_{10} & k_6 & – & – \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & – & – \end{Bmatrix} \quad (P8)$$

Получаем первый 8-битный подключ $k_1 = 10100100$.

4) Циклический сдвиг влево на две позиции.

$$K=\begin{Bmatrix} 7 & 4 & 3 & 5 & 2 & 8 & 6 & 10 & 1 & 9 \\ k_7 & k_4 & k_3 & k_5 & k_2 & k_8 & k_6 & k_{10} & k_1 & k_9 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{Bmatrix}$$

5) Перестановка P8.

$$K=\begin{Bmatrix} 6 & 3 & 7 & 4 & 8 & 5 & 10 & 9 & – & – \\ k_8 & k_3 & k_6 & k_5 & k_{10} & k_2 & k_9 & k_1 & – & – \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & – & – \end{Bmatrix} \quad (P8)$$

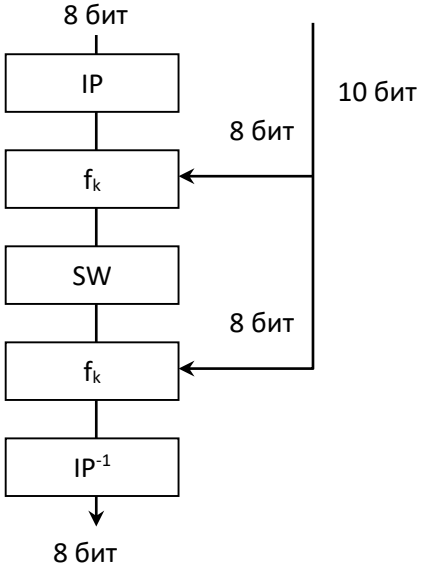
Получаем второй 8-битный подключ $k_2 = 01000011$.

Т.о. получено два подключа, в каждом из которых выделим правую и левую части:

$k_1 = 1010|0100$

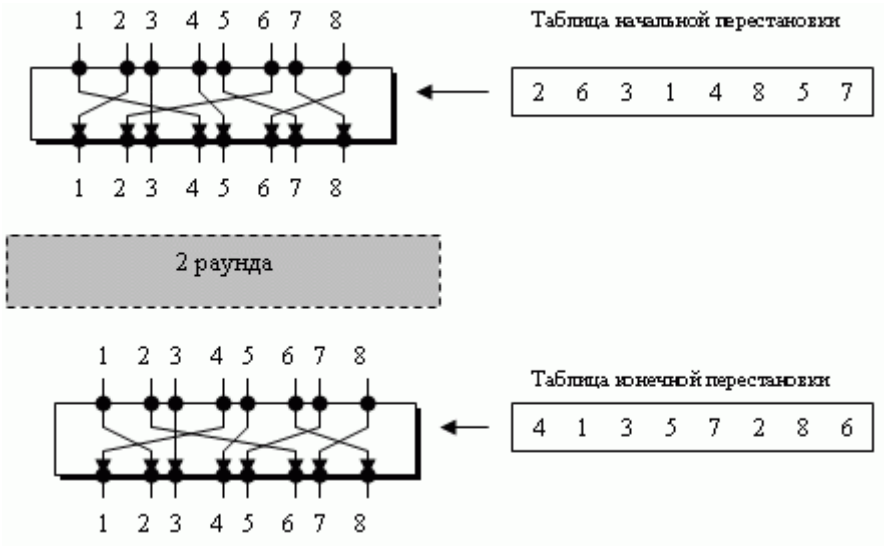
$k_2 = 0100|0011$

3. Алгоритм шифрования.



Алгоритм шифрования S-DES

Начальная и конечная перестановки IP и IP^{-1} представлены на рисунке ниже.



Начальная и конечная перестановки IP и IP^{-1}

Операция 1: Начальная перестановка $IP:\{2\ \ 6\ \ 3\ \ 1\ \ 4\ \ 8\ \ 5\ \ 7\}$.
Пусть шифруемый символ $"t"=116_{10}=01110100_2$, $L=0111, R=0100$), тогда
$IP=11101000, L=1110, R=1000$.

Операции 2: Функция f_k. представляет собой комбинацию перестановки и подстановки:

$$f_k.(L, R) = (L \oplus F(R, SK), R),$$

где L и R – левые и правые 4 бита 8-битовой последовательности, подаваемой на вход f_k.;
SK – подключ.

Отображение F.
На входе отображения имеем 4-битовое значение.
а) Сначала выполняется E/P — операция расширения/перестановки: $E/P=41232341$, применяется к правой части;
для примера имеем: $E/P\ (R)$ = 01000001;
$k_1=1010|0100$
б) $XOR(E/P, k_1)=11100101, L=1110, R=0101$;
в) применение S-матриц:

$$S_L=\begin{matrix}1&0&3&2\\3&2&1&0\\0&2&1&3\\3&1&3&1\end{matrix},\ S_R=\begin{matrix}1&1&2&3\\2&0&1&3\\3&0&1&0\\2&1&0&3\end{matrix}.$$

S-матрицы работают следующим образом: 1-ый и 4-ый биты входной последовательности рассматриваются как двухбитовые числа, определяющие строку S-матрицы, 2-ой и 3-ий биты — как числа, определяющие столбец S-матрицы. Элементы, находящиеся на пересечении строки и столбца, задают двухбитовые выходные значения:

L = 1|11|0 *oper* $S_L = S_L[\ 10_2, 11_2] = S_L[\ 2_{10}, 3_{10}] = 3_{10} = 11_2$;
R = 0|10|1 *oper* $S_R = S_R[\ 01_2, 10_2] = S_R[\ 1_{10}, 2_{10}] = 1_{10} = 01_2$;
Получаем 4-битовую последовательность: 1101;

г) Перестановка $P4 = 2431$ даёт на выходе: *P4*(1101) = 1101.
е) *XOR(L, P4)* = 1110 *XOR* 1101 = 0011.

Операция 3: *SW* – перестановка, меняет местами первые и последние 4 бита: *SW* (0011, *R*) = *R*|0011 = 1000|0011
.

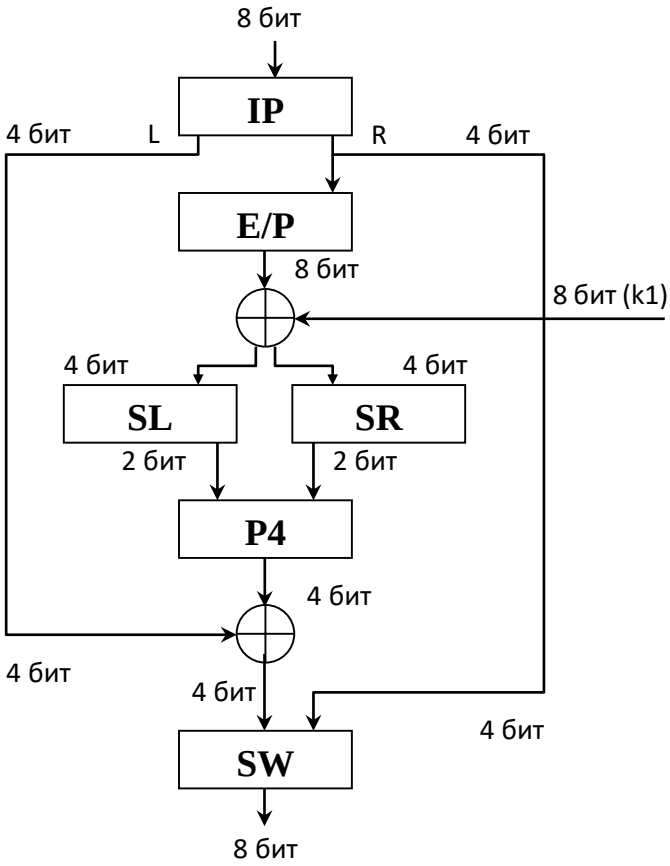


Схема f_k (операции 2, 4)

Операция 4: Функция f_k. К полученной последовательности битов применяем операцию 2, с той лишь разницей, что используется подключ k_2
.

Операция 5: Завершающая перестановка. Является обратной по отношению к начальной.

$$IP^{-1}:\{4\quad 1\quad 3\quad 5\quad 7\quad 2\quad 8\quad 6\}$$

На выходе получим 8-битовый блок, который затем преобразуем в символ, который и будет являться зашифрованным.

Для расшифрования используется тот же алгоритм, что и для шифрования, только в операции 2 используется подключ k_2, а в операции 4 — подключ k_1.

Задания на лабораторную работу

1. Изучить основы симметричного шифрования
2. Изучить алгоритмы шифрования DES и S-DES.
3. Написать программы шифрования и расшифрования одного символа с использованием алгоритма S-DES.

При шифровании/расшифровании ключ и символ вводить с клавиатуры в двоичном или десятичном виде (как значение ASCII-кода символа). Промежуточные значения выводить на экран в двоичном виде. Результат шифрования/расшифрования выводить на экран в двоичном или десятичном виде.

Содержание отчета по лабораторной работе

7. Цель работы.
8. Описание программы.
9. Текст программы.
10. Результаты работы программы.
11. Выводы.

Контрольные вопросы

1. Какие преобразования называются несингулярными? Приведите пример сингулярного и несингулярного преобразований.
2. Какова структура сети Файстеля?
3. Какая сеть Файстеля называется классической? Гомогенной?
4. Как выполняется расшифрование в сетях Файстеля?
5. От чего зависит криптоаналитическая стойкость шифра Файстеля?
6. Классифицируйте алгоритм DES. Является ли он сетью Файстеля? Почему?
7. Проведите сравнение алгоритмов DES и S-DES по основным параметрам (длина ключа, длина блока шифрования, количество раундов, количество подключей, размер и количество S-блоков).
8. Являются ли криптостойким алгоритм DES? Можно ли его применять на практике? Обоснуйте ответ.
9. Какие типы операций используются в современных блочных алгоритмах симметричного шифрования?

3.3 Лабораторная работа № 3

РЕЖИМЫ РАБОТЫ БЛОЧНЫХ ШИФРОВ

Цель работы

Изучить режимы работы блочных шифров и их применение.

Методические указания

1. Режимы работы блочных шифров

Режим шифрования - это алгоритм применения блочного шифра, который позволяет преобразовывать открытый текст произвольной длины в шифротекст, а затем выполнить обратное преобразование. Сам блочный шифр при этом является частью другого алгоритма – алгоритма режима шифрования. Это обусловлено тем, что блочный шифр работает только с отдельным *блоком* данных, в то время как алгоритм *режима шифрования* имеет дело уже с целым *сообщением*, которое может иметь произвольную длину и состоять из любого числа блоков.

Основные режимы шифрования:

1) ECB (Electronic Code Book) – электронная кодовая книга;
6) CBC (Cipher Block Chaining) – сцепление шифрованных блоков;
7) CFB (Cipher Feed Back) – шифрованная обратная связь;
8) OFB (Output Feed Back) – обратная связь по выходу алгоритма шифрования;
9) CTR (Counter) – шифрование со счётчиком.

Режимы работы блочных шифров были разработаны для стандарта DES.

В настоящее время применяются для любых симметричных блочных шифров.

Существуют международные стандарты ИСО/МЭК 10116:2006 «Информационные технологии. Методы обеспечения безопасности. Режимы работы для n-битовых блочных шифров» (ISO/IEC 10116:2006 Information technology - Security techniques - Modes of operation for an n-bit block cipher).

Режимы, в которых функция шифрования применяется в алгоритме режима после суммирования с блоком открытого текста называют блочными.

К блочным относятся режимы:

1) ECB;
2) CBC.

Режимы, в которых функция шифрования применяется до суммирования с блоком открытого текста называют поточными.

К поточным относятся режимы:

1) CFB;
2) OFB;
3) CTR.

Задания на лабораторную работу

Написать программу шифрования и расшифрования открытого текста, состоящего из произвольного количества символов, в одном из режимов согласно варианту:

1) электронной шифровальной книги;
2) сцепления шифрованных блоков;
3) шифрованной обратной связи;
4) обратной связи по выходу алгоритма шифрования;
5) шифрования со счетчиком.

Открытый текст и ключ вводит пользователь. Результат шифрования должен выводиться в десятичном или двоичном виде.

Привести схему и уравнения процессов шифрования и расшифрования для своего варианта задания.

Контрольные вопросы

1. Что такое режим шифрования?
2. Перечислите режимы работы блочных шифров.
3. Для чего были разработаны различные режимы работы блочных шифров?
4. Какие из режимов являются блочными, какие поточными?
5. В каких случаях применяется тот или иной режим работы? Какие режимы могут быть применены для получения криптографической контрольной суммы?
6. Охарактеризуйте преимущества и недостатки каждого из режимов работы блочных шифров.
7. В каких режимах шифрования используется вектор инициализации? Какие требования предъявляются к вектору инициализации в разных режимах? Должно ли его значение быть секретным?
8. Как ошибка в бите шифртекста или ошибка в векторе инициализации может повлиять на результат в различных режимах шифрования?

3.4 Лабораторная работа № 4

ЗАЩИТА ИНФОРМАЦИИ С ПОМОЩЬЮ СИММЕТРИЧНОЙ КРИПТОСИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ .NET

Цель работы

Изучить основные характеристики современных блочных алгоритмов симметричного шифрования. Изучить возможности реализации симметричной криптосистемы и настройки параметров шифра средствами .NET Framework.

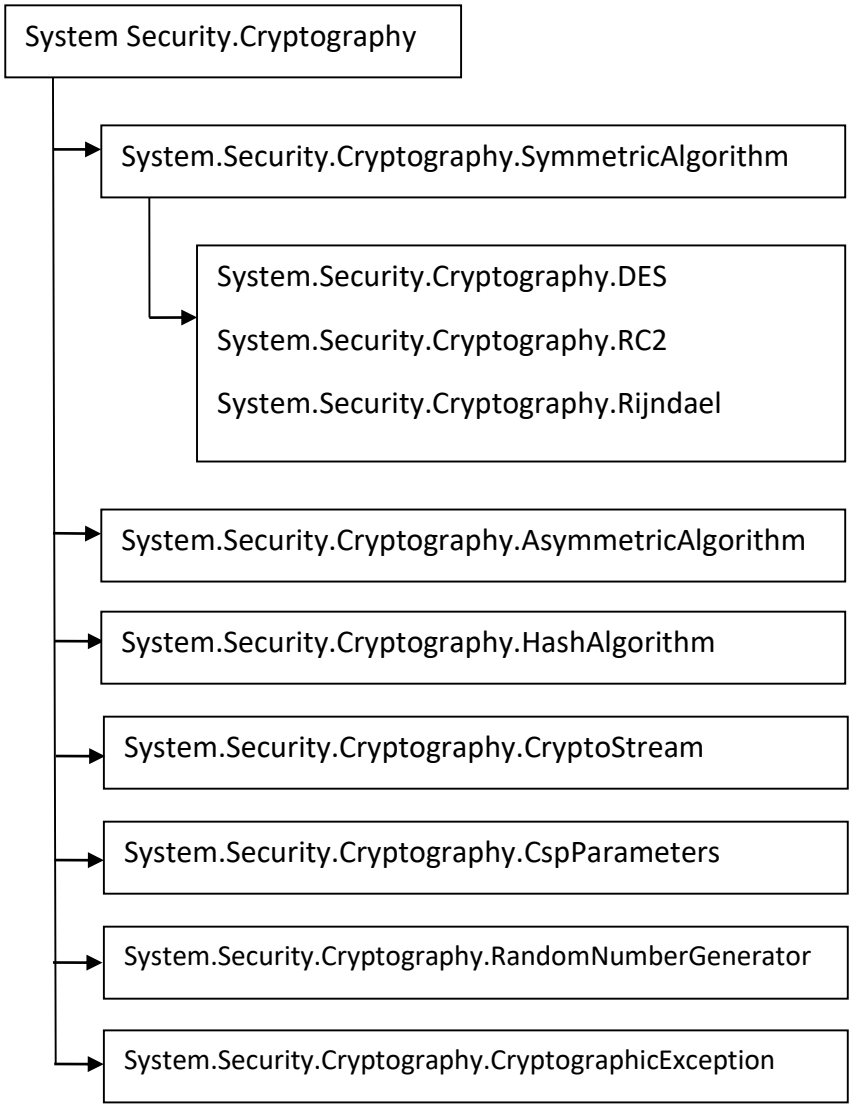
Методические указания

1. Модель криптографии .NET Framework

.NET Framework предоставляет реализации некоторых стандартных криптографических алгоритмов.

Пространство имен System.Security.Cryptography среды .NET Framework обеспечивает возможность реализации средств безопасности в приложении при помощи криптографических классов.

На схеме представлена иерархия криптографических классов пространства имен System.Security.Cryptography.



Классы SymmetricAlgorithm, AsymmetricAlgorithm, HashAlgorithm инкапсулируют соответственно симметричные алгоритмы, асимметричные алгоритмы и алгоритмы хеширования.

Класс CryptoStream предназначен для соединения потока данных с криптографическим алгоритмом.

Класс CspParameters инкапсулирует параметры, специфичные для алгоритма, которые можно сохранить или загрузить посредством поставщика услуг криптографии – CSP (Cryptographic Service Providers).

Класс RandomNumberGenerator – базовый класс для получения программных генераторов псевдослучайных чисел.

Класс CryptographicException инкапсулирует информацию об ошибках, возникающих при выполнении криптографических операций.

Система безопасности .NET Framework реализует **расширяемую модель наследования производных классов**. Иерархия имеет следующий вид.

- **Классы типа алгоритма**, например, SymmetricAlgorithm, AsymmetricAlgorithm или HashAlgorithm представляют собой абстрактные классы, создание экземпляров которых невозможно. Вместо этого работа осуществляется с производными классами, реализующими конкретные

публичные свойства, а также абстракные и виртуальные методы этих классов, каждый раз в зависимости от используемого алгоритма.

- **Классы алгоритмов** являются производными от классов типа алгоритма. Это абстрактные классы, инкапсулирующие соответствующие алгоритмы. Например, Aes, DES или ECDiffieHellman.

Из всех этих абстрактных классов далее наследуются конкретные классы, реализующие все поддерживаемые алгоритмы.

- **Классы реализации алгоритма** наследуются от класса алгоритма. Например, RC2CryptoServiceProvider – класс-оболочка, предоставляющий доступ к стандартной реализации алгоритма RC2 поставщиком услуг криптографии, которая является частью Microsoft Win32 CryptoAPI.

Используя данный шаблон производных классов, можно легко добавить новый алгоритм или новую реализацию существующего алгоритма. Например, для создания нового алгоритма шифрования с секретным ключом можно наследовать от класса SymmetricAlgorithm. Для создания новой реализации некоторого алгоритма можно создать неабстрактный класс, производный от класса, соответствующего этому алгоритму.

Часть криптографических классов представляют собой явные управляемые коды .NET. Имена управляемых провайдеров имеют суффикс Managed.

Другая часть классов является оболочками для Microsoft Win32 CryptoAPI. Имена всех неуправляемых провайдеров заканчиваются суффиксом CryptoServiceProvider.

Microsoft Cryptography API – это API компании Microsoft, предназначенный для доступа к криптографическим функциям, встроенным в платформу Windows. CryptoAPI предоставляет полный набор функций генерации ключей, шифрования/расшифрования, хеширования и цифровой подписи, функций для безопасного экспорта и совместного использования ключей, а также обеспечивает поддержку сертификатов и управление ими.

Например, класс Aes наследуется двумя классами: AesCryptoServiceProvider и AesManaged. Класс AesCryptoServiceProvider представляет собой оболочку для реализации алгоритма AES в CryptoAPI (CAPI), а класс AesManaged написан с использованием только управляемого кода.

Реализации в управляемом коде доступны на всех платформах, поддерживающих платформу .NET Framework. Реализации CAPI доступны в предыдущих версиях операционных систем и больше не развиваются. Однако реализации на основе управляемого кода не сертифицированы Федеральным стандартом обработки информации (FIPS) и могут работать медленнее реализаций на основе классов-оболочек.

Помимо реализации в управляемом коде и реализации в CAPI, существует также третий тип реализации, криптография следующего поколения (CNG) - это новейшая реализация, дальнейшая разработка которой продолжается. Примером алгоритма CNG является класс ECDiffieHellmanCng. Алгоритмы CNG доступны в Windows Vista и последующих версиях.

2. Симметричные алгоритмы в .NET.

В основе всех симметричных алгоритмов в .NET Framework лежит класс SymmetricAlgorithm. Все классы, реализующие симметричные алгоритмы, являются производными от него.

В .NET Framework поддерживаются следующие классы симметричных алгоритмов:

- DES;
- RC2;
- Rijndael;
- TripleDES;
- Aes.

Ниже перечислены классы реализаций симметричных алгоритмов (классы провайдеров), которые наследуются от класса алгоритма:

- AesManaged и AesCryptoServiceProvider (введены в .NET Framework 3.5)
- DESCryptoServiceProvider
- RC2CryptoServiceProvider
- RijndaelManaged
- TripleDESCryptoServiceProvider.

Класс SymmetricAlgorithm включает несколько защищенных полей, которые доступны через соответствующие открытые свойства во всех симметричных алгоритмах. Например, защищенное поле BlockSizeValue доступно через открытое свойство BlockSize. Таким образом, попытка присвоить свойству значение, которое запрещено для конкретного используемого асимметричного алгоритма, вызовет исключение CryptographicExeption. Защищенное поле и соответствующее ему публичное свойство относятся к одному типу данных, а их имена отличаются только суффиксом Value.

В таблице представлены некоторые из этих полей и свойств.

Член класса	**Описание**
Key и KeyValue	Возвращают или задают для симметричного алгоритма секретный ключ. Key – это открытое свойство, KeyValue – защищенное поле, значения обоих - массив байтов.
KeySize и KeySizeValue	Возвращают или задают размер секретного ключа в битах. KeySize – это открытое свойство, KeySizeValue – защищенное поле, оба свойства возвращают целое значение, представляющее длину ключа в битах.
LegalKeySizes и LegalKeySizesValue	Задают для данного симметричного алгоритма корректные размеры ключей в байтах. LegalKeySizes – это открытое свойство,

	LegalKeySizesValue – защищенное поле, оба свойства возвращают массив KeySizes.
IV и IVValue	Задают вектор инициализации для симметричного алгоритма. IV – это открытое свойство, IVValue - защищенное поле, оба свойства возвращают массив байтов.
BlockSize и BlockSizeValue	Задают размер блока в битах для текущего симметричного алгоритма. BlockSize - это открытое свойство, BlockSizeValue - защищенное поле, оба свойства возвращают целое число.
LegalBlockSizes и LegalBlockSizesValue	Задают допустимые размеры блока, поддерживаемые текущим симметричным алгоритмом. LegalBlockSizes - это открытое свойство, LegalBlockSizesValue - защищенное поле.
Mode и ModeValue	
Padding и PaddingValue	

IV и IVValue	Задают вектор инициализации для симметричного алгоритма. IV — это открытое свойство, а IVValue — защищенное поле, оба свойства возвращают массив байтов.
BlockSize и BlockSizeValue	Задают размер блока в битах для текущего симметричного алгоритма. BlockSize — это открытое свойство, а BlockSizeValue — защищенное поле, и оба свойства возвращают целое число.
LegalBlockSizes и LegalBlockSizesValue	Задают допустимые размеры блока, поддерживаемые текущим симметричным алгоритмом. LegalBlockSizes — это открытое свойство, а LegalBlockSizesValue — защищенное поле, и оба свойства возвращают массив KeySize.
Mode и ModeValue	Задают режим симметричных операций, используемых текущим алгоритмом. Mode — это открытое свойство, ModeValue — защищенное поле, и оба возвращают CipherMode.
Padding и PaddingValue	Задают режим заполнения, используемый текущим симметричным алгоритмом. Padding — это открытое свойство, PaddingValue — защищенное поле, и оба возвращают PaddingMode.
CreateEncryptor()	Метод CreateEncryptor() создает объект симметричного шифрования, используя указанные ключ и вектор инициализации. CreateEncryptor() — это открытый метод, возвращающий интерфейс ICryptoTransform.
CreateDecryptor()	Метод CreateDecryptor() создает объект симметрического дешифрования, используя ключ и вектор инициализации. CreateDecryptor() — открытый метод, возвращающий интерфейс ICryptoTransform.
GenerateKey()	Метод GenerateKey() генерирует для симметричного алгоритма случайный ключ и заменяет им значение, хранящееся в свойстве Key. GenerateKey() — открытый метод, возвращающий в массиве байтов случайный ключ.
GenerateIV()	Метод GenerateIV() генерирует для симметричного алгоритма случайный вектор инициализации и подменяет им значение, хранящееся в свойстве IV. GenerateIV() — открытый метод, возвращающий случайный вектор в массиве байтов.
ValidKeySize()	Показывает, допустим ли указанный размер ключа для текущего симметричного алгоритма. ValidKeySize() — открытый метод, возвращающий целое число.

3. Поточно-ориентированный подход.

В .NET используется поточно-ориентированный подход для реализации алгоритмов симметричного шифрования и хеширования. Основой такого подхода является класс CryptoStream, производный от класса Stream. Основанные на потоках криптографические объекты поддерживают единый стандартный интерфейс (CryptoStream) для управления своими частями, ответственными за передачу данных.

В зависимости от выбранного алгоритма в методах шифрования и расшифрования необходимо сделать следующие объявления, т.е. создать объект соответствующего типа провайдера (см. рисунок):

- **DES**

```
DESCryptoServiceProvider DESProvider = new DESCryptoServiceProvider();
```

- **RC2**

```
RC2CryptoServiceProvider RC2Provider = new RC2CryptoServiceProvider();
```

- **Triple-DES**

```
TripleDESCryptoServiceProvider tDESProvider = new
                                TripleDESCryptoServiceProvider();
```

- **Rijndael**

```
RijndaelManaged RijndaelProvider = new RijndaelManaged();
```

После этого назначить секретный ключ, введенный пользователем, свойству Key. Например, для алгоритма DES:

DESProvider.Key = ASCIIEncoding.ASCII.GetBytes(cryptoKey);

Затем с помощью метода GenerateIV() сгенерировать вектор инициализации. Например:

DESProvider. GenerateIV();

Далее создать объект шифрования для соответствующего алгоритма, вызывая метод CreateEncryptor(). Например, для DES:

ICryptoTransform DESEncrypt = DESProvider.CreateEncryptor();

Для задания режима заполнения последнего блока в свойстве Padding класса SymmetricAlgorithm могут использоваться следующие значения:

ANSIX923 - заполнение нулями, последний байт - числом дополненных байтов;

ISO10126 - заполнение случайными значениями, последний байт - числом дополненных байтов;

None - без заполнения;

PKCS7 - согласно указанному стандарту;

Zeros - нулями.

Задания на лабораторную работу

Разработать приложение, выполняющее шифрование и расшифрование данных с использованием симметричного алгоритма согласно варианту задания.

В программе задать режим шифрования, длину ключа, размер блока, используемый способ заполнения последнего блока открытого текста, при необходимости, исходные данные для режима шифрования (вектор инициализации, счетчики).

Открытый текст должен вводиться в окне диалога пользователем.

Вывести на экран:

1) Полученный шифртекст;

2) Используемые параметры алгоритма: режим шифрования; размер ключа; размер блока; способ заполнения последнего блока;

3) допустимые размеры ключей для данного алгоритма;

4) допустимые размеры блока шифрования для данного алгоритма.

Для алгоритмов DES и 3DES проверить ключи на слабость с помощью метода IsWeakKey.

Необходимые для работы алгоритма случайные числа генерировать методом RNGCryptoServiceProvider.

В вариантах с **нечетным** номером секретный ключ создавать методом GenerateKey. В вариантах с **четным** номером генерировать ключ на основе введенного пользователем пароля методом Rfc2898DeriveBytes.

Варианты заданий:

1. Алгоритм DES, режим работы ECB, метод заполнения последнего блока PKCS7
2. DES, CBC, ANSIX923
3. DES, CFB, ISO10126
4. 3DES, ECB, ANSIX923
5. 3DES, CBC, ISO10126
6. 3DES, CFB, PKCS7
7. RC2, ECB, ISO10126
8. RC2, CBC, PKCS7
9. RC2, CFB, ANSIX923
10. Rijndael, ECB, ANSIX923
11. Rijndael, CBC, PKCS7
12. Rijndael, CFB, ISO10126
13. Aes, ECB, PKCS7
14. Aes, CBC, ANSIX923
15. Aes, CFB, ISO10126
16. Aes, CTR, None

Контрольные вопросы

1. Перечислите основные характеристики (возможные размеры ключей, блоков, количество раундов) популярных алгоритмов симметричного шифрования: DES, RC, ГОСТ28147-89, ГОСТ Р 34.12-2015, AES, Rijndael.
2. Назовите государственный стандарт симметричного шифрования и контроля целостности Республики Беларусь. Приведите его основные характеристики.
3. Какие требования предъявляются к стандартному симметричному алгоритму шифрования?
4. Что такое многократное шифрование? Приведите пример.
5. Почему применение средств криптозащиты, реализующих стандартные (т.е. официально принятые в государстве) алгоритмы – наиболее распространенный способ защиты информации?

6. Перечислите криптографические классы высокого уровня, которые содержит пространство имен System.Security.Cryptography среды .NET Framework?

7. Какие симметричные алгоритмы поддерживаются в .NET? Какие из симметричных алгоритмов представляют собой явные управляемые коды .NET, а какие являются оболочками для Microsoft CryptoAPI?

8. Как можно получить корректные размеры ключей, поддерживаемые алгоритмом? Допустимые размеры блока, поддерживаемые текущим алгоритмом?

9. Перечислите способы заполнения последнего блока, предоставляемые .NET Framework. Как заполняется последний блок, если используется стандарт PKCS7?

10. Выберите из предложенных вариантов заданий те, которые могут быть использованы на практике для

а) обеспечения конфиденциальности передаваемой по сети информации произвольного размера;

б) обеспечения защиты секретных ключей;

в) обеспечения конфиденциальности поточных приложений;

г) получения криптографической контрольной суммы.

3.5 Лабораторная работа № 5

АСИММЕТРИЧНЫЕ АЛГОРИТМЫ В .NET

Цель работы

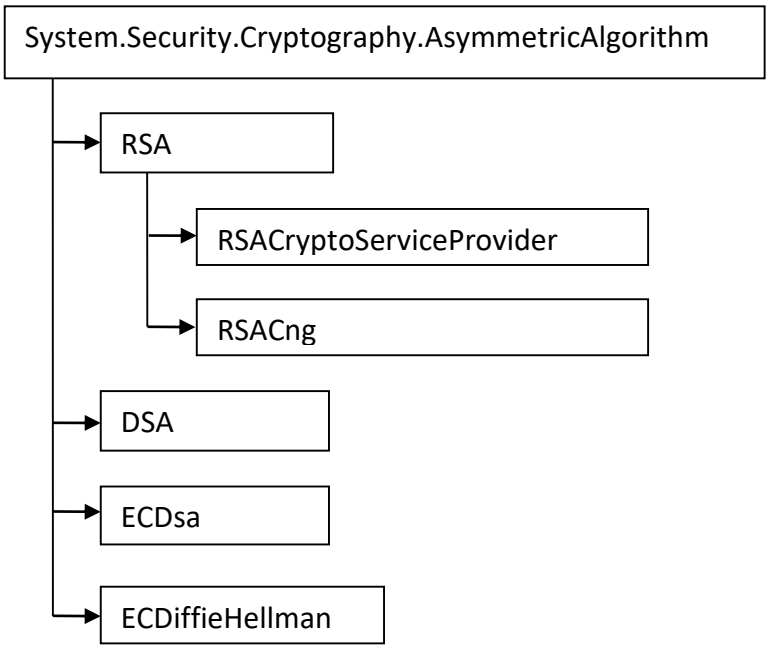
Изучить возможности реализации асимметричной криптосистемы средствами .NET

Методические указания

1. Класс AsymmetricAlgorithm.

Класс AsymmetricAlgorithm пространства имен System.Security.Cryptography среды .NET Framework представляет абстрактный базовый класс, от которого должны наследоваться все реализации алгоритмов асимметричного шифрования.

На схеме представлена иерархия наследования криптографических классов асимметричных алгоритмов шифрования.



Иерархия наследования криптографических классов асимметричных алгоритмов шифрования

Класс AsymmetricAlgorithm предоставляет несколько общих полей, свойств и методов, которые можно использовать во всех асимметричных алгоритмах.

Члены класса и их описания представлены в таблице.

Член класса (поле, свойство или метод	Описание
KeySize и KeySizeValue	Получает или задает размер модуля ключа, используемого алгоритмом асимметричного шифрования. KeySize – это открытое свойство, KeySizeValue – защищенное поле, оба свойства возвращают целое значение, представляющее длину ключа в битах.
LegalKeySizes и LegalKeySizesValue	Указывает допустимые размеры ключей для текущего асимметричного алгоритма. LegalKeySizes – это открытое свойство, LegalKeySizesValue – защищенное поле, оба свойства возвращают массив KeySize.
KeyExchangeAlgorithm	Свойство при переопределении в производном классе возвращает имя алгоритма обмена ключами. В противном случае создается исключение NotImplementedException
SignatureAlgorithm	Свойство при реализации в производном классе возвращает имя алгоритма подписи. В противном случае создается исключение NotImplementedException
FromXmlString(String)	Метод, если переопределен в производном классе, восстанавливает объект AsymmetricAlgorithm из XML-строки. В противном случае создается исключение NotImplementedException
ToXmlString(Boolean)	Метод, если переопределен в производном классе, создает и возвращает представление текущего объекта AsymmetricAlgorithm в виде XML-строки. В противном случае создается исключение NotImplementedException

Класс RSA является производным от класса типа алгоритма AsymmetricAlgorithm. Это абстрактный класс, от которого наследуются конкретные классы, реализующие алгоритм RSA. Класс RSA позволяет иметь несколько реализаций алгоритма RSA. В настоящее время класс RSA наследуется двумя классами: RSACryptoServiceProvider и RSACng.

Класс RSACryptoServiceProvider представляет собой оболочку для реализации алгоритма RSA в CryptoAPI, а класс RSACng написан с использованием криптографии следующего поколения (CNG) - это новейшая реализация, дальнейшая разработка которой продолжается. Алгоритмы CNG доступны в Windows Vista и последующих версиях.

2. Класс RSACryptoServiceProvider

Как было сказано выше, класс RSACryptoServiceProvider выполняет асимметричное шифрование и расшифрование с помощью реализации алгоритма RSA, предоставляемой криптопровайдером CSP (Cryptographic Service Provider). Этот класс не наследуется.

Конструкторы класса RSACryptoServiceProvider представлены в таблице:

Имя	**Описание конструктора**
RSACryptoServiceProvider()	Инициализирует новый экземпляр класса RSACryptoServiceProvider, используя ключ по умолчанию.
RSACryptoServiceProvider(CspParameters)	Инициализирует новый экземпляр класса RSACryptoServiceProvider с заданными параметрами.
RSACryptoServiceProvider(Int32)	Инициализирует новый экземпляр класса RSACryptoServiceProvider с указанным размером ключа.
RSACryptoServiceProvider(Int32, CspParameters)	Инициализирует новый экземпляр класса RSACryptoServiceProvider указанным размером ключа и параметрами.

Некоторые свойства класса RSACryptoServiceProvider представлены в таблице:

Имя свойства	**Описание свойства**
KeySize	Возвращает размер текущего ключа. (Наследуется от AsymmetricAlgorithm.KeySize.)
LegalKeySizes	Возвращает размеры ключа, которые поддерживаются алгоритмом асимметричного шифрования. (Наследуется от AsymmetricAlgorithm.)
PersistKeyInCsp	Возвращает или задает значение, указывающее, следует ли сохранить ключ поставщике служб шифрования CSP (т.е. в экземпляре класса CspParameters).
UseMachineKeyStore	Получает или задает значение, указывающее, следует ли сохранять ключ в

	хранилище ключей компьютера, а не в хранилище профилей пользователей.

Некоторые методы класса RSACryptoServiceProvider представлены в таблице:

Имя метода	Описание метода
Clear()	Освобождает все ресурсы, используемые классом AsymmetricAlgorithm. (Наследуется от AsymmetricAlgorithm.)
Dispose()	Освобождает все ресурсы, используемые текущим экземпляром класса AsymmetricAlgorithm. (Наследуется от AsymmetricAlgorithm.)
Encrypt(byte[] rgb, bool fOAEP)	Шифрует данные с помощью алгоритма RSA.
Decrypt(byte[] rgb, bool fOAEP)	Расшифровывает данные с помощью алгоритма RSA.
ToString()	Возвращает строковое представление текущего объекта. (Наследуется от Object.)
ExportParameters(Boolean)	Экспортирует параметры алгоритма RSA в структуру RSAParameters. (Переопределяет RSA.ExportParameters(Boolean).)
ImportParameters(RSAParameters)	Импортирует параметры алгоритма RSA из структуры RSAParameters в объект класса RSACryptoServiceProvider. (Переопределяет RSA.ImportParameters(RSAParameters).)
ToXmlString(Boolean)	Создает и возвращает строку XML, содержащую ключ текущего объекта RSA.(Наследуется от RSA.)
FromXmlString(String)	Инициализирует объект RSA, используя данные ключа из строки XML.(Наследуется от RSA.)

Основные методы класса RSACryptoServiceProvider – Encrypt и Decrypt.

Метод Encrypt – шифрует данные с помощью алгоритма RSA.

Синтаксис метода: ***public byte[] Encrypt(byte[] rgb, bool fOAEP)***

Первый параметр ***rgb*** - данные, предназначенные для шифрования (массив байт).

Максимально допустимая длина параметра ***rgb*** определяется длиной блока алгоритма RSA. Длина блока в свою очередь зависит от используемой версии ОС Microsoft Windows и метода заполнения блока. При установке пакета High Encryption – 16 байтов, иначе – 5 байтов. (Более подробную информацию можно получить из описания метода encrypt в документации MSDN).

Второй параметр ***fOAEP*** определяет метод дополнения блока для алгоритма RSA.

Дополнение блока необходимо потому, что алгоритм RSA требует фиксированного размера блока, а размер входных данных, в общем случае, может не соответствовать этому размеру.

Для параметра *fOAEP* рекомендуется использовать значение ***true***, при этом для дополнения блока будет применяться технология OAEP.

OAEP (Optimal Asymmetric Encryption Padding) – оптимальное дополнение для асимметричного шифрования. Это техника дополнения PKCS#1 v2, разработанная специально для алгоритма RSA, которая обеспечивает дополнение, гораздо более качественное с точки зрения безопасности, в сравнении с техникой PKCS#1 v1.5. Дополнение доступно на компьютерах под управлением ОС Microsoft Windows XP и всех последующих версий при установке соответствующего пакета шифрования.

Значение ***false*** - для дополнения блока будет использоваться метод PKCS#1 v1.5.

Возвращаемое значение: метод возвращает зашифрованный массив байтов.

Исключения:

CryptographicException – не удалось получить криптопровайдера (CSP), либо длина параметра ***rgb*** превышает максимально допустимую длину, либо параметр ***fOAEP*** имеет значение true, а заполнение OAEP не поддерживается.

ArgumentNullException – параметр ***rgb*** имеет значение ***null***.

Метод Decrypt - расшифровывает данные с помощью алгоритма RSA.

Синтаксис метода аналогичен синтаксису метода *Encrypt*:

public byte[] Decrypt(byte[] rgb, bool fOAEP)

Пример кода шифрования/расшифрования данных с использованием класса RSACryptoServiceProvider и методов Encrypt/Decrypt можно найти в MSDN.

Для задания нужного размера ключа можно использовать специальный конструктор класса RSACryptoServiceProvider.

3. Ключи асимметричного шифрования. Хранение ключей.

В асимметричных алгоритмах шифрования используются пары ключей. Если шифрование выполнялось одним ключом из пары, то расшифрование производится другим. Открытые (public) ключи могут передаваться другим

лицам для проверки цифровых подписей и шифрования пересылаемых данных.

Личные (private) ключи не могут быть экспортированы; они используются для создания цифровых подписей и расшифрования данных. Личный ключ должен быть известен только его владельцу.

За хранение и разрушение ключей отвечает криптопровайдер. Программист не имеет доступа непосредственно к двоичным данным ключа, за исключением операций экспорта открытых ключей.

Криптопровайдер поддерживает защищенные области, называемые **контейнерами ключей**. Контейнеры позволяют приложениям сохранять и использовать в дальнейшем сгенерированные один раз ключи, обеспечивая защиту самого ключа от злоумышленника.

Контейнеры бывают двух типов – **пользовательские** (этот тип используется по умолчанию) и **машинные** (CRYPT_MACHINE_KEYSET). Пользовательский контейнер доступен только приложениям, выполняемым от имени владельца контейнера. Приложение может использовать такой контейнер для сохранения персональных ключей данного пользователя. Доступ к машинным контейнерам разрешен только администраторам. В них обычно сохраняются ключи, используемые сервисами и системными программами.

Свойство UseMachineKeyStore класса RSACryptoServiceProvider получает или задает значение, указывающее, следует ли сохранять ключи в контейнере ключей компьютера, либо в контейнере профилей пользователей. По умолчанию ключи сохраняются в пользовательском контейнере.

В модели криптографии .NET Framework ключи для шифрования/расшифрования и создания/проверки подписей разделены. Называются они соответственно «пара для обмена ключами» и «пара для подписи».

Т. о. база данных ключей состоит из контейнеров, в каждом из которых хранятся ключи, принадлежащие определенному пользователю. Контейнер ключей имеет уникальное имя. В контейнере может существовать не более одной пары ключей подписи, одной пары ключей обмена и одного симметричного ключа. Если поддерживается несколько алгоритмов симметричного шифрования, то симметричных ключей может быть несколько, по одному ключу каждого алгоритма. Все ключи хранятся в защищенном виде.

Пары ключей и симметричные ключи могут находиться только в контейнере. Только открытый ключ пары может находиться вне контейнера.

По умолчанию для каждого пользователя создается контейнер с именем этого пользователя. Можно создавать дополнительные контейнеры и назначать им произвольные имена, которые обязательно должны быть уникальными.

База данных криптографических ключей

Всякий раз, когда создается новый, используемый по умолчанию экземпляр класса RSACryptoServiceProvider, автоматически создается готовая к использованию новая пара открытого и личного ключей.

Допустимые размеры ключей в алгоритме RSA зависят от поставщика криптографических услуг (CSP), который используется экземпляром RSACryptoServiceProvider. Windows CSP разрешают размеры ключей от 384 до 16384 бит с шагом 8 бит для версий Windows до Windows 8.1 и размеры ключей от 512 до 16384 бит для Windows 8.1. Длина ключа по умолчанию при этом 1024 бита.

4. Сохранение ключей в контейнере ключей.

Следующий пример кода показывает, как создать новую пару ключей шифрования/расшифрования (сгенерировать новые параметры алгоритма RSA) и сохранить ключи в пользовательском контейнере ключей с заданным именем.

Пример взят из MSDN:

```
using System;
using System.Security.Cryptography;

class RSACSPSample
{
   static void Main()
   {
      string KeyContainerName = "MyKeyContainer";
      RSAPersistKeyInCSP(KeyContainerName);
      RSADeleteKeyInCSP(KeyContainerName);
   }
   public static void RSAPersistKeyInCSP(string ContainerName)
   {
      try
      {
         CspParameters cspParams = new CspParameters();
         cspParams.KeyContainerName = ContainerName;
         RSACryptoServiceProvider RSAalg = new
RSACryptoServiceProvider(cspParams);
```

```
            Console.WriteLine("The RSA key was persisted in the container, \"{0}\".", ContainerName);
            }
            catch(CryptographicException e)
            {
               Console.WriteLine(e.Message);
            }
         }
         public static void RSADeleteKeyInCSP(string ContainerName)
         {
            try
            {
               CspParameters cspParams = new CspParameters();
               cspParams.KeyContainerName = ContainerName;
               RSACryptoServiceProvider RSAalg = new RSACryptoServiceProvider(cspParams);
               RSAalg.PersistKeyInCsp = false;
               RSAalg.Clear();
               Console.WriteLine("The RSA key was deleted from the container, \"{0}\".", ContainerName);
            }
            catch(CryptographicException e)
            {
               Console.WriteLine(e.Message);
            }
         }
      }
```

Если надо повторно использовать созданные ранее и сохраненные в пользовательском контейнере ключи, этого можно достичь, проинициализировав класс RSACryptoServiceProvider заполненным объектом CspParameters, используя соответствующий конструктор.

RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(cspParams);

5. Использования хранилища ключей компьютера.

Для использования хранилища ключей компьютера вместо хранилища ключей профилей пользователей нужно задать свойство UseMachineKeyStore класса RSACryptoServiceProvider:

RSACryptoServiceProvider. UseMachineKeyStore = true;

Все экземпляры класса RSACryptoServiceProvider, созданные с помощью конструктора по умолчанию (т.е. не инициализированные объектом CspParameters) будут использовать этот параметр.

Можно установить флаг использования хранилища ключей компьютера непосредственно для объекта класса CspParameters.

Операции криптопровайдера базируются на значении перечисления CspProviderFlags. В этом перечислении поддерживаются два значения: UseDefaultKeyContainer и UseMachineKeyStore. Если задан флаг UseDefaultKeyContainer, то информация о ключах будет считываться из контейнера ключей по умолчанию (т.е. из пользовательского контейнера). Если указан флаг UseMachineKeyStore, то информация о ключах будет считываться из контейнера ключей компьютера.

В следующем примере кода в начале создается экземпляр класса CspParameters, затем устанавливается флаг UseMachineKeyStore и задается имя контейнера ключей.

```
CspParameters cspParam = new CspParameters();
cspParam.Flags = CspProviderFlags. UseMachineKeyStore;
cspParam.KeyContainerName = “MachineKeyStore”;
```

6. Сохранение ключей в XML-формате.

Если надо повторно использовать созданные ранее ключи либо передать открытый ключ между двумя приложениями, использующими разные платформы или разные криптографические библиотеки, можно сохранить ключевую информацию в формате XML.

Рассмотрим фрагмент кода, в котором ключи шифрования/расшифрования сохраняются в двух XML-файлах. В файле PublicPrivateKey.xml сохраняется пара ключей: открытый и личный. В файле PublicOnlyKey.xml – только открытый ключ. Для сохранения ключей используется метод ToXmlString.

Сначала создаем экземпляр RSACryptoServiceProvider с ключами по умолчанию.

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
```

Затем сохраняем ключи в соответствующих файлах.

```
// Пару ключей
StreamWriter writer = new StreamWriter(“PublicPrivateKey.xml”);
string publicPrivateKeyXML = rsa. ToXmlString(true);
writer.Write(publicPrivateKeyXML);
writer.Close();
// Только открытый ключ
writer = new StreamWriter(“PublicOnlyKey.xml”);
string publicOnlyKeyXML = rsa. ToXmlString(false);
writer.Write(publicOnlyKeyXML);
writer.Close();
```

Теперь рассмотрим, как можно загрузить ключи из XML-файла с помощью метода FromXmlString().

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
StreamReader reader = new StreamReader(“PublicPrivateKey.xml”);
```

```
string publicPrivateKeyXML = reader.ReadToEnd();
rsa. FromXmlString(publicPrivateKeyXML);
reader.Close();
```

После извлечения ключей можно выполнять процедуру расшифрования данных.

Задания на лабораторную работу

Разработать программу, демонстрирующую применение алгоритма RSA для шифрования/расшифрования данных. Использовать класс RSACryptoServiceProvider.

Программа должна обеспечивать следующие функции:

1) Генерацию пары ключей шифрования/расшифрования и сохранение их в машинном контейнере ключей, пользовательском контейнере ключей с заданным именем, либо в XML-файлах в соответствии с вариантом задания.
2) Шифрование текстовой строки, введенной пользователем. Ключ шифрования должен извлекаться из контейнера ключей или XML-файла.

При шифровании отображать на экране:

а) используемые открытые параметры алгоритма RSA (модуль и показатель степени);

б) текущий размер ключа.

3) Расшифрование заданного шифртекста. Ключ расшифрования должен извлекаться из контейнера ключей или XML-файла. При расшифровании отображать на экране используемые параметры алгоритма RSA (P, Q, N, e, d).
4) Вывод информации о допустимых размерах ключей для используемой реализации алгоритма RSA.

Варианты заданий:

Размер ключей в битах / Хранение ключей	1024	2048	4096
в XML-файлах	1	2	3
в машинном контейнере ключей	4	5	6
в пользовательском контейнере ключей	7	8	9

Контрольные вопросы

1. Какие асимметричные алгоритмы поддерживает пространство имен System.Security.Cryptography?
2. Какова иерархия наследования криптографических классов асимметричных алгоритмов шифрования в .NET Framework?
3. Что такое CSP?
4. От чего зависят допустимые размеры ключей в алгоритме RSA? Каковы их значения?
5. Каков используемый по умолчанию размер ключей в алгоритме RSA? Можно ли задать размер ключей отличный от используемого по умолчанию? Как это сделать?
6. Как можно сохранить и использовать в дальнейшем сгенерированные один раз ключи для алгоритма RSA?
7. Что такое контейнер ключей? Какие бывают типы контейнеров ключей, для чего они используются, кто имеет к ним доступ? Какой тип контейнера ключей используется по умолчанию?

3.6 Лабораторная работа № 6

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Цель работы

Изучить применение электронной цифровой подписи, основанной на криптографическом хэше и асимметричном алгоритме для решения задач аутентификации, контроля целостности и подтверждения обязательств с использованием средств .NETSecurityFramework.

Методические указания

1. Функции хеширования.

Хеш – это функция, которая ставит в соответствие небольшой, фиксированного размера объем двоичных данных произвольному, сколь угодно большому объему входных данных.

$$H = H(M),$$

где M – прообраз - сообщение произвольной длины,

H(M) – значение функции хеширования фиксированной длины.

Значение функции хеширования называют также ***хеш-код, свертка, функция сжатия, профиль сообщения, дайджест сообщения, отпечаток пальца***.

Значение хэш-кода присоединяется к сообщению отправителем. Получатель устанавливает аутентичность сообщения путем повторного вычисления значения функции хеширования.Т.о. целью использования функции хеширования является получение «дактилоскопической» характеристики файла или любого блока данных.

Однако в общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше чем вариантов входного массива; существует множество массивов, дающих одинаковые хеш-коды — так называемые **коллизии.** Вероятность возникновения коллизий играет важную роль в оценке качества хеш-функций.

Существует множество алгоритмов хеширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкостьи т.п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи.

Простые хеш-функции широко используются в программировании, не связанном с криптографией, например, для обнаружения ошибок или для быстрого поиска объектов по хеш-таблице. Виртуальный метод

GetHashCodeобъекта Object – это пример простой хеш-функции, создающей 32-битовый хеш.

Криптографические хеш-функции должны обладать дополнительными свойствами, обеспечивающими их криптографическую стойкость и позволяющими применять их в криптографических приложениях.

Идеальная криптографическая хеш-функция должна обладать следующими свойствами:

1. Быть применимой к блоку данных любой длины.
2. Давать на выходе значение фиксированной длины.
3. Значение H(x) должно вычисляться относительно легко для любого заданного x, а алгоритм вычисления должен быть практичным с точки зрения аппаратной и программной реализации.
4. Изменение в прообразе даже одного бита должно приводить к изменению приблизительно половины битов в значении хеш-функции (диффузия или лавинный эффект).
5. Свойство однонаправленности.Однонаправленной хеш-функцией называют хеш-функцию для которой вычислить значение хеш-функции по прообразу несложно, а сгенерировать прообраз, который свернется к данной величине, очень трудно.
6. Для любого данного блока x должно быть практически невозможным вычислить y≠x, для которого H(x)=H(y). Такое свойство называется слабой сопротивляемостью коллизиям или стойкость к коллизиям первого рода.
7. Должно быть практически невозможно подобрать пару сообщений (M, M’), имеющих одинаковый хеш-код. Это свойство называется сильной сопротивляемости коллизиям или стойкость кколлизиям второго рода.

2. Классы, реализующие хеш-алгоритмы, поддерживаемые в .NET Framework

Перечисленные ниже классы, реализующие известные алгоритмы хеширования, являются производными абстрактного класса HashAlgorithm пространства имен System.Security.Cryptography. В таблице слева приводится имя абстрактного класса, а справа – имя производного конкретного класса.

Абстрактный класс	Производный класс
MD5	MD5CryptoServiceProvider
SHA1	SHA1Managed и SHA1CryptoServiceProvider
SHA256	SHA256Managed
SHA384	SHA384Managed
SHA512	SHA512Managed

Абстрактные классы невозможно использовать напрямую, создавая на их основе экземпляры объектов. Из каждого такого класса производятся

конкретные классы реализации. Классы, имена которых заканчиваются на CryptoServiceProvider, реализованы с использованием интерфейса CryptoAPI, предоставляемого операционной системой. Классы, имена которых заканчиваются на Managed, реализованы полностью средствами контролируемого C#-кода, без использования CryptoAPI.

Класс HashAlgorithm обладает публичным свойством Hash, которое представляет собой байтовый массив, содержащий вычисленный хеш-код.

Публичное свойство HashSizeсодержит значение размера хеш-кода в битах.

Самый важный публичный метод класса HashAlgorithm – метод ComputeHash. Он возвращает значение хеш-кода в виде массива байтов, вычисленное для заданных во входном параметре входных данных.Входные данные также представляются в виде массива байтов.

В следующем примере иллюстрируется использование класса HashAlgorithmна примере конкретного производного класса, инкапсулирующего алгоритм SHA-1. Предполагается, что входные данные заданы в байтовом массиве messageByteArray.Значение хеш-кода возвращается в массив sha1Hash.

```
HashAlgorithmsha1 = newSHA1CryptoServiceProvider();
byte[] sha1Hash = sha1.ComputeHash(messageByteArray);
```

3. Идентификаторы объектов

Международный стандарт ASN.1 OID (ObjectIdentifiers) предназначен для формирования уникальных идентификаторов для компьютерных форматов, логически организованных в иерархию имен. Он поддерживается многими организациями, включая ANSI. Существует большое число идентификаторов OID, идентифицирующих конкретные протоколы, алгоритмы и форматы данных. В частности, уникальные идентификаторы OIDимеют многие криптографические алгоритмы, признанные ANSI.

Некоторые значения идентификаторов приведены в таблице.

Криптографический хеш-алгоритм	**OID**
MD5	1.2.840.113549.2.5
SHA-1	1.3.14.3.2.26
SHA-2	562.16.840.1.101.3.4.2.1
SHA-3	842.16.840.1.101.3.4.2.2
SHA-5	122.16.840.1.101.3.4.2.3

Идентификаторы OID необходимо использовать в определенных методах классов .NETSecurityFramework.

Например, в методах SignHash и VerifyHashклассов RSACryptoServiceProviderи DSACryptoServiceProvider.

В приведенном фрагменте кода показано, как идентификатор OID алгоритма хеширования используется в качестве параметра метода SingHashкласса RSACryptoServiceProvider.

RSACryptoServiceProviderrsa = new RSACryptoServiceProvider();

Signaturebytes = rsa.SignHash(hashbytes, “1.3.14.3.2.26”);

Сначала создается объект RSAс ключом по умолчанию. Затем с помощью методаSingHashхеш-код подписывается (шифруется)личным ключом. Параметрами метода являются: байтовый массив, содержащий сам хеш-код и идентификатор OID алгоритма хеширования SHA-1. Предполагается, что хеш-код, переменная hashbytes,уже создан вызовом метода ComputeHashкласса SHA1.

4. Электронная цифровая подпись

Аутентификация сообщений защищает две обменивающиеся сообщениями стороны от любой третьей, но не обеспечивает защиту каждой из сторон от другой. Поэтому в ситуациях, когда нет полного доверия между отправителем и получателем, требуется нечто большее, чем аутентификация. Наиболее привлекательное решение проблемы – использование электронной цифровой подписи (ЭЦП).

Цифровая подпись обеспечивает целый комплекс защиты, который было бы сложно осуществить любым другим способом. Она аналогична подписи, сделанной от руки.

ЭЦП должна обладать следующими свойствами:

1. Достоверность (ЭЦП должна давать возможность установить автора, дату и время подписи, а также достоверность содержимого сообщения на время подписи).
2. Неподдельность (Никто не может выдать себя за автора. ЭЦП доказывает, что именно подписавший и никто другой сознательно подписал документ).
3. Невозможность использовать подпись повторно (т.к. она – часть документа, два сообщения – две разных подписи).
4. Невозможность изменить подписанный документ (изменение документа – изменение подписи).
5. Невозможность отрицания авторства.

Из свойств следуют требования к ЭЦП:

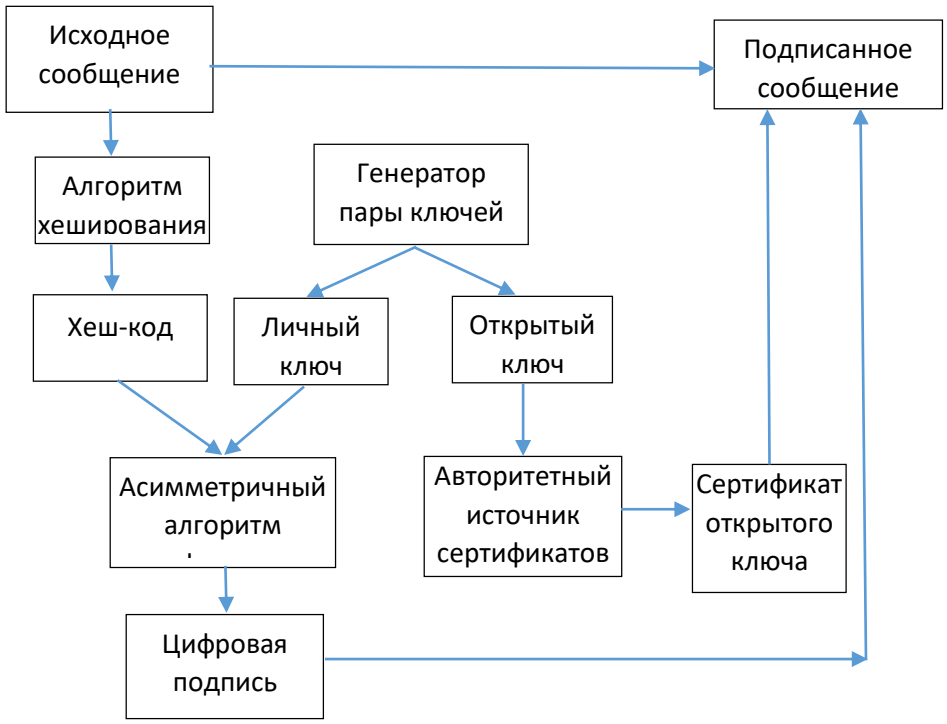
1. ЭЦП должна быть функцией подписываемого сообщения.
2. ЭЦП должна использовать информацию, уникальную для отправителя, чтобы предотвратить возможность фальсификации и отрицания авторства.
3. С точки зрения вычислений должно быть нереально фальсифицировать цифровую подпись (ни с помощью создания нового сообщения для имеющейся ЭЦП, ни с помощью создания фальшивой ЭЦП для имеющегося сообщения).
4. ЭЦП должна легко генерироваться.
5. ЭЦП должно быть легко распознать и проверить.
6. ЭЦП должно быть удобно хранить в запоминающих устройствах.

Всем этим требованиям удовлетворяет защищенная функция хеширования.

5. Создание и верификация цифровой подписи

Технология применения электронной цифровой подписипредполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: открытый и личный. Личный ключ хранится абонентом в тайне и используется для формирования ЭЦП.Открытый ключ известен всем другим пользователям и предназначен для проверки ЭЦП получателем документа.

Общая схема применения электронной цифровой подписи изображена на рисунке.



Берется исходное сообщение и создается его хеш-код при помощи одного из алгоритмов хеширования. Затем хеш-код шифруется при помощи личного ключа отправителя сообщения. Результат шифрования и называют электронной цифровой подписью

Никто, кроме владельца личного ключа, не сможет создать такую подпись, даже располагая оригиналом исходного сообщения. Никто не сможет изменить сообщение или создать поддельное сообщение так, чтобы обман не раскрылся.

Подписанное сообщение формируется объединением исходного сообщения, его цифровой подписи и сертификата открытого ключа, соответствующего тому личному ключу, которым шифровался хеш.

При получении подписанное сообщение верифицируется. Получатель хочет убедиться в том, что сообщение действительно отправлено

отправителем, а не кем-либо еще. Также получатель хочет убедиться, что сообщение не было изменено в пути следования.

Полученное сообщение разбивается на три компоненты – исходное сообщение, открытый ключ отправителя, цифровая подпись.

Получатель извлекает сертификат открытого ключа отправителя и проверяет его с помощью центра выдачи сертификатов. С помощью открытого ключа, содержащегося в сертификате, получатель может расшифровать хеш-код. После этого заново вычисляется хеш-код сообщения и сравнивается с полученным значением. Если хеши совпали, сообщение аутентично и не изменено.

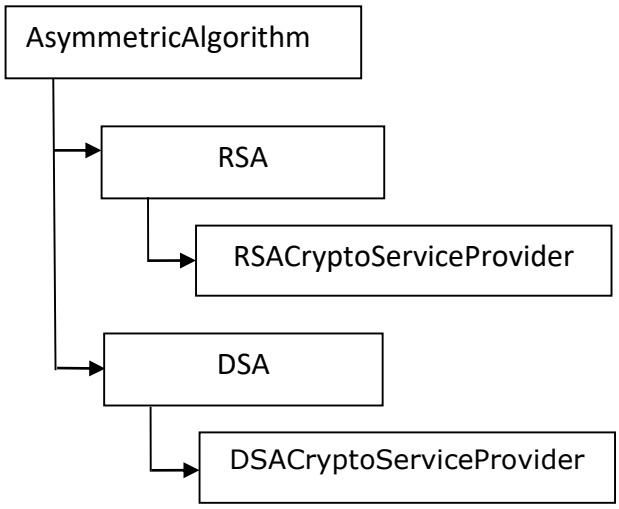
Схема верификации ЭЦП приведена на рисунке.

6. Класс AsymmetricAlgorithm

В библиотеке классов .NETSecurityподдерживаются асимметричные алгоритмы DSAи RSA. В основе всех асимметричных алгоритмов лежит класс AsymmetricAlgorithm.

Класс AsymmetricAlgorithmрасполагается в пространстве имен SystemSecurity.Cryptography и является абстрактным классом. Из него производятся классы алгоритмов: RSAи DSA, которые также являются абстрактными. Из классов RSAи DSA затем производятся классы RSACryptoServiceProvideri DSACryptoServiceProvider, которые обеспечивают реализацию алгоритмов. Эти классы являются оболочками для MicrosoftCrypto API.

Иерархия класса асимметричного алгоритма приведена на рисунке.



Способы работы с цифровой подписью с помощью классов DSACryptoServiceProvider и RSACryptoServiceProvider практически идентичны. Публичные методы и свойства классов DSACryptoServiceProvider и RSACryptoServiceProvidertакже во многом аналогичны.Конструктор класса автоматически генерирует ключевую информацию в момент создания экземпляра.

Некоторые публичные свойства классовпредставлены в таблице:

KeyExchangeAlgorithm	Свойство получает имя алгоритма обмена ключами
KeySize	Свойство получаетразмер ключа в битах
LegalKeySizes	Разрешенные размеры ключей
SignatureAlgorithm	Получает имя алгоритма ЭЦП

Основные публичные методы классов представлены в таблице:

CreateSignature	Создает цифровую подпись DSAдля заданного сообщения
VerifySignature	Верифицирует заданную подпись DSAдля заданного сообщения
SignData	Вычисляет хеш сообщения и подписывает его
VerifyData	Верифицирует заданную подпись, сравнивая ее с подписью, вычисленной для заданного сообщения
SignHash	Вычисляет подпись для заданного значения хеша
VerifyHash	Верифицирует заданную подпись, сравнивая ее с подписью, вычисленной для заданного хеша
ToXmlString	Создает и возвращает XML-представление текущего объекта
FromXmlString	Создает объект из XML-данных
ExportParameters	Экспортирует параметры DSAв объект DSAParameters

ImportParameters	Импортирует параметры DSAиз объектаDSAParameters

В функциях, обеспечиваемых этими методами, наблюдается некоторая избыточность, а это означает, что одну и ту же задачу можно решить разными способами.

7. Пример программы с использованием подписи RSA и алгоритма хеширования SHA-1

Создание и верификацию ЭЦП для заданного пользователем сообщения реализуем в виде отдельных методов CreateSignature и VerifySignature.

Метод CreateSignature накладывает подпись на сообщение и состоит из следующих шагов.

Шаг 1. Преобразоватьвведенное пользователем текстовое сообщение в массив байтов.

byte[] messagebytes = Encoding.UTF8.GetBytes(text);

Шаг 2. СоздатьобъектклассаSHA1CryptoServiceProvider.

SHA1sha1 = newSHA1CryptoServiceProvider();

Шаг 3. Сгенерировать хеш-код исходного сообщения.

byte[] hashbytes = sha1.ComputeHash(messagebytes);

Шаг 4. СоздатьобъектклассаRSACryptoServiceProviderс ключом по умолчанию.

RSACryptoServiceProviderrsa = newRSACryptoServiceProvider();

Шаг 5. Зашифровать хеш-кодличным ключом отправителя.

signaturebytes = rsa.SignHash(hashbytes, “1.3.14.3.2.26”);

Параметрами метода SignHashявляются: байтовый массив, содержащий сам хеш-код и идентификатор OID алгоритма хеширования SHA-1.

Шаг 6.Экспортироватьпараметры RSA при помощи метода ExportParametersс параметром false (извлекает открытый ключ, необходимый для верификации):

rsaparams = rsa.ExportParameters(false);

Метод VerifySignatur еверифицирует подпись и состоит из следующих шагов.

Шаг 1 – Шаг 4 идентичны шагам метода CreateSignature.

Шаг 5. Вызвать метод ImportParameters с использованием того объекта rsaparams, что был создан ранее подписывающим методом. В этом случае параметры RSAбудут идентичны параметрам, использованным при создании подписи.

rsa.ImportParameters(rsaparams);

Шаг 6. Верифицировать подпись с помощью метода VerifyHashкласса RSACryptoServiceProvider:

bool match = rsa.VerifyHash(hashbytes, “1.3.14.3.2.26”, signaturebytes);

Для передачи информации между методами создания и верификации подписи используются три параметра:

1) text–строка, содержащая исходный текст сообщения.

2) RSAParametersrsaparams - объект, инкапсулирующий информацию открытого ключа.

3) byte[] signaturebytes – массив, содержащий цифровую подпись сообщения.

Задания на лабораторную работу

Написать программу создания и верификации электронной цифровой подписи на основе указанных алгоритмов.

Вывести на экран значение хеш-кода в шестнадцатиричном формате, его размер, значение ЭЦП, параметры используемого асимметричного алгоритма (ключи, их размер и т.д.).

№ варианта	Асимметричный алгоритм	Хеш-алгоритм
1	RSA	MD5
2	RSA	SHA1
3	RSA	SHA256
4	RSA	SHA384
5	RSA	SHA512
6	DSA	MD5
7	DSA	SHA1
8	DSA	SHA256
9	DSA	SHA384
10	DSA	SHA512

Контрольные вопросы

1. Что такое хеш-код? Какие еще термины используются для обозначения данного понятия?
2. Какими свойствами должна обладать функция хеширования, чтобы ее можно было использовать в криптографии?
3. Какие современные алгоритмы хеширования вы знаете? Какие размеры дайджеста они поддерживают?
4. Какими свойствами должна обладать ЭЦП? Какие требования предъявляются к ЭЦП?
5. Какой метод распределения открытых ключей используется в схеме ЭЦП, предложенной в лабораторной работе?
6. Какие стандарты ЭЦП вы знаете? Какие размеры ключей и размеры ЭЦП в них используются?

4 Вопросы к зачету

1. Понятие информационной безопасности.
2. Модели безопасности.
3. Уязвимости и угрозы. Виды угроз.
4. Общая схема процесса обеспечения безопасности. Управление рисками.
5. Уровни информационной безопасности.
6. Законодательный уровень ИБ.
7. Административные меры. Политика ИБ.
8. Программно-технические меры. Сервисы и базовые принципы ИБ.
9. Криптография как наука. Основные понятия.
10. Модель традиционной криптосистемы. Правило Керкхоффа.
11. Понятие криптосистемы. Типы криптографических систем.
12. Сети Файстеля.
13. Блочные режимы работы шифров. Методы заполнения последнего блока.
14. Поточные режимы работы блочных шифров. Преимущества, недостатки, применение.
15. Характеристика современных симметричных блочных шифров. Объединение блочных шифров.
16. Современные симметричные алгоритмы шифрования (ГОСТ, AES, СТБ, RC, TripleDES).
17. Модель криптосистемы с открытым ключом.
18. Принципы построения, применение, недостатки криптосистем с открытым ключом.
19. Алгоритм RSA. Безопасность и быстродействие RSA.
20. Криптографические протоколы.
21. Распределение секретных ключей.
22. Методы распределения открытых ключей.
23. Понятие аутентификации. Факторы и виды аутентификации.
24. Парольная аутентификация.
25. Биометрия.
26. Методы аутентификации сообщений.
27. Понятие хеш-функции. Свойства криптографически стойкой хеш-функции.
28. Обобщенная структура функции хэширования.
29. Алгоритмы хэширования.
30. Непосредственная и арбитражная цифровая подпись. Протоколы ЭЦП.
31. Основные концепции Kerberos.
32. Диалог аутентификации Kerberos.
33. Понятие сертификата. Система сертификации. Использование сертификатов открытых ключей.

34. Стандарт X.509. Структура сертификата.
35. Использование криптографических алгоритмов. Аппаратное и программное шифрование. Защита файлов для хранения.
36. Канальное и сквозное шифрование и их связь с эталонной моделью OSI.
37. Понятие авторизации. Методы управления доступом.
38. Дискреционный и мандатный методы управления доступом
39. Ролевое управление доступом.
40. Управление доступом в операционных системах.
41. Общая характеристика, применение и преимущества протокола IPSec.
42. Архитектура IPSec.
43. Протокол SSL/TLS. Общая характеристика.
44. Средства защиты электронной почты.
45. Средства защиты WWW.

5 Тесты

5.1 Тест №1

1. Меры информационной безопасности направлены на защиту от:
   a) нанесения неприемлемого ущерба
   b) нанесения любого ущерба
   c) посягательств на информационную независимость государства
   d) раскрытия коммерческой тайны
2. Что из перечисленного относится к числу основных аспектов информационной безопасности согласно модели КЦД?
   a) доступность
   b) надежность
   c) аутентификация
   d) конфиденциальность
   e) целостность
   f) защита от копирования
3. Что такое защита информации?
   a) защита от разрушения
   b) комплекс мероприятий, направленных на обеспечение информационной безопасности
   c) гарантия того, что информация будет доступна только авторизованным пользователям
   d) защита от несанкционированного доступа
4. Окно опасности – это:
   a) промежуток времени
   b) часть пространства
   c) окно для ввода имени пользователя и пароля

d) реализованная угроза

5. Самыми опасными угрозами являются:

a) вирусные инфекции

b) атаки хакеров

c) стихийные бедствия

d) непреднамеренные ошибки штатных сотрудников

6. Агрессивное потребление ресурсов является угрозой:

a) конфиденциальности

b) целостности

c) доступности

7. Потенциальная возможность нарушить информационную безопасность- это:

a) атака

b) угроза

c) риск

d) окно опасности

8. DDos-атаки - это атаки на:

a) конфиденциальность

b) целостность

c) доступность

9. Вычислите величину риска по заданным значениям

Угроза 1: ущерб = 10руб., вероятность ущерба = 0,2
Угроза 2: ущерб = 100руб., вероятность ущерба = 0,5

10. Из принципа разнообразия защитных средств следует, что:

a) в разных точках подключения корпоративной сети к Интернет необходимо устанавливать разные межсетевые экраны

b) каждую точку подключения корпоративной сети к Интернет необходимо защищать несколькими видами средств безопасности

c) защитные средства нужно менять как можно чаще

d) каждому сотруднику предприятия должны предоставляться максимальные привилегии для выполнения должностных обязанностей.

5.2 Тест № 2

1. Криптосистема – это
   a) восстановление исходного текста без знания ключа
   b) пара алгоритмов - шифрование и расшифрование
   c) обработка информации с помощью одного из криптографических алгоритмов
   d) пара процедур - шифрование и дешифрование
2. Правило Керкхоффа гласит, что надежность традиционного шифрования определяется:
   a) сложностью алгоритма шифрования
   b) секретностью алгоритма шифрования
   c) сложностью ключа
   d) секретностью алгоритма шифрования и секретностью ключа
   e) секретностью ключа
3. Криптостойкость - это:
   a) наука, изучающая методы взлома шифров
   b) сложность алгоритма раскрытия шифра
   c) отсутствие информации, достаточной для однозначного восстановления открытого текста
4. В основе классификации криптосистем лежит
   a) число применяемых ключей
   b) стойкость алгоритма шифрования
   c) метод взлома шифра
   d) размер сообщения
5. На практике применяются
   a) криптосистемы, защищенные по вычислениям
   b) абсолютно стойкие криптосистемы

c) классические криптосистемы

6. Если шифрованный текст не содержит информации, достаточной для однозначного восстановления открытого текста, то криптосистема называется

a) абсолютно стойкой

b) защищенной по вычислениям

c) идеальным шифром

d) криптостойкой

7. Шифр Цезаря является (выберите правильные варианты)

a) классическим

b) симметричным

c) асимметричным

d) подстановочным

e) перестановочным

8. Совокупностью методов, предназначенных для сокрытия факта существования сообщения занимается

a) криптология

b) криптография

c) стеганография

d) криптоанализ

9. Классические шифры

a) использовались до появления компьютеров

b) никогда не применялись на практике

c) использовались до изобретения криптографии с открытым ключом

d) применяются в настоящее время

10. По методу обработки открытого текста различают:

a) блочные и поточные шифры

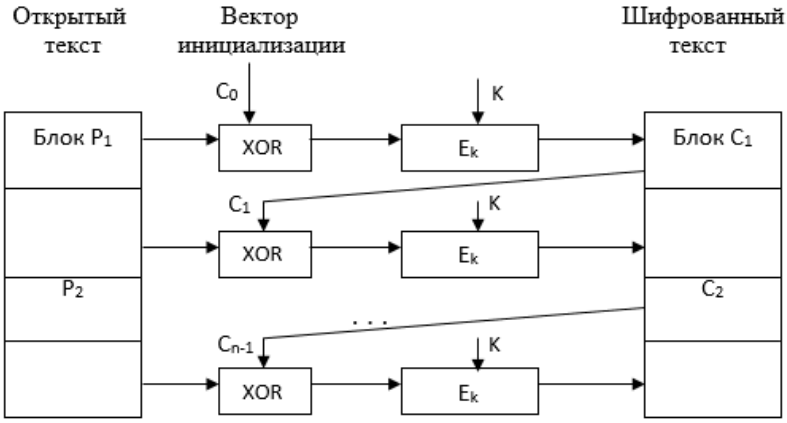
b) подстановочные и перестановочные шифры

c) симметричные и асимметричные шифры

d) низкоскоростные и высокоскоростные

5.3 Тест № 3

1. Укажите максимальный размер ключа алгоритма AES
2. Симметричное шифрование сообщения обеспечивает
   a) электронную цифровую подпись
   b) конфиденциальность
   c) аутентификацию
   d) целостность
3. Гарантия того, что доступ к информации смогут получить только санкционированные пользователи, это
   a) конфиденциальность
   b) целостность
   c) доступность
   d) аутентификация
4. В каком случае сеть Файстеля называется гетерогенной?
   a) размер левой части блока не равен размеру правой части
   b) размер левой части блока равен размеру правой части
   c) параметры функции раунда изменяются от раунда к раунду
   d) параметры функции раунда не изменяются от раунда к раунду
5. Схема какого режима шифрования приведена на рисунке?



a) ECB

b) CBC

c) CFB

d) OFB

e) CTR

6. Чтобы обеспечить возможность расшифрования в криптографии используются:

a) сингулярные преобразования

b) несингулярные преобразования

c) диффузия

d) конфузия

e) умножение и деление

7. Выберите из перечисленных режимов шифрования блочные:

a) электронная кодовая книга

b) режим сцепления шифрованных блоков

c) шифрованная обратная связь

d) обратная связь по выходу алгоритма шифрования

e) режим со счетчиком

8. Какие из перечисленных криптографических алгоритмов и стандартов относятся к симметричным?

a) ГОСТ 28147-89

b) Диффи-Хеллмана

c) 3DES

d) RC

e) SHA

f) RSA

9. Какие режимы шифрования могут использоваться для аутентификации?

    a) ECB

    b) CBC

    c) CFB

    d) OFB

    e) CTR

10. Назовите первый стандарт шифрования

5.4 Тест № 4

1. В асимметричной криптосистеме отправитель и получатель сообщения используют:
   a) один и тот же секретный ключ
   b) разные секретные ключи
   c) открытый и секретный ключ
   d) открытый и личный ключ
2. Каким ключом отправитель должен зашифровать сообщение для обеспечения конфиденциальности сообщений с помощью асимметричной криптосистемы?
   a) своим личным ключом
   b) своим открытым ключом
   c) открытым ключом получателя
   d) личным ключом получателя
3. Какие из перечисленных криптографических алгоритмов и стандартов относятся к асимметричным?
   a) RSA
   b) AES
   c) DSS
   d) ГОСТ 28147-89
   e) Диффи- Хеллмана
   f) MD
   g) SHA
   h) Кузнечик
4. Какой из перечисленных алгоритмов может использоваться для обеспечения конфиденциальности сообщений?
   a) RSA
   b) DSS

c) Диффи- Хеллмана

d) SHA

e) Стрибог

5. Выберите лишний термин из списка

a) свертка

b) имитовставка

c) профиль

d) дайджест

e) функция сжатия

6. Чем вычисление кода аутентификации отличается от хеширования?

a) используется секретный ключ

b) для вычислений не требуется обратимость

c) размер кода аутентификации фиксирован и не зависит от размера исходного сообщения

7. Аутентификация сообщения - это

a) подтверждение подлинности сообщения

b) гарантия сохранения целостности сообщения

c) гарантия того, что сообщение пришло от указанного источника

d) гарантия того, что сообщение пришло от указанного источника и не было изменено в пути следования

8. Для верификации ЭЦП используется

a) личный ключ создателя ЭЦП

b) открытый ключ создателя ЭЦП

c) личный ключ верификатора

d) открытый ключ верификатора

9. Как задается значение переменной сцепления в итерированной функции хеширования?

a) является частью алгоритма

b) вводится пользователем

c) генерируется случайным образом

10. Сколько раз к сообщению будет применена функция сжатия при использовании итерированной функции хеширования при заданных значениях?

Длина сообщения - 160 байт

Длина блока 128 бит

5.5 Тест № 5

1. В криптографических протоколах Боб - это
   a) первый участник
   b) второй участник
   c) активный взломщик
   d) пассивный взломщик
   e) верификатор
   f) доверенный посредник
2. Сколько открытых ключей требуется на 15 участников обмена?
3. Каким из перечисленных требований должна удовлетворять "соль", используемая с паролем?
   a) соль должна быть случайным значением
   b) значение соли является секретным
   c) значение соли хранится в БД в открытом виде
   d) соль вводит пользователь
   e) размер соли должен соответствовать длине хеш-кода
   f) значение соли генерируется ОС одинаковым для всех пользователей
4. Выберите группу аутентификаторов, к которой относится криптографический ключ:
   a) то, что пользователь знает
   b) то, чем пользователь владеет
   c) то, что является частью пользователя
5. Для чего в протоколе Kerberos предназначены мандаты?
   a) для аутентификации клиента
   b) для аутентификации сервера
   c) для взаимной аутентификации сторон
   d) для передачи ключей шифрования

6. Для чего используется алгоритм Диффи-Хеллмана?
    a) распределения секретных ключей
    b) шифрования сообщений
    c) распределения открытых ключей
    d) генерации ЭЦП
    e) аутентификации пользователей
7. В качестве оказии могут использоваться:
    a) время
    b) счетчик
    c) случайное число
    d) сеансовый ключ
    e) главный ключ
    f) ответ на запрос
8. Сертификат - это (выберите правильные утверждения)
    a) способ распределения секретных ключей
    b) способ распределения открытых ключей
    c) комплекс программных средств и методик, предназначенный для администрирования открытых ключей
    d) набор данных, позволяющий сохранить пару ключей (открытый, личный) каждого участника
    e) набор данных, позволяющий сопоставить открытый ключ с объектом, имеющим соответствующий личный ключ.
9. Какой вид аутентификации является более безопасным, чем все остальные?
    a) многоразовые пароли
    b) одноразовые пароли
    c) биометрия
    d) аппаратные аутентификаторы
    e) примерно одинаковый уровень безопасности

10. Подтверждение подлинности - это

a) аутентификация

b) идентификация

c) авторизация

5.6 Тест № 6

1. К какому уровню стека TCP/IP относится протокол IPSec?
   a) прикладной
   b) транспортный
   c) межсетевого взаимодействия
   d) сетевых интерфейсов
2. В каком режиме работы протокола IPSec защита распространяется только на полезный груз IP-пакета?
   a) туннельном
   b) транспортном
3. Над каким уровнем стека TCP/IP работает протокол SSL/TLS?
   a) прикладной
   b) транспортный
   c) межсетевого взаимодействия
   d) сетевых интерфейсов
4. Какой протокол используется для защиты информации, передаваемой http-протоколом?
5. Какой вид шифрования обеспечивает безопасность трафика?
   a) канальное шифрование
   b) сквозное шифрование
6. В каком методе управления доступом информация о разрешениях хранится в виде списков ACL?
   a) дискреционном
   b) мандатном
   c) ролевом
7. В каком из методов управления доступом за управление доступом отвечает владелец ресурса?
   a) дискреционном

b) мандатном

c) ролевом

8. С помощью какого протокола в SSL передаются данные прикладных протоколов?

a) протокол квитирования

b) протокол изменения параметров шифрования

c) протокол извещения

d) протокол записи

9. Использование чужого IP-адреса, логина, пароля это:

a) спуфинг

b) фишинг

c) снифинг

10. Атака SYN-flood – это

a) атака на протокол TCP

b) DoS-атака

c) атака на протокол IP

d) атака перенаправления трафика

6 Список литературы

Основная литература

1. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов/ В.Г. Олифер, Н.А. Олифер. - 6-е изд. - СПб: Питер, 2020. - 1008с.
2. Джейсон Андресс Защита данных. От авторизации до аудита. (Серия «Для профессионалов»)/ А. Джейсон. — СПБ.: Питер, 2021. — 272 с..
3. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, и исходный код на С./ Б. Шнайер. - СПБ.: ООО «Альфа-книга», 2017 – 1040 с.
4. Нестеров, С.А. Информационная безопасность: Учебник и практикум для академического бакалавриата/ С.А.Нестеров. – М.: Изд-во Юрайт, 2017. – 321с. – Серия: Университеты России (biblio-online.ru)
5. Курило, А. П. Основы управления информационной безопасностью. Учебное пособие для вузов/ А.П. Курило, Н.Г. Милославская, М.Ю. Сенаторов, А.И. Толстой. 2-е изд. – М.: Горячая Линия – Телеком, 2019. – 244 с.
6. Кудашов, В. И. Управление интеллектуальной собственностью: учеб. пос. для студ. вузов / В. И. Кудашов. – Минск: ИВЦ Минфина, 2017. – 359 с.
7. Олифер, В. Г. Безопасность компьютерных сетей / В. Г. Олифер, Н. А. Олифер. - Москва : Горячая линия-Телеком, 2016. - 643 с. : ил., табл.
8. Джонсон Д.Б., Деоган Д., Савано Д. Безопасно by design (Серия «Библиотека программиста»). — СПб.: Питер, 2021. — 432 с.
9. Хоффман Эндрю Безопасность веб-приложений (Серия «Бестселлеры O’Reilly»). — СПб.: Питер, 2021. — 336 с.

Дополнительная литература

1. Родичев, Ю.А. Нормативная база и стандарты в области информационной безопасности: Учебник для вузов/ Ю.А. Родичев. – СПб.: Питер, 2017. – 256 с.
2. Шеннон, К.Э. Теория связи в секретных системах // К.Э: Шеннон Работы по теории информации и кибернетике. М.: ИЛ, 1963. – с.333-402.
3. СТБ ГОСТ Р 50922-2000 "Защита информации. Основные термины и определения"
4. СТБ 34.101.27-2022 "Информационные технологии и безопасность. Средства криптографической защиты информации. Требования безопасности".
5. СТБ 34.101.31-2020 "Информационные технологии и безопасность. Алгоритмы шифрования и контроля целостности".
6. СТБ 34.101.45-2013 "Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых".

7. СТБ 34.101.47-2017 "Информационные технологии и безопасность. Алгоритмы генерации псевдослучайных чисел".
8. СТБ 34.101.60-2014 "Информационные технологии и безопасность. Алгоритмы разделения секрета".
9. СТБ 34.101.65-2014 "Информационные технологии и безопасность. Протокол защиты транспортного уровня (TLS)".
10. СТБ 34.101.66-2014 "Информационные технологии и безопасность. Протоколы формирования общего ключа на основе эллиптических кривых".
11. СТБ 34.101.77-2020 "Информационные технологии и безопасность. Криптографические алгоритмы на основе sponge-функции". (Стандарт хеширования)
12. СТБ 34.101.87-2022 "Информационные технологии и безопасность. Инфраструктуры аутентификации".