МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра управления в технических системах

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ Е.И. Луковникова
«_____» _____ 201__ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ ТЕХНОЛОГИИ ПРОГРАММНОЙ ЗАЩИТЫ В ИНТЕРНЕТЕ

Б1.В.ДВ.11

НАПРАВЛЕНИЕ ПОДГОТОВКИ

11.03.02 Инфокоммуникационные технологии и системы связи

ПРОФИЛЬ ПОДГОТОВКИ

Многоканальные телекоммуникационные системы

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к экспериментально-исследовательскому виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Изучение студентами особенностей технологии программной защиты в Интернете, осуществляющих защиту электронной почты, сетевую диагностику, безопасности доступа, конфиденциальности и целостности информации в Интернет.

Задачи дисциплины

Формирование знаний, умений и навыков, позволяющих проводить самостоятельный анализ сетевых технологий глобальных сетей, как изучаемых в настоящей дисциплине, так и находящихся за ее рамками.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ОПК-1	Способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной тайны.	**знать:** - основные закономерности передачи информации в инфокоммуникационных системах, основные виды сигналов, используемых в телекоммуникационных системах, особенности передачи различных сигналов по каналам и тракам телекоммуникационных систем; **уметь:** - формулировать основные технические требования к телекоммуникационным сетям и систем; **владеть:** - начальными навыками разработки программного обеспечения сигнальных процессов и микроконтроллеров.
ПК-17	Способность применять современные теоретические и экспериментальные методы исследования с целью создания новых перспективных средств электросвязи и информатики	**знать:** - основы цифровой вычислительной техники, структуры и функционирование локальных вычислительных сетей и глобальной сети Интернет; **уметь:** - оценивать основные проблемы, связанные с эксплуатацией и внедрением новой телекоммуникационной техники **владеть:** - начальными навыками отладки с использованием соответствующих отладочных средств программного обеспечения сигнальных процессов и микроконтроллеров

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.ДВ.11 Технологии программной защиты в интернете относится к дисциплинам по выбору

Дисциплина технологии программной защиты в интернете базируется на знаниях, полученных при изучении дисциплин Б1.В.11 Сетевые технологии высокоскоростной передачи данных, Б1.Б.17 Основы построения инфокоммуникационных систем и сетей, Б1.В.18 Моделирование сетей связи, Б1.Б.16 Вычислительная техника и информационные технологии.

Основываясь на изучении перечисленных дисциплин, технологии программной

защиты в интернете представляет основу для Б2.П.2 производственная (преддипломная) практика и подготовки к Б3 государственной итоговой аттестации.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Контрольная работа*	*Вид промежуточной аттестации*
			Всего часов (с экз.)	*Аудиторных часов*	*Лекции*	*Лабораторные работы*	*Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Очная	4	8	144	60	24	24	12	48	кр	Экзамен
Заочная	-	-	-	-	-	-	-	-	-	-
Заочная (ускоренное обучение)	-	-	-	-	-	-	-	-	-	-
Очно-заочная	-	-	-	-	-	-	-	-	-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудо-емкость (час.)*	*в т.ч. в интерактивной, актив-ной, иннова-ционной формах, (час.)*	*Распределение по семестрам, час*
			8
1	2	3	4
I. Контактная работа обучающихся с преподавателем (всего)	60	14	60
Лекции (Лк)	24	4	24
Лабораторные работы (ЛР)	24	6	24
Практические работы (ПР)	12	4	12
Контрольная работа (кр)	+	-	+
Индивидуальные(групповые) консультации	+	-	+
II.Самостоятельная работа обучающихся (СР)	48	-	48
Подготовка к лабораторным работам	12	-	12
Подготовка к практическим работам	12		12
Подготовка к экзамену в течение семестра	12	-	12
Выполнение контрольной работы	12	-	12
III. Промежуточная аттестация экзамен	36	-	36
Общая трудоемкость дисциплины …..... час.	144	-	144
зач. ед.	4	-	4

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий - для очной формы обучения:

№ раз-дела и темы	*Наименование раздела и тема дисциплины*	*Трудоем-кость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*			
			учебные занятия			*самостоя тельная работа обучаю-щихся*
			лекции	*лабораторные работы*	*практиче ские работы*	
1	2	3	4	5	6	7
1.	**Информационная безопасность**	**6**	**2**	**-**	**-**	**4**
1.1.	Введение в информационную безопасность	3	1	-	-	2
1.2.	Модель сетевой безопасности .Классификация сетевых атак	3	1	-	-	2
2.	**Защита информации при помощи криптографии**	**16**	**2**	**9**	**-**	**5**
2.1.	Защита передаваемых и хранимых секретных данных от разглашения и искажения	8	1	4.5	-	2.5
2.2.	Задача подтверждения авторства сообщения	8	1	4.5	-	2.5
3.	**Классификация шифров**	**17**	**2**	**10**	**-**	**5**
3.1.	Перестановочные шифры	8.5	1	5	-	25
3.2.	Классификация методов дешифрования. Модель предполагаемого противника. Правила Керкхоффа	8.5	1	5	-	2.5
4.	**Криптосистемы**	**25**	**3**	**5**	**12**	**5**
4.1.	Устройство шифров	5	1	-	3	1
4.2.	Поточные шифры	11	1	5	3	2
4.3.	Алгоритм DES	4.5	0.5	-	3	1
4.4.	Алгоритм ГОСТ 28147	4.5	0.5	-	3	1
5.	**Современные криптографические системы**	**8**	**3**	**-**	**-**	**5**
5.1.	Предварительные математические понятия	2	1	-	-	1
5.2.	Обоснование разработки	2	1	-	-	1
5.3.	Спецификация алгоритма	1.5	0.5	-	-	1
5.4.	Преимущества алгоритма	2.5	0.5	-	-	2
6.	**Теория сложности вычислений**	**6**	**2**	**-**	**-**	**4**
6.1.	Сложность вычислений.	3	1	-	-	2

6.2.	Сложность алгоритмов.	3	1	-	-	2
7.	**Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма**	**7**	**2**	**-**	**-**	**5**
7.1.	Алгоритм RSA	3.5	1	-	-	2.5
7.2.	Криптосистема Эль-Гамаля.	3.5	1	-	-	2.5
8.	**Электронная подпись.**	**8**	**3**	**-**	**-**	**5**
8.1.	Общая схема цифровой подписи	3	1	-	-	2
8.2.	Стандарт цифровой подписи DSS	2	1	-	-	1
8.3.	Электронная подпись на основе алгоритма RSA	3	1	-	-	2
9.	**Хеш-функции**	**7**	**2**	**-**	**-**	**5**
9.1.	Требования к хэш-функциям	3.5	1	-	-	2.5
9.2.	Простые хэш-функции	3.5	1	-	-	2.5
10.	**Аутентификация и обмен ключами.**	**8**	**3**	-	-	**5**
10.1.	Алгоритмы распределения ключей с использованием третьей доверенной стороны Понятие мастер-ключа	3	1	-	-	2
10.2.	Протоколы аутентификации	2	1	-	-	1
10.3.	Использование шифрования с открытым ключом	3	1	-	-	2
	ИТОГО	**108**	**24**	**24**	**12**	**48**

4.2. Содержание дисциплины, структурированное по разделам и темам

1. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ.

Введение в информационную безопасность

За несколько последних десятилетий требования к информационной безопасности существенно изменились. До начала широкого использования автоматизированных систем обработки данных безопасность информации достигалась исключительно физическими и административными мерами. С появлением компьютеров стала очевидной необходимость использования автоматических средств защиты файлов данных и программной среды. Следующий этап развития автоматических средств защиты связан с появлением распределенных систем обработки данных и компьютерных сетей, в которых средства сетевой безопасности используются в первую очередь для защиты передаваемых по сетям данных. В наиболее полной трактовке под средствами сетевой безопасности мы будем иметь в виду меры предотвращения нарушений безопасности, которые возникают при передаче информации по сетям, а также меры, позволяющие определять, что такие нарушения безопасности имели место. Именно изучение средств сетевой безопасности и связанных с ними теоретических и прикладных проблем, составляет основной материал книги.

Термины "безопасность информации" и "защита информации" отнюдь не являются синонимами. Термин "безопасность" включает в себя не только понятие защиты, но также и *аутентификацию*, аудит, обнаружение проникновения.

Перечислим некоторые характерные проблемы, связанные с безопасностью, которые возникают при использовании компьютерных сетей:

1. Фирма имеет несколько офисов, расположенных на достаточно большом расстоянии друг от друга. При пересылке конфиденциальной информации по общедоступной сети (например, Internet) необходимо быть уверенным, что никто не сможет ни подсмотреть, ни изменить эту информацию.
2. Сетевой администратор осуществляет удаленное управление компьютером. Пользователь перехватывает управляющее сообщение, изменяет его содержание и отправляет сообщение на данный компьютер.
3. Пользователь несанкционированно получает доступ к удаленному компьютеру с правами законного пользователя, либо, имея право доступа к компьютеру, получает доступ с гораздо большими правами.
4. Фирма открывает *Internet*-магазин, который принимает оплату в электронном виде. В этом случае продавец должен быть уверен, что он отпускает товар, который действительно оплачен, а покупатель должен иметь гарантии, что он, во-первых, получит оплаченный товар, а во-вторых, номер его кредитной карточки не станет никому известен.
5. Фирма открывает свой сайт в *Internet*. В какой-то момент содержимое сайта заменяется новым, либо возникает такой поток и такой способ обращений к сайту, что сервер не справляется с обработкой запросов. В результате обычные посетители сайта либо видят информацию, не имеющую к фирме никакого отношения, либо просто не могут попасть на сайт фирмы.

Рассмотрим основные понятия, относящиеся к информационной безопасности, и их взаимосвязь.

Собственник определяет множество **информационных ценностей**, которые должны быть защищены от различного рода *атак*. *Атаки* осуществляются *противниками* или *оппонентами*, использующими различные *уязвимости* в защищаемых ценностях. Основными нарушениями безопасности являются раскрытие информационных ценностей (потеря *конфиденциальности*), их неавторизованная модификация (потеря *целостности*) или неавторизованная потеря доступа к этим ценностям (потеря *доступности*).

Собственники информационных ценностей анализируют *уязвимости* защищаемых ресурсов и возможные *атаки*, которые могут иметь место в конкретном окружении. В результате такого анализа определяются *риски* для данного набора информационных ценностей. Этот анализ определяет выбор контрмер, который задается политикой безопасности и обеспечивается с помощью *механизмов* и *сервисов безопасности*. Следует учитывать, что отдельные *уязвимости* могут сохраниться и после применения *механизмов* и *сервисов безопасности*. ***Политика безопасности*** определяет согласованную совокупность *механизмов* и *сервисов безопасности*, адекватную защищаемым ценностям и окружению, в котором они используются.

На рис.1.1 показана взаимосвязь рассмотренных выше понятий информационной безопасности. Дадим следующие определения:

Уязвимость - слабое место в системе, с использованием которого может быть осуществлена *атака*.

Риск - вероятность того, что конкретная *атака* будет осуществлена с использованием конкретной *уязвимости*. В конечном счете, каждая организация должна принять решение о допустимом для нее уровне *риска*. Это решение должно найти отражение в политике безопасности, принятой в организации.

Политика безопасности - правила, директивы и практические навыки, которые определяют то, как информационные ценности обрабатываются, защищаются и распространяются в организации и между информационными системами; набор критериев для предоставления *сервисов безопасности*.

Атака - любое действие, нарушающее безопасность информационной системы. Более формально можно сказать, что *атака* - это действие или последовательность связанных между собой действий, использующих *уязвимости* данной информационной системы и приводящих к нарушению политики безопасности.

Рис. 1.1. Взаимосвязь основных понятий безопасности информационных систем

Механизм безопасности - программное и/или аппаратное средство, которое определяет и/или предотвращает *атаку*.

Сервис безопасности - сервис, который обеспечивает задаваемую политикой безопасность систем и/или передаваемых данных, либо определяет осуществление *атаки*. *Сервис* использует один или более механизмов безопасности.

Рассмотрим модель сетевой безопасности и основные типы *атак*, которые могут осуществляться в этом случае. Затем рассмотрим основные типы *сервисов* и *механизмов безопасности*, предотвращающих такие *атаки*.

Модель сетевой безопасности .Классификация сетевых атак

В общем случае существует информационный поток от отправителя (файл, пользователь, компьютер) к получателю (файл, пользователь, компьютер):

Рис. 1.2. Информационный поток

Все *атаки* можно разделить на два класса: *пассивные* и *активные*.

I. Пассивная атака

Пассивной называется такая ***атака***, при которой *противник* не имеет возможности модифицировать передаваемые сообщения и вставлять в информационный канал между отправителем и получателем свои сообщения. Целью *пассивной атаки* может быть только прослушивание передаваемых сообщений и анализ трафика.

Активной называется такая ***атака***, при которой *противник* имеет возможность модифицировать передаваемые сообщения и вставлять свои сообщения. Различают следующие типы *активных атак*:

Рис. 1.3. Пассивная атака

II. Активная атака

Отказ в обслуживании - *DoS-атака (Denial of Service)*

Отказ в обслуживании нарушает нормальное функционирование сетевых сервисов. *Противник* может перехватывать все сообщения, направляемые определенному адресату. Другим примером подобной *атаки* является создание значительного трафика, в результате чего сетевой сервис не сможет обрабатывать запросы законных клиентов. Классическим примером такой *атаки* в сетях TCP/IP является SYN-атака, при которой нарушитель посылает пакеты, инициирующие установление TCP-соединения, но не посылает пакеты, завершающие установление этого соединения. В результате может произойти переполнение памяти на сервере, и серверу не удастся установить соединение с законными пользователями.

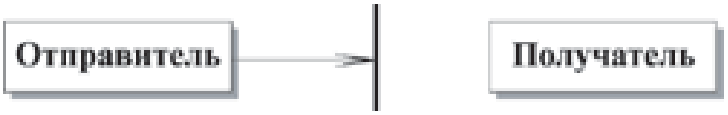


Рис. 1.4. DoS-атака

Модификация потока данных - *атака "man in the middle"*

Модификация потока данных означает либо изменение содержимого пересылаемого сообщения, либо изменение порядка сообщений.

Рис. 1.5. Атака "man in the middle"

1. **Создание ложного потока (фальсификация)**

Фальсификация (нарушение аутентичности) означает попытку одного субъекта выдать себя за другого.

Рис. 1.6. Создание ложного потока

Повторное использование

Повторное использование означает пассивный захват данных с последующей их пересылкой для получения несанкционированного доступа - это так называемая ***replay-атака***. На самом деле *replay-атаки* являются одним из вариантов фальсификации, но в силу того, что это один из наиболее распространенных вариантов *атаки* для получения несанкционированного доступа, его часто рассматривают как отдельный тип *атаки*.

Рис. 1.7. Replay-атака

Перечисленные *атаки* могут существовать в любых типах сетей, а не только в сетях, использующих в качестве транспорта протоколы TCP/IP, и на любом уровне модели OSI. Но в сетях, построенных на основе TCP/IP, *атаки* встречаются чаще всего, потому что, во-первых, Internet стал самой распространенной сетью, а во-вторых, при разработке протоколов TCP/IP требования безопасности никак не учитывались.

Сервисы безопасности

Основными *сервисами безопасности* являются следующие:

Конфиденциальность - предотвращение *пассивных атак* для передаваемых или хранимых данных.

Аутентификация - подтверждение того, что информация получена из законного источника, и получатель действительно является тем, за кого себя выдает. В случае передачи единственного сообщения *аутентификация* должна гарантировать, что получателем сообщения является тот, кто нужно, и сообщение получено из заявленного источника. В случае установления соединения имеют место два аспекта. Во-первых, при инициализации соединения *сервис* должен гарантировать, что оба участника являются требуемыми. Во-вторых, *сервис* должен гарантировать, что на соединение не воздействуют таким образом, что *третья сторона* сможет маскироваться под одну из легальных сторон уже после установления соединения.

Целостность - *сервис*, гарантирующий, что информация при хранении или передаче не изменилась. Может применяться к потоку сообщений, единственному сообщению или отдельным полям в сообщении, а также к хранимым файлам и отдельным записям файлов.

Невозможность отказа - невозможность, как для получателя, так и для отправителя, отказаться от факта передачи. Таким образом, когда сообщение отправлено, получатель может убедиться, что это сделал легальный отправитель. Аналогично, когда сообщение пришло, отправитель может убедиться, что оно получено легальным получателем.

Контроль доступа - возможность ограничить и контролировать доступ к системам и приложениям по коммуникационным линиям.

Доступность - результатом *атак* может быть потеря или снижение доступности того или иного сервиса. Данный *сервис* предназначен для того, чтобы минимизировать возможность осуществления *DoS-атак*.

Механизмы безопасности

Перечислим основные *механизмы безопасности*:

Алгоритмы симметричного шифрования - алгоритмы шифрования, в которых для шифрования и дешифрования используется один и тот же ключ или ключ дешифрования легко может быть получен из ключа шифрования.

Алгоритмы асимметричного шифрования - алгоритмы шифрования, в которых для шифрования и дешифрования используются два разных ключа, называемые открытым и закрытым ключами, причем, зная один из ключей, вычислить другой невозможно.

Хэш-функции - функции, входным значением которых является сообщение произвольной длины, а выходным значением - сообщение фиксированной длины. *Хэш-функции* обладают рядом свойств, которые позволяют с высокой долей вероятности определять изменение входного сообщения.

Модель сетевого взаимодействия

Модель безопасного сетевого взаимодействия в общем виде можно представить следующим образом:

Рис. 1.8. Модель сетевой безопасности

Сообщение, которое передается от одного участника другому, проходит через различного рода сети. При этом будем считать, что устанавливается логический информационный канал от отправителя к получателю с использованием различных коммуникационных протоколов (например, TCP/IP).

Средства безопасности необходимы, если требуется защитить передаваемую информацию от *противника*, который может представлять угрозу *конфиденциальности*, *аутентификации*, *целостности* и т.п. Все технологии повышения безопасности имеют два компонента:

1. Относительно безопасная передача информации. Примером является шифрование, когда сообщение изменяется таким образом, что становится нечитаемым для *противника*, и, возможно, дополняется кодом, который основан на содержимом сообщения и может использоваться для *аутентификации* отправителя и обеспечения *целостности* сообщения.
2. Некоторая секретная информация, разделяемая обоими участниками и неизвестная *противнику*. Примером является ключ шифрования.

Кроме того, в некоторых случаях для обеспечения безопасной передачи бывает необходима *третья доверенная сторона* (third trusted party - TTP). Например, *третья сторона* может быть ответственной за распределение между двумя участниками секретной информации, которая не стала бы доступна *противнику*. Либо *третья сторона* может использоваться для решения споров между двумя участниками относительно достоверности передаваемого сообщения.

Из данной общей модели вытекают три основные задачи, которые необходимо решить при разработке конкретного *сервиса безопасности*:

1. Разработать алгоритм шифрования/дешифрования для выполнения безопасной передачи информации. Алгоритм должен быть таким, чтобы *противник* не мог расшифровать перехваченное сообщение, не зная секретную информацию.
2. Создать секретную информацию, используемую алгоритмом шифрования.
3. Разработать протокол обмена сообщениями для распределения разделяемой секретной информации таким образом, чтобы она не стала известна *противнику*.

Модель безопасности информационной системы

Существуют и другие относящиеся к безопасности ситуации, которые не соответствуют описанной выше модели сетевой безопасности. Общую модель этих ситуаций можно проиллюстрировать следующим образом:

Данная модель иллюстрирует концепцию безопасности информационной системы, с помощью которой предотвращается нежелательный доступ. Хакер, который пытается осуществить незаконное проникновение в системы, доступные по сети, может просто получать удовольствие от взлома, а может стараться повредить информационную систему и/или внедрить в нее что-нибудь для своих целей. Например, целью хакера может быть получение номеров кредитных карточек, хранящихся в системе.

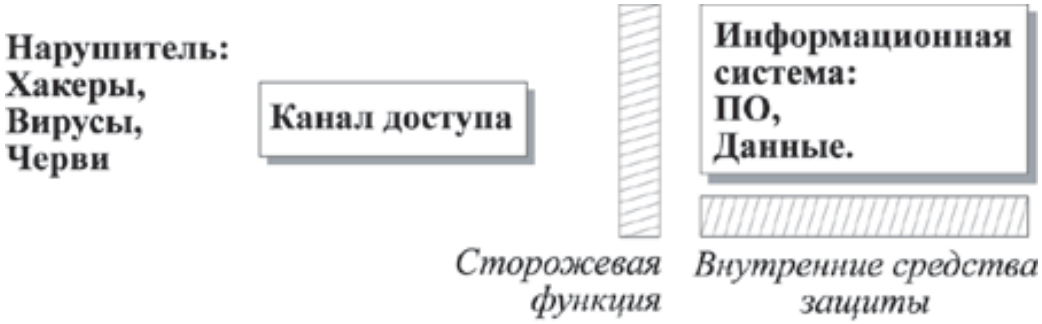


Рис. 1.9. Модель безопасности информационной системы

Другим типом нежелательного доступа является размещение в вычислительной системе чего-либо, что воздействует на прикладные программы и программные утилиты, такие как редакторы, компиляторы и т.п. Таким образом, существует два типа *атак*:

1. Доступ к информации с целью получения или модификации хранящихся в системе данных.
2. *Атака* на сервисы, чтобы помешать использовать их.

Вирусы и черви - примеры подобных *атак*. Такие *атаки* могут осуществляться как с помощью дискет, так и по сети.

Сервисы безопасности, которые предотвращают нежелательный доступ, можно разбить на две категории:

1. Первая категория определяется в терминах сторожевой функции. Эти *механизмы* включают процедуры входа, основанные, например, на использовании пароля, что позволяет разрешить доступ только авторизованным пользователям. Эти *механизмы* также включают различные защитные экраны (firewalls), которые предотвращают *атаки* на различных уровнях стека протоколов TCP/IP, и, в частности, позволяют предупреждать проникновение червей, вирусов, а также предотвращать другие подобные *атаки*.
2. Вторая линия обороны состоит из различных внутренних мониторов, контролирующих доступ и анализирующих деятельность пользователей.

Одним из основных понятий при обеспечении безопасности информационной системы является понятие ***авторизации*** - определение и предоставление прав доступа к конкретным ресурсам и/или объектам.

В основу безопасности информационной системы должны быть положены следующие основные принципы:

1. Безопасность информационной системы должна соответствовать роли и целям организации, в которой данная система установлена.
2. Обеспечение информационной безопасности требует комплексного и целостного подхода.
3. Информационная безопасность должна быть неотъемлемой частью системы управления в данной организации.
4. Информационная безопасность должна быть экономически оправданной.
5. Ответственность за обеспечение безопасности должна быть четко определена.
6. Безопасность информационной системы должна периодически переоцениваться.
7. Большое значение для обеспечения безопасности информационной системы имеют социальные факторы, а также меры административной, организационной и физической безопасности.

Интерактив 2 часа (фильма + обсуждение)

2. ЗАЩИТА ИНФОРМАЦИИ ПРИ ПОМОЩИ КРИПТОГРАФИИ

Как было отмечено ранее, злоумышленник в криптографии есть персонифицированный набор целей по отклонению информационного процесса от его штатного протекания, и возможностей по достижению этих целей. Рассмотрение проблемы ведется в предположении, что злоумышленник действует наилучшим возможным в его ситуации образом. В качестве злоумышленников могут выступать:

- законные участники процесса;
- субъекты, не являющиеся законными участниками процесса, но имеющие доступ к информации, передаваемой и обрабатываемой в ходе осуществления информационного взаимодействия и могущие повлиять на его протекание.

Если законные участники процесса не могут выступать в качестве злоумышленников, такой процесс называется "**информационным взаимодействием со взаимным доверием сторон друг другу**", понятно, что в противном имеет место "**процесс информационного взаимодействия в условии отсутствия взаимного доверия сторон**". Классы методов защиты процессов обоих типов существенно отличаются друг от друга, вторая задача сложнее - общеизвестно, что практически любую систему намного легче защитить от проникновения извне, чем от злоумышленных действий со стороны ее законных пользователей.

Надо отметить, что когда речь идет о **взаимном доверии сторон**, имеется в виду нечто большее, чем просто отношение субъектов информационного взаимодействия друг к другу. При определении этого должны рассматриваться многие факторы, например среда и окружение, в которых они работают. Для иллюстрации сказанного рассмотрим задачу защиты программного обеспечения компьютерных систем от несанкционированной модификации, которая обычно решается следующим образом:
1. При инсталляции и "легальной" модификации программ для каждой защищаемой единицы (обычно это исполняемый файл) вырабатывается контрольный код, являющийся "дальним родственником" обыкновенной контрольной суммы.
3. В соответствии с некоторым регламентом (например, при каждой загрузке системы, или один раз в определенный промежуток времени, или перед запуском программы на выполнение) проверяется соответствие защищаемой единицы контрольному коду.

Если компьютер действительно персональный, то оба эти действия выполняет один и тот же человек, но требования к "чистоте" среды совершенно различные. Первое действие выполняется при добавлении в систему нового программного обеспечения или перенастройке уже существующего, что в большинстве реальных узкоспециализированных систем происходит достаточно редко. Необходимым условием выполнения этой операции является отсутствие "закладок" в п/о, вырабатывающем контрольный код. "Чистая" среда обычно создается загрузкой операционной системы с носителя, физически защищенного от записи - единожды сформированного и выверенного, и с тех пор остающегося неизменным.

Второе действие осуществляется значительно чаще, и по определению, может выполняться в не столь "чистой" среде. Поэтому, даже если обе процедуры выполняет один и тот же человек, с точки зрения задачи это все равно два различных субъекта, и пользователь-создатель контрольного кода не должен доверять пользователю-проверяющему. В силу вышеизложенного для решения данной задачи больше подходят схемы, основанные на электронно-цифровой подписи, эффективно работающие в условиях отсутствия взаимного доверия сторон, нежели использование криптографической контрольной комбинации на основе симметричных шифров.

Задачи, решаемые криптографическими методами, отличаются друг от друга следующим:

- характером защищаемого информационного взаимодействия;
- целями злоумышленников;
- возможностями злоумышленников.

Простейший случай информационного взаимодействия - это передача данных от одного субъекта другому. Соответственно, самая распространенная задача из сферы защиты - защита передаваемой по каналам связи или хранимой в компьютерной системе информации, она исторически самая первая и до сих пор наиболее важная. Впрочем, необходимо добавить, что в последнее время в связи с проникновением электронных технологий во многие сферы жизни человека и общества возникают и принципиально новые проблемы. Одни из них первичны, такие, как уже упомянутая проблема защиты данных в каналах связи. Другие вторичны и существуют только в рамках конкретного решения той или иной первичной задачи. Например, "открытое распределение ключей" - совместная выработка двумя субъектами в ходе сеанса связи по открытому каналу общего секретного ключа таким образом, чтобы злоумышленники, "прослушивающие" канал, не смогли получить тот же самый ключ.

Рассмотрение задач из сферы криптографии начнем с задачи защиты данных, передаваемых по открытым каналам связи в наиболее полной постановке: В системе имеются две легальные стороны - "отправитель" и "получатель". Информационный процесс заключается в передаче сообщения от первого второму и считается протекающим нормально, если получатель получит сообщение без искажений, кроме него никто не ознакомится с содержанием сообщения, и если стороны не будут выставлять претензий друг другу. В задаче также присутствует злоумышленник, имеющий доступ к каналу передачи данных и стремящийся добиться отклонений от нормального течения процесса. Кроме того, каждая из легальных сторон может предпринять злоумышленные действия в отношении другой стороны. Перечислим возможные угрозы:

1. **Угрозы со стороны злоумышленника.**
1.1. Ознакомление с содержанием переданного сообщения.
1.2. Навязывание получателю ложного сообщения - как полная его фабрикация, так и внесение искажений в действительно переданное сообщение.
1.3. Изъятие переданного отправителем сообщения из системы таким образом, чтобы получатель не узнал о факте передачи сообщения;
1.4. Создание помех для нормальной работы канала передачи связи, то есть нарушение работоспособности канала связи.

2. **Угрозы со стороны законного отправителя сообщения:**
2.1. Разглашение переданного сообщения.
2.2. Отказ от авторства в действительности переданного им сообщения.
2.3. Утверждение, что некоторое сообщение отправлено получателю когда в действительности отправка не производилась.

3. **Угрозы со стороны законного получателя сообщения:**
3.1. Разглашение полученного сообщения.
3.2. Отказ от факта получения некоторого сообщения когда в действительности оно было им получено.
3.3. Утверждение, что некоторое сообщение получено от отправителя когда в действительности предъявленное сообщение сфабриковано самим получателем.

Как правило, угроза работоспособности канала связи (угроза 1.4) наиболее эффективно достигается нарушением физической среды передачи данных (разрушение линий передачи и узлов обработки данных) или созданием помех ("глушение" радиосигнала, бомбардировка системы большим количеством ложных сообщений - "спам", и т.д.). Близко к ней находится угроза 1.3 - изъятие сообщения из канала связи. Эффективной защиты криптографическими средствами от этих угроз не существует, поэтому они обычно не рассматривается в работах по криптографии, проблема решается другими методами. Так, для устранения угрозы 1.3 обычно используется квитирование - высылка получателем отправителю квитанции (подтверждения) на полученное сообщение. Также в рамках криптографии отсутствует решение, которое бы устранило угрозы 2.1 и 3.1 - разглашение секретных данных одной из легальных сторон со "списыванием" этого на другую сторону или на ненадежность канала связи.

Оказалось, что сформулированная выше задача за вычетом угроз 1.3, 1.4, 2.1, 3.1 может быть разделена на три подзадачи, которые решаются независимо друг от друга и характеризуются собственными наборами угроз из приведенного списка:

- "классическая задача криптографии" - защита данных от разглашения и искажения при передаче по открытому каналу связи;
- "подпись электронного документа" - защита от отказа от авторства сообщения;
- "вручение заказного письма" - защита от отказа от факта получения сообщения; Ниже все три задачи рассмотрены с необходимой степенью подробности.

Защита передаваемых и хранимых секретных данных от разглашения и искажения

Это исторически первая и до сих пор наиболее важная задача криптографии, в ней учитываются угрозы 1.1 и 1.2 из приведенного выше списка. "Классическая" задача возникает, если создание и использование массивов данных разделены во времени и/или в пространстве, и на своей пространственно-временной "линии жизни" информация оказывается в зоне досягаемости злоумышленника. В первом случае говорят о защите данных при хранении, во втором - при передаче. При достаточной общности каждый из вариантов задачи имеет свои особенности, соответственно и методы решения также могут отличаться. В задаче присутствуют две легальные стороны:

- отправитель или источник сообщения (О, назовем его Олегом, чтобы избежать фраз типа "отправитель отправляет, а получатель получает");
- получатель или приемник сообщения (П, назовем его Петром);

Между отправителем и получателем сообщения есть взаимное доверие, поэтому в качестве злоумышленника может выступать только субъект, отличный от них обоих (З, назовем его Захаром). Олег отправляет Петру сообщение, при этом Захар может попытаться выполнить одного или нескольких следующих действий:

(1) чтение сообщений;
(2) внесение изменений в реальное переданное сообщение "на лету";
(3) создание нового сообщения и отправка его Петру от имени Олега;
(4) повторная передача ранее переданного сообщения;
(5) уничтожение переданного сообщения.

Каждое из перечисленных выше действий является **атакой** на наш информационный процесс. Возможности по доступу к каналу передачи, необходимые для их выполнения, различны, и в реальной ситуации может оказаться, что одни атаки осуществимы, а другие - нет. Для реализации атаки №1 необходим доступ к каналу передачи данных на чтение, для атаки №3 - на запись, для атаки №4 - на чтение и запись. Для осуществления атак №№2 и 5 необходим полный контроль над каналом, то есть возможность разорвать его и встроить туда собственные узлы обработки данных, или получить контроль над существующем узлом обработки. Какие из атак доступны злоумышленнику, зависит от конкретных условий протекания информационного процесса - от среды передачи данных, от аппаратуры, которой он располагает, и т.д.. Так, если средой передачи информации служит радиоэфир, осуществление атак №2 и частично №5 (если не рассматривать "глушение") невозможно, доступны только атаки №№1,3,4. При использовании оптоволоконной линии связи злоумышленнику может быть доступна только атака №1 - незаметно "врезаться" в оптоволоконную линию практически невозможно. Некоторые атаки из приведенного списка могут рассматриваться как последовательное осуществление других из того же списка. Так, атака №2 может рассматриваться как последовательное исполнение атак №№1,5,3, а атака № 4 - как исполнение атак №1 и 3, разнесенное по времени.

Подведем итог, постановка "классической" задачи криптографии следующая:

1. Законные стороны информационного процесса - отправитель и получатель сообщения. Задача первого отправить, а второго получить сообщение и понять его содержание.

2. Процесс считается нормально идущим, если к получателю придет именно то сообщение, которое отправил отправитель, и кроме него никто не сможет ознакомиться с его содержанием. Возможны следующие отклонения от его нормального течения:

- передаваемые данные станут известны кому либо еще помимо законного получателя;
- передаваемые данные будут искажены, то есть получатель получит не то или не в точности то, что отправлено отправителем.

4. Между отправителем и получателем есть взаимное доверие и ни один из них не осуществляет злоумышленных действий, злоумышленником является третья сторона, которая ставит перед собой цели ознакомиться с содержанием переданного сообщения или навязать получателю ложное сообщение, полностью сфабриковав его самостоятельно или исказив переданное отправителем сообщение. Злоумышленник имеет доступ к каналу связи и для осуществления своей цели может "слушать" канал или передавать в него свои данные. В наилучшей для себя ситуации злоумышленник может захватить полный контроль над каналом и ему станут доступны любые манипуляции с переданными данными.

Задача подтверждения авторства сообщения.

В этой задаче принимаются во внимание угрозы 2.2 и 3.3 из приведенного выше списка - между отправителем и получателем сообщения отсутствует взаимное доверие и возможно возникновение конфликта по поводу переданных данных. Каждый из них может совершать злоумышленные действия, направленные против другой стороны, и по этой причине в системе необходимо наличие инстанции, которая выполняет "арбитражные" функции, то есть в случае конфликта между абонентами решает, кто из них прав, а кто нет - эта сторона по вполне понятным причинам называется в криптографии "независимым арбитражем". Вместе с тем, злоумышленник как отдельный субъект информационного процесса здесь отсутствует. Опишем задачу по той же схеме:

1. Законные стороны информационного процесса - отправитель и получатель сообщения.

- отправитель или источник сообщения (О, Олег);
- получатель или приемник сообщения (П, Петр);
- независимый арбитр (А, Антон), разрешающий конфликт между Олегом и Петром в случае его возникновения.

Задача первого отправить, а второго получить сообщение и понять его содержание, задача последнего - вынести суждение о том, кто из двух предыдущих участников прав в случае возникновения конфликта между ними.

2. Процесс считается проходящим нормально, если Петр получит именно то сообщение, которое отправил Олег, и стороны не будут предъявлять претензий друг другу касательно переданных данных. Возможны следующие отклонения от нормального течения процесса:

- отправитель откажется от авторства переданного им сообщения;
- получатель будет утверждать, что некоторое сообщение получено им от отправителя, хотя в действительности тот его не передавал.

3. Между отправителем и получателем отсутствует взаимное доверие, каждый из них может осуществить злоумышленные действия в отношении другого: Олег может попытаться убедить Антона, что он не отправлял сообщения, которое в действительности отправил Петру.

Петр, в свою очередь, может попытаться убедить Антона в том, что он получил некоторое сообщение от Олега, хотя тот его в действительности не отправлял.

Вручение сообщения под расписку.

В этой задаче принимаются во внимание угрозы 2.3 и 3.2 из приведенного выше списка - между отправителем и получателем сообщения отсутствует взаимное доверие и возможно возникновение конфликта по поводу переданных данных. Каждый из них может совершать злоумышленные действия, направленные против другой стороны, злоумышленник как отдельный субъект информационного процесса здесь также отсутствует. Охарактеризуем задачу по нашей схеме:

1. Законные стороны информационного процесса - отправитель и получатель сообщения.
- отправитель или источник сообщения (О, Олег);
- получатель или приемник сообщения (П, Петр);

Задача первого отправить, а второго получить сообщение и понять его содержание.

2. Процесс считается проходящим нормально, если получатель ознакомится с содержанием полученного сообщения и стороны не будут предъявлять претензий друг другу касательно переданных данных. Возможны следующие отклонения от нормального течения процесса:
- отправитель будет утверждать, что передал получателю сообщение, хотя в действительности не отправлял его;
- получатель ознакомится с содержанием сообщения, но будет утверждать, что никакого сообщения не получал.

3. Между отправителем и получателем отсутствует взаимное доверие, каждый из них может осуществить злоумышленные действия в отношении другого, Петр может утверждать, что он не получал сообщения, которое в действительности получил и прочитал, а Олег, в свою очередь, может утверждать, что Петр прочитал сообщение, хотя в действительности он его не передавал Петру.

Решением этой задачи может являться такая схема информационного взаимодействия, которая группирует в одну транзакцию два следующих действия, не позволяя ни одному из них осуществиться без другого:
- Олег получает и читает сообщение;
- Петр получает подтверждение о том, что Олег получил сообщение.

Две следующие задачи касаются проблемы обмена ключевой информацией. Для защиты передаваемых по открытым каналам связи данных обычно применяется шифрование, одним из наиболее распространенных вариантов является использование симметричных шифров, то есть шифров с секретным ключом. В таких системах возникает проблема распределения ключевой информации, так как для ее передачи участникам информационного обмена нужен защищенный канал связи. В системах с большим числом абонентов эта проблема превращается в серьезную головную боль администраторов. Способов ее обойти всего два:
- использовать асимметричные алгоритмы шифрования, в которых для процедуры за- и расшифрования используются различные ключи и знание ключа зашифрования не позволяет определить соответствующий ключ расшифрования, поэтому он может быть несекретным и передаваться по открытым каналам связи;
- вырабатывать общий секретный ключ в ходе некоторого сеанса информационного взаимодействия по открытому каналу, организованного таким образом, чтобы ключ было невозможно выработать на основе только перехваченных в канале данных.

Первый подход получил название **асимметричного** или **двухключевого шифрования**, второй - **открытого распределения ключей.**

Вот, пожалуй и все наиболее популярные задачи практической криптографии. Конечно, есть и другие, но они или менее известны, или отсутствует их удовлетворительное решение, или это решение есть, но оно не получило заметного практического применения. Кратко перечислим наиболее известные из этих "теоретических" задач:

1. Синхронный обмен сообщениями. Требуется организовать обмен двух субъектов сообщениями таким образом, чтобы ни один из них, получив сообщение другой стороны, не смог отказаться от передачи своего и не смог сформировать свое сообщение в зависимости от сведений из сообщения другой стороны. В качестве дополнительного условия иногда требуется, чтобы передаваемые сообщения удовлетворяли определенным заранее критериям.

Как вариант предыдущей задачи - проблема "подписи контракта": есть два документа в электронной форме и есть процедура выработки цифровой подписи под документами. Требуется организовать обмен подписанными документами между двумя субъектами таким образом, чтобы ни один из них не смог отказаться передавать "свой" подписанный документ, получив подписанный документ от другой стороны. Очевидно, что это вариант предыдущей задачи, в котором "определенный критерий" - это корректность подписи документа, то есть соответствие подписи содержанию.

3. "Передача с забыванием" - организовать передачу сообщения одним субъектом другому таким образом, чтобы вероятность получения сообщения была ровно 0.5 и чтобы на эту вероятность никто не мог повлиять.

4. "Бросание монеты по телефону" - организовать такое взаимодействие не доверяющих друг другу субъектов через канал передачи данных, которое позволит выработать один бит информации (значение 0 или 1) таким образом, чтобы он был случайным, несмещенным (вероятность каждого исхода равна 0.5), чтобы на исход "бросания монеты" не мог повлиять никто извне ("третьи лица"), и чтобы в исходе "бросания" можно было убедить независимый арбитраж при возникновении у участников разногласий о результате.

5. "Сравнение с нулевым разглашением" или "проблема двух миллионеров" - два миллионера хотят узнать, кто из них богаче, но при этом никто из них не хочет сообщить другой стороне истинную величину своего состояния. В более общей постановке задача формулируется следующим образом: есть два субъекта, каждый из которых располагает некоторым элементом данных - соответственно **a** и **b**, и которые желают совместно вычислить значение некоторой согласованной функции **f(a,b)**. Требуется организовать процедуру вычисления таким образом, чтобы никто из субъектов не узнал значения параметра другого.

6. "**(n,k)**-пороговая схема" - имеется некоторый ресурс, например - зашифрованный набор данных (файл), также есть **n** субъектов. Необходимо построить процедуру, разрешающую доступ к ресурсу (дающую возможность расшифровать файл), только если его запросят одновременно не менее **k** субъектов.

7. "Тайное голосование по телефону" - имеется **n** субъектов, взаимодействующих по линиям связи, некоторый вопрос ставится им на голосование, каждый из субъектов может проголосовать либо "за", либо "против". Требуется организовать процедуру голосования таким образом, чтобы можно было вычислить ее исход - подсчитать, сколько подано голосов "за" или, в более простом случае, выяснить, что "за" было подано достаточное (большее или равное некоторой величине), или недостаточное (меньшее этой величины) число голосов, и чтобы при этом результаты голосования каждого из субъектов оставались в тайне.

Конечно, это далеко не все задачи, рассматриваемые криптографией. Но, как сказал классик, "никто не может объять необъятного", поэтому автор вынужден поставить точку в данном выпуске. Следующий выпуск будет посвящен алгоритмам шифрования.

3. КЛАССИФИКАЦИЯ ШИФРОВ

Симметричные алгоритмы представляют собой алгоритмы, в которых ключ шифрования может бать рассчитан по ключу дешифрирования и наоборот. В большинстве симметричных систем ключи шифрования и дешифрирования одни и те же. Эти алгоритмы также называют алгоритмами с секретным ключом или алгоритмами с одним ключом. Для работы такой системы требуется, чтобы отправитель и получатель согласовали используемый ключ перед началом безопасной передачи сообщения (имели защищенный канал для передачи ключа). Безопасность симметричного алгоритма определяется ключом, т.о. раскрытие ключа дает возможность злоумышленнику шифровать и дешифрировать все сообщения.

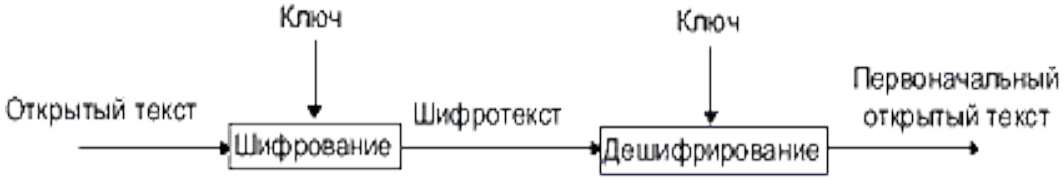


Рис. 3.1 Схема канала защищенной связи.

Из-за большой избыточности естественных языков в зашифрованное сообщение трудно внести осмысленные изменения, поэтому помимо защиты информации обеспечивается защита от навязывания ложных данных. Если же естественная избыточность недостаточна, то используется специальная контрольная комбинация - имитовставка.

Так как используется один ключ, то каждый из участников обмена может зашифровывать и дешифрировать сообщения, поэтому данная схема шифрования работает на взаимном доверии. Если его нет, то могут возникать различные коллизии, так как при возникновении какого-либо спора по поводу достоверности сообщения, независимый наблюдатель не может сказать кем из участников было отправлено сообщение.

Симметричные алгоритмы делятся на две категории. Одни из них обрабатывают текст побитно(иногда побайтно) и называются *потоковыми алгоритмами* или *потоковыми шифрами*. Те же, которые работают с группами битов открытого текста называются *блочными алгоритмами (шифрами)*.

Перестановочные шифры

Простой столбцевой перестановочный шифр

В данном виде шифра текст пишется на горизонтально разграфленном листе бумаги фиксированной ширины, а шифротекст считывается по вертикали. Дешифрирование заключается в записи шифротекста вертикально на листе разграфленной бумаги фиксированной ширины и затем считывании открытого текста горизонтально.

Открытый текст:

ВОЛОГОДСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ

```
В О Л О Г О
Д С К И Й
Г О С У Д А
Р С Т В Е Н
Н Ы Й   П Е
Д А Г О Г И
Ч Е С К И Й
  У Н И В Е
Р С И Т Е Т
```

Зашифрованный текст:

ВДГРНДЧ РОСОСЫАЕУСЛКСТЙГСНИОИУВ ОКИТГЙДЕПГИВЕО АНЕИЙЕТ

Перестановочный шифр с ключевым словом

Буквы открытого текста записываются в клетки прямоугольной таблицы по ее строчкам. Буквы ключевого слова пишутся над столбцами и указывают порядок этих столбцов (по возрастанию номеров букв в алфавите). Чтобы получить зашифрованный текст, надо выписывать буквы по столбцам с учетом их нумерации:

Открытый текст: Прикладная математика Ключ: Шифр

Ш	и	ф	р
4	1	3	2
П	р	и	к
л	а	д	н
а	я	м	а
т	е	м	а
т	и	к	а

Криптограмма: Раяеикнаааидммкплатт

Ключевое слово(последовательность столбцов) известно адресату, который легко сможет расшифровать сообщение.

Так как символы криптотекста те же, что и в открытом тексте, то частотный анализ покажет, что каждая буква встречается приблизительно с той же частотой, что и обычно. Это дает криптоаналитику информацию о

том, что это перестановочный шифр. Применение к криптотексту второго перестановочного фильтра значительно повысит безопасность. Существуют и еще более сложные перестановочные шифры, но с применением компьютера можно раскрыть почти все из них.

Хотя многие современные алгоритмы используют перестановку, с этим связана проблема использования большого объема памяти, а также иногда требуется работа с сообщениями определенного размера. Поэтому чаще используют подстановочные шифры.

Подстановочные шифры

В подстановочных шифрах буквы исходного сообщения заменяются на подстановки. Замены в криптотексте расположены в том же порядке, что и в оригинале. Если использование замен постоянно на протяжение всего текста, то криптосистема называется *одноалфавитной (моноалфавитной)*. В *многоалфавитных* системах использование подстановок меняется в различных частях текста.

Шифр Цезаря

Юлий Цезарь повествует о посылке зашифрованного сообщения Цицерону. Используемая при этом система подстановок была одноалфавитной, но не являлась системой Цезаря: латинские буквы заменялись на греческие способом, который не был ясен из рассказа Цезаря. Информация о том, что Цезарь действительно использовал систем у Цезаря, пришла от Светония.

В шифре Цезаря каждая буква замещается на букву, находящуюся **k** символами правее по модулю равному количеству букв в алфавите. (Согласно Светонию у Цезаря k=3 n=50)

$C_k(j)=(j+k)(\mathrm{mod}\ n)$, n - количество букв в алфавите

Очевидно, что обратной подстановкой является

$C_k^{-1}(j)=C_{n-k}=(j+n-k)(\mathrm{mod}\ n)$

Шифр Цезаря с ключевым словом

В данной разновидности шифра Цезаря ключ задается числом **k (0<=k<=n-1)** и коротким ключевым словом или предложением. Выписывается алфавит, а под ним, начиная с **k**-й позиции, ключевое слово. Оставшиеся буквы записываются в алфавитном порядке после ключевого слова. В итоге мы получаем подстановку для каждой буквы. Требование, чтобы все буквы ключевого слова были различными не обязательно - можно записывать ключевое слово без повторения одинаковых букв. Количество ключей в системе Цезаря с ключевым словом равно **n!**.

Многоалфавитные системы

Полиалфавитные подстановочные шифры были изобретены Лином Баттистой (Lean Battista) в 1568 году. Основная идея многоалфавитных систем состоит в том, что на протяжении всего текста одна и та же буква может быть зашифрована по-разному. Т.е. замены для буквы выбираются *из многих алфавитов* в зависимости от положения в тексте. Это является хорошей защитой от простого подсчета частот, так как не существует единой маскировки для каждой буквы в криптотексте. В данных шифрах используются множественные однобуквенные ключи, каждый из которых используется для шифрования одного символа открытого текста. Первым ключом шифруется первый символ открытого текста, вторым - второй, и т.д. После использования всех ключей они повторяются циклически.

Шифр Вернама

Шифр Вернама, или *одноразовый блокнот*, был изобретен в 1917 году Мейджором Джозефом Моборном (Major Joseph Mauborn) и Гильбертом Вернамом (Gilbert Vernam) из AT&T (American Telephone & Telegraph). В классическом понимании одноразовый блокнот является большой неповторяющейся последовательностью символов ключа, распределенных случайным образом. Первоначально это была одноразовая лента для телетайпов. Отправитель использовал каждый символ ключа для шифрования только одного символа открытого текста. Шифрование представляет собой сложение по модулю **n** (мощность алфавита) символа открытого текста и символа ключа из одноразового блокнота. Каждый символ ключа используется только один раз и для единственного сообщения, иначе даже если использовать блокнот размером в несколько гигабайт, при получении криптоаналитиком нескольких текстов с перекрывающимися ключами он сможет восстановить исходный текст. Он сдвинет каждую пару шифротекстов относительно друг друга и подсчитает число совпадений в каждой позиции. если шифротексты смещены правильно, соотношение совпадений резко возрастет. С этой точки зрения криптоанализ не составит труда. Если же ключ не повторяется и случаен, то криптоаналитик, перехватывает он тексты или нет, всегда имеет одинаковые знания. Случайная ключевая последовательность, сложенная с неслучайным открытым текстом, дает совершенно случайный криптотекст, и никакие вычислительные мощности не смогут это изменить.

Рис. 3.1 схема шифра Вернама.

В реальных системах сначала подготавливают две одинаковые ленты со случайными цифрами ключа. Одна остается у отправителя, а другая передается "неперехватываемым" образом например, курьером с охраной, законному получателю. Когда отправитель хочет передать сообщение, он сначала преобразует его в двоичную форму и помещает в устройство, которое к каждой цифре сообщения прибавляет по модулю два цифры, считанные с ключевой ленты. На принимающей стороне кодированное сообщение записывается и пропускается через машину, похожую на устройство, использованное для шифрования, которое к каждой двоичной цифре сообщения прибавляет (вычитает, так как сложение и вычитание по модулю два эквивалентны) по модулю два цифры, считанные с ключевой ленты, получая таким образом открытый текст. При этом, естественно, ключевая лента должна продвигаться абсолютно синхронно со своим дубликатом, используемым для зашифрования.

Главным недостатком данной системы является то, что для каждого бита переданной информации должен быть заранее подготовлен бит ключевой информации, причем эти биты должны быть случайными. При шифровании большого объема данных это является серьезным ограничением. Поэтому данная система используется только для передачи сообщений наивысшей секретности. По слухам "горячая линия" между США и СССР шифровалась с помощью одноразового блокнота. Многие сообщения советских шпионов были зашифрованы с использованием одноразовых блокнотов. Эти сообщения нераскрыты сегодня, и не будут раскрыты никогда (если не найдется способа вернуться в прошлое и достать эти блокноты :)

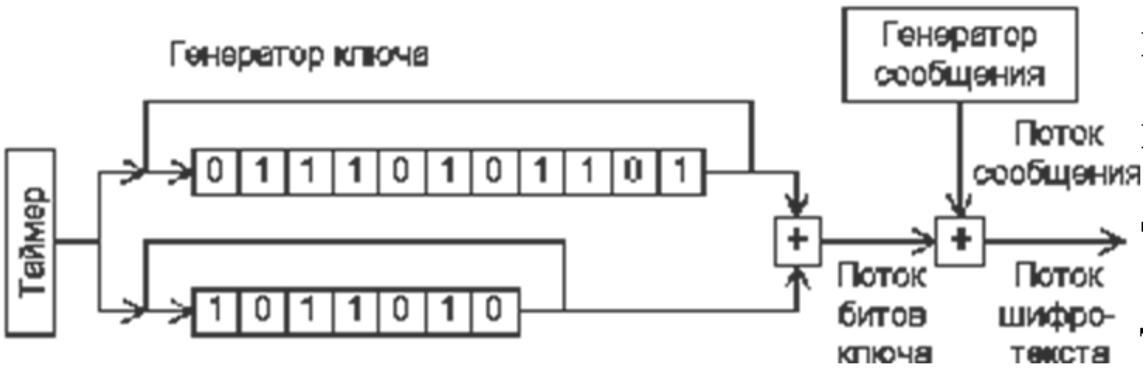


Рис. 3.2 Шифр гаммирования псевдослучайной последовательносью.

Чтобы обойти проблему предварительной передачи секретного ключа большого объема, инженеры и изобретатели придумали много остроумных схем генерации очень длинных потоков псевдослучайных цифр из нескольких коротких потоков в соответствии с некоторым алгоритмом. Получателя шифрованного сообщения при этом необходимо снабдить точно таким же генератором, как и у отправителя. Но такие алгоритмы добавляющих регулярности в шифротекст, обнаружение которых может помочь аналитику дешифровать сообщение. Один из основных методов построения подобных генераторов заключается в использовании двух или более битовых лент, считанные с которых данные побитно складываются для получения "смешанного" потока. Например, простая одноразовая лента может быть заменена двумя циклическими лентами, длины которых являются простыми или взаимно простыми числами. Так как в этом случае длины лент не имеют общих множителей, полученный из них поток имеет период повторения, равный произведению их длин: две ленты, имеющие длину 1000 и 1001 соответственно, дают в результате составной поток с периодом

1000x1001=1001000 цифр. Ленты циркулируют через сумматор, который складывает по модулю два считанные с них цифры. Выход сумматора служит ключом, используемым для зашифрования сообщения. Поэтому важно, чтобы составной поток превышал по длине все вместе взятые сообщения, которые могут быть переданы за разумный период времени. Поскольку побитовый сумматор является линейным устройством, он изначально криптографически слаб, но может быть усилен большим количеством различных способов. Другой способ - указание местонахождения ключа как места в книге, например, *Дональд Э. Кнут Искусство Программирования Том 2. Получисленные алгоритмы. Третье издание. стр 83, 3-й абзац.* Все символы, входящие в алфавит, начиная с этого места используются как одноразовый ключ для какого-либо сообщения. Но в данном случае *ключ не будет случайным* и может быть использована информация о частотах распределения букв.

Как не удивительно, но класс шифров Вернама - единственный класс шифров, для которого может быть доказана (и была доказана Шенноном) невскрываемость в абсолютном смысле этого термина.

Шифр Виженера

Одной из старейших и наиболее известных многоалфавитных криптосистем является система Виженера, названная в честь французского криптографа Блейза Виженера(Vigenere), в *М.Н.Аршинов, Л.Е.Садовский Коды и математика* данный шифр назван шифром Тритемиуса. Этот метод был впервые опубликован в 1586 году. В данном шифре ключ задается набором из **d** букв. Такие наборы подписываются с повторением под сообщением, а, затем, полученную последовательность складывают с открытым текстом по модулю **n**(мощность алфавита). Т.е. получается следующая формула:$Vig_d(m_i)=(m_i+k_{i \bmod d})(\bmod n)$

Также букву шифротекста можно находить из следующей таблицы, как пересечение столбца, определяемого буквой открытого текста, и строки, определяемой буквой ключа:

В частном случае, при **d=1**, получаем шифр Цезаря. обратная подстановка легко определяется из квадрата, или по формулам

$Vig_d^{-1}(m_i)=(m_i-k_{i \bmod d})(\bmod n)$ и $Bof_d^{-1}(m_i)=Bof_d(m_i)=(k_i-m_{i \bmod d})(\bmod n)$

соответственно.

Повторное применение двух или более шифров Виженера будет называться *составным шифром Виженера*. Он имеет уравнение

$Vig^*(m_i)=(m_i + k_{i \bmod dk} + l_{i \bmod dl} + ... + s_{i \bmod ds}) (\bmod n)$

где $k_i + l_i + ... + s_i$ вообще говоря, имеют различные периоды dk, dl, ..., ds соответственно. Период их суммы $k_i + l_i + ... + s_i$ будет наименьшим общим кратным отдельных периодов.

Если ключ k не повторяется, то получится шифр Вернама. Если в качестве ключа используется текст, имеющий смысл, то имеем *шифр "бегущего ключа"*.

Табл. 1 – Таблица Виженера для русского алфавита

	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Шифр Виженера с перемешанным один раз алфавитом.

Такой шифр представляет собой простую подстановку с последующим применением шифра Виженера:

$Vig^{\wedge}(m_i)=f(m_i) + k_{i \bmod d}$, $Vig^{\wedge -1}(m_i)=f^{-1}(m_i - k_{i \bmod d})$.

Шифр с автоключом

Дальнейшей модификацией системы Виженера является система шифров с *автоключом(auto-key)*, приписываемая математику XVIв. Дж. Кардано, AUTOCLAVE. Шифрование начинается с помощью "первичного ключа" (который является настоящим ключом в нашем смысле) и продолжается с помощью сообщения или криптограммы, смещенной на длину первичного ключа, затем производится сложение по модулю, равному мощности алфавита. Например:

Сообщение	П Р И В Е Т П Р И М А Т У
Первичный ключ	В Г П У
Автоключ	П Р И В Е Т П Р И
Шифротекст	С У Ч Х Ф В Ч Т Н Ю П В Ы

Легальная расшифровка не представляет труда: по первичному ключу получается начало сообщения, после чего найденная часть исходного сообщения используется в качестве ключа. В другом варианте данной системы в качестве ключа служит текст сообщения, зашифрованный с помощью ключа по системе Виженера. Но данный криптоалгоритм слабее оригинального.

Методы анализа многоалфавитных систем

Если ключ повторяется периодически и период известен, то криптоанализ данных систем может быть сведен к криптоанализу одноалфавитных систем. Пусть период равен 5. Буквы упорядочиваются по столбцам следующим образом:

0

1 2 3 4 5

.. ..

Два появления одной буквы в одном столбце представляют одну букву сообщения. Поэтому можно расшифровать каждый столбец простым подсчетом частот.

Классификация методов дешифрования. Модель предполагаемого противника. Правила Керкхоффа.

Фундаментальное правило криптоанализа, впервые сформулированное голландцем А.Керкхоффом еще в XIX веке заключается в том, что стойкость шифра (криптосистемы) должна определяться только секретностью ключа. Иными словами, правило Керкхоффа состоит в том, что весь алгоритм шифрования, кроме значения секретного ключа, известен криптоаналитику противника. Это обусловлено тем, что криптосистема, реализующая семейство криптографических преобразований, обычно рассматривается как открытая система. Такой подход отражает очень важный принцип технологии защиты информации: защищенность системы не должна зависеть от секретности чего-либо такого, что невозможно быстро изменить в случае утечки секретной информации.

Обычно криптосистема представляет собой совокупность аппаратных и программных средств, которую можно изменить только при значительных затратах времени и средств, тогда как ключ является легко изменяемым объектом. Именно поэтому стойкость криптосистемы определяется только секретностью ключа.

Другое почти общепринятое допущение состоит в том, что криптоаналитик имеет в своем распоряжении шифртексты сообщений. Существует четыре основных типа криптоаналитических атак 10.2.4 . Конечно, все они формулируются в предположении, что криптоаналитику известны применяемый алгоритм шифрования и шифртексты сообщений.

1. Криптоаналитическая атака при наличии только известного шифртекста. Криптоаналитик имеет только шифртексты C1, C2, …, Ci нескольких сообщений, причем все они зашифрованы с использованием одного и того же алгоритма шифрования Ek. Работа криптоаналитика заключается в том, чтобы раскрыть исходные тексты M1, M2, …, Mi по возможности большинства сообщений или, еще лучше, вычислить ключ K, использованный для шифрования этих сообщений, с тем, чтобы расшифровать и другие сообщения, зашифрованные этим шифром.
Этот вариант соответствует модели внешнего нарушителя, который имеет физический доступ к линии связи, но не имеет доступ к аппаратуре шифрования и дешифрования.

2. Криптоаналитическая атака при наличии известного открытого текста. Криптоаналитик имеет доступ не только к шифртекстам C1, C2, …, Ci и нескольких сообщений, но также к открытым текстам M1, M2, …, Mi этих сообщений. Его работа заключается в нахождении ключа K, используемого при шифровании этих сообщений, или алгоритма расшифрования Dk любых новых сообщений, зашифрованных тем же ключом. причем все они зашифрованы с использованием одного и того же алгоритма шифрования Ek.

Возможность проведения такой атаки складывается при шифровании стандартных документов, подготавливаемых по стандартным формам, когда определенные болки данных повторяются и известны. Он также применим при использовании режима глобального шифрования, когда вся информация на встроенном магнитном носителе записывается в виде шифртекста, включая главную корневую запись, загрузочный сектор, системные программы и пр. При хищении этого носителя (или компьютера) легко установить, какая часть криптограммы соответствует системной информации и получить большой объем известного исходного текста для выполнения криптоанализа.

3. Криптоаналитическая атака при возможности выбора открытого текста. Криптоаналитик не только имеет доступ к шифртекстам C1, C2, …, Ci и связанным с ними открытым текстам M1, M2, …, Mi этих сообщений, но и может по желанию выбирать открытые тексты, которые затем получает в зашифрованном виде. Такой криптоанализ получается более мощным по сравнению с криптоанализом с известным открытым текстом, потому что криптоаналитик может выбрать для шифрования такие блоки открытого текста, которые дадут больше

информации о ключе. Работа криптоаналитика состоит в поиске ключа K, использованного для шифрования сообщений, или алгоритма рашифрования Dk новых сообщений, зашифрованных тем же ключом.
Этот вариант атаки соответствует модели внутреннего нарушителя. На практике такая ситуация может возникнуть при вовлечении в криптоатаку лиц, которые не знают секретного ключа, но в силу своих служебных полномочий имеют возможность использовать шифрование для передачи своих сообщений.

4. Криптоаналитическая атака с адаптивным выбором открытого текста. Это - особый вариант атаки с выбором открытого текста. Криптоаналитик может не только выбирать открытый текст, который затем шифруется, но и изменять свой выбор в зависимости от результатов предыдущего шифрования. При криптоанализе с простым выбором открытого текста криптоаналитик обычно может выбирать несколько крупных блоков открытого текста для их шифрования; при криптоанализе с адаптивным выбором открытого текста он имеет возможность выбрать сначала более мелкий пробный блок открытого текста, затем выбрать следующий блок в зависимости от результатов первого выбора, и т.д. Эта атака предоставляет криптоаналитику еще больше возможностей, чем предыдущие типы атак.

Кроме перечисленных основных типов криптоаналитических атак, можно отметить, по крайней мере, еще два типа.

5. Криптоаналитическая атака с использованием выбранного шифртекста. Криптоаналитик может выбирать для расшифрования различные шифртексты и имеет доступ к расшифрованным открытым текстам . Например, криптоаналитик получил досту к защищенному от несанкционированного вскрытия блоку, который выполняет автоматическое расшифрование. Работа криптоаналитика заключается в нахождении ключа. Этот тип криптоанализа представляет особый интерес для раскрытия алгоритмов с открытым ключом.

6. Криптоаналитическая атака методом полного перебора всех возможных ключей. Эта атака предполагает использование криптоаналитикомизвестного шифртекста и осуществляется посредством полного перебора всех возможных ключей с проверкой, является ли осмысленным получающийся открытый текст. Такой подход требует привлечения предельных вычислительных ресурсов и иногда называется силовой атакой.

Предположим, что имеется конечное число возможных сообщений $M_1,...,M_n$ с априорными вероятностями $P(M_1),...,P(M_n)$ и что эти сообщения преобразуются в возможные криптограммы $E_1,...,E_m$, так что

$E = T_i M$.

После того как шифровальщик противника перехватил некоторую криптограмму E, он может вычислить, по крайней мере в принципе, апостериорные вероятности различных сообщений $P_E(M)$. Естественно определить *совершенную секретность* с помощью следующего условия: для всех E апостериорные вероятности равны априорным вероятностям независимо от величины этих последних. В этом случае перехват сообщения не дает шифровальщику противника никакой информации. Теперь он не может корректировать никакие свои действия в зависимости от информации, содержащейся в криптограмме, так как все вероятности, относящиеся к содержанию криптограммы, не изменяются. С другой стороны, если это условие равенства вероятностей не выполнено, то имеются такие случаи, в которых для определенного ключа и определенных выборов сообщений апостериорные вероятности противника отличаются от априорных. А это в свою очередь может повлиять на выбор противником своих действий и, таким образом, совершенной секретности не получится. Следовательно, приведенное определение неизбежным образом следует из нашего интуитивного представления о совершенной секретности.

Необходимое и достаточное условие для того, чтобы система была совершенно секретной, можно записать в следующем виде. По теореме Байеса

$$P_E(M) = \frac{P(M) \cdot P_M(E)}{P(E)},$$

где

$P(M)$ – априорная вероятность сообщения M;

$P_M(E)$ – условная вероятность криптограммы E при условии, что выбрано сообщение M, т.е. сумма вероятностей всех тех ключей, которые переводят сообщение M в криптограмму E;

$P(E)$ – вероятность получения криптограммы E;

$P_E(M)$ – апостериорная вероятность сообщения M при условии, что перехвачена криптограмма E.

Для совершенной секретности системы величины $P_E(M)$ и $P(M)$ должны быть равны для всех E и M. Следовательно, должно быть выполнено одно из равенств: или $P(M) = 0$ [это решение должно быть отброшено, так как требуется, чтобы равенство осуществлялось при любых значениях $P(M)$], или же

$P_M(E) = P(E)$

для любых M и E. Наоборот, если $P_M(E) = P(E)$, то

$P_E(M) = P(M)$,

и система совершенно секретна. Таким образом, можно сформулировать следующее:

Теорема 6. *Необходимое и достаточное условие для совершенной секретности состоит в том, что*

$P_M(E) = P(E)$

для всех M и E, т.е. $P_M(E)$ не должно зависеть от M.

Другими словами, полная вероятность всех ключей, переводящих сообщение M_i в данную криптограмму E, равна полной вероятности всех ключей, переводящих сообщение M_j в ту же самую криптограмму E для всех M_i, M_j и E.

Далее, должно существовать по крайней мере столько же криптограмм E, сколько и сообщений M, так как для фиксированного i отображение T_i дает взаимнооднозначное соответствие между всеми M и некоторыми из E. Для совершенно секретных систем для каждого из этих E и любого M $P_M(E) = P(E) \neq 0$. Следовательно, найдется по крайней мере один ключ, отображающий данное M в любое из E. Но все ключи, отображающие фиксированное M в различные E, должны быть различными, и *поэтому число различных ключей не меньше числа сообщений M*. Как показывает следующий пример, можно получить совершенную секретность, когда число сообщений точно равно числу ключей. Пусть M_i занумерованы числами от 1 до n, так же как и E_i, и пусть используются n ключей. Тогда

$T_i M_j = E_s$,

где $s = i + j \pmod n$. В этом случае оказывается справедливым равенство $P_E(M) = 1/n = P(E)$ и система является совершенно секретной. Один пример такой системы показан на рис. 5, где

$s = i + j - 1 \pmod 5$.

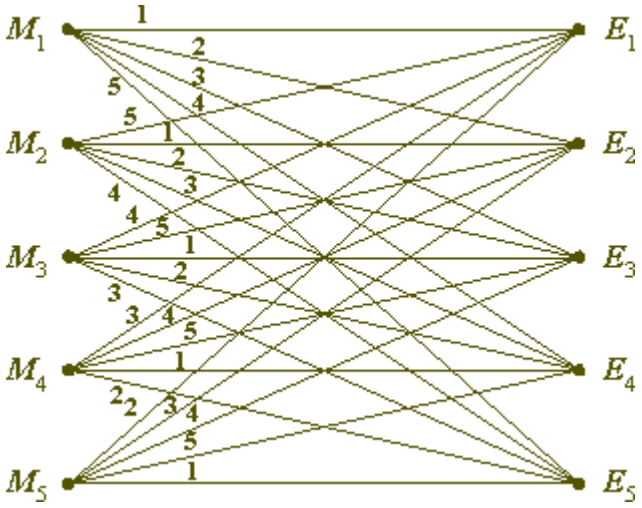


Рис 5. Совершенная система.

Совершенно секретные системы, в которых число криптограмм равно числу сообщений, а также числу ключей, характеризуются следующими двумя свойствами: 1) каждое ***M*** связывается с каждым ***E*** только одной линией; 2) все ключи равновероятны. Таким образом, матричное представление такой системы является "латинским квадратом".

В "Математической теории связи" показано, что количественно информацию удобно измерять с помощью энтропии. Если имеется некоторая совокупность возможностей с вероятностями $p_1,...,p_n$, то энтропия дается выражением

$$H = -\sum p_i \log p_i$$

Секретная система включает в себя два статистических выбора: выбор сообщения и выбор ключа. Можно измерять количество информации, создаваемой при выборе сообщения, через $H(M)$

$$H(M) = -\sum P(M) \log P(M)$$

где суммирование выполняется по всем возможным сообщениям. Аналогично, неопределенность, связанная с выбором ключа, дается выражением

$$H(K) = -\sum P(K) \log P(K)$$

В совершенно секретных системах описанного выше типа количество информации в сообщении равно самое большее $\log n$ (эта величина достигается для равновероятных сообщений). Эта информация может быть скрыта полностью лишь тогда, когда неопределенность ключа не меньше $\log n$. Это является первым примером общего принципа, который будет часто встречаться ниже: существует предел, которого нельзя превзойти при заданной неопределенности ключа – количество неопределенности, которое может быть введено в решение, не может быть больше, чем неопределенность ключа.

Положение несколько усложняется, если число сообщений бесконечно. Предположим, например, что сообщения порождаются соответствующим марковским процессом в виде бесконечной последовательности букв. Ясно, что никакой конечный ключ не даст совершенной секретности. Предположим тогда, что источник ключа порождает ключ аналогичным образом, т.е. как бесконечную последовательность символов.

Предположим далее, что для зашифрования и расшифрования сообщения длины L_M требуется только определенная длина ключа L_K. Пусть логарифм числа букв в алфавите сообщений будет R_M, а такой же логарифм для ключа – R_K. Тогда из рассуждений для конечного случая, очевидно, следует, что для совершенной секретности требуется, чтобы выполнялось неравенство

$R_M L_M \leqslant R_K L_K$.

Такой вид совершенной секретности реализован в системе Вернама.

Эти выводы делаются в предположении, что априорные вероятности сообщений неизвестны или произвольны. В этом случае ключ, требуемый для того, чтобы имела место совершенная секретность, зависит от полного числа возможных сообщений.

Можно было бы ожидать, что если в пространстве сообщений имеются фиксированные известные статистические связи, так что имеется определенная скорость создания сообщений R в смысле, принятом в "Математической теории связи", то необходимый объем ключа можно было бы снизить в среднем в R/R_M раз, и это действительно верно. В самом деле, сообщение можно пропустить через преобразователь, который устраняет избыточность и уменьшает среднюю длину сообщения как раз во столько раз. Затем к результату можно применить шифр Вернама. Очевидно, что объем ключа, используемого на букву сообщения, статистически уменьшается на множитель R/R_M, и в этом случае источник ключа и источник сообщений в точности согласован – один бит ключа полностью скрывает один бит информации сообщения. С помощью методов, использованных в "Математической теории связи", легко также показать, что это лучшее, чего можно достигнуть.

Совершенно секретные системы могут применяться и на практике, их можно использовать или в том случае, когда полной секретности придается чрезвычайно большое значение, например, для кодирования документов высших военных инстанций управления, или же в случаях, где число возможных сообщений мало. Так, беря крайний пример, когда имеются в виду только два сообщения – "да" или "нет", – можно, конечно, использовать совершенно секретную систему со следующей таблицей отображений:

M	***K***	
	A	***B***
да	0	1
нет	1	0

Недостатком совершенно секретных систем для случая корреспонденции большого объема является, конечно, то, что требуется посылать эквивалентный объем ключа. В следующих разделах будет рассмотрен вопрос о том, чего можно достигнуть при помощи меньших объемов ключа, в частности, с помощью конечного ключа.

Подделка ключа - наиболее слабое место в криптографии с открытым ключом. Злоумышленник может изменить пользовательскую связку ключей или подделать открытый ключ пользователя и посылать его другим для загрузки и использования. Например, предположим, что Хлой хочет отслеживать сообщения, которые Элис посылает Блэйку. Она могла бы использовать атаку, называемую *человек в середине (man in the middle)*. Для этого Хлой создает новую пару ключей. Она заменяет копию открытого ключа Блэйка, принадлежащую Элис, новым открытым ключом. Затем она перехватывает сообщения, которые Элис посылает Блэйку. Каждое перехваченное сообщение она расшифровывает, используя новый секретный ключ, зашифровывает настоящим

открытым ключом Блэйка и направляет их ему. Все сообщения, направленные Элис Блэйку, теперь могут быть прочитаны Хлой.

Правильное управление ключами является решающим фактором для обеспечения целостности не только Ваших связок ключей, но и связок ключей других пользователей. Основой управления ключами в GnuPG является подписание ключей. Подписание ключей имеет два основных применения: это позволяет Вам обнаруживать вмешательство в Вашу связку ключей, а также позволяет удостоверять, что ключ действительно принадлежит человеку, чьим идентификатором пользователя он помечен. Подписи на ключах также используются в схеме известной как *сеть доверия (web of trust)*, которая расширяет допустимые ключи не только подписанными лично Вами, но и подписанными людьми, которым Вы доверяете. Аккуратные пользователи, правильно реализовавшие управление ключами, могут исключить подмену ключей как вид атаки на конфиденциальность связи при помощи GnuPG.

Управление Вашей парой ключей

Пара ключей состоит из открытого и секретного ключей. Открытый ключ состоит из открытой части главного подписывающего ключа, открытых частей подчиненных подписывающих и шифрующих ключей, набора идентификаторов пользователя, ассоциирующих открытый ключ с реальным человеком. Каждая часть содержит данные о себе. Для ключа это его идентификатор, дата создания, срок действия и т.д. Для идентификатора пользователя это имя человека, дополнительный комментарий и адрес email. Структура секретного ключа аналогична, но содержит секретные части ключей и не содержит информацию об идентификаторе пользователя.

Для просмотра пары ключей используется команда `--edit-key`. Например,

```
chloe% gpg --edit-key chloe@cyb.org
Secret key is available.

pub  1024D/26B6AAE1  created: 1999-06-15 expires: never      trust: -/u
sub  2048g/0CF8CB7A  created: 1999-06-15 expires: never
sub  1792G/08224617  created: 1999-06-15 expires: 2002-06-14
sub   960D/B1F423E7  created: 1999-06-15 expires: 2002-06-14
(1)  Chloe (Jester) <chloe@cyb.org>
(2)  Chloe (Plebian) <chloe@tel.net>
Command>
```

При отображении открытого ключа выводится информация о том, доступен секретный ключ или нет. Затем выводится информация о каждом компоненте открытого ключа. Первая колонка показывает тип ключа. Символы `pub` указывают открытую часть главного подписывающего ключа, а символы `sub` открытую часть подчиненных ключей. Вторая колонка показывает длину ключа в битах, тип и идентификатор. Тип `D` это ключ DSA, `g` - ключ ElGamal, пригодный только для шифрования, и `G` - ключ ElGamal, который может использоваться и для подписи и для шифрования. Дата создания и окончания срока действия показана в колонках три и четыре. Идентификаторы пользователя перечислены после ключей.

Дополнительная информация о ключе может быть получена различными командами. Команда **toggle** переключает между открытой и закрытой компонентами пары ключей, если обе из них доступны.

```
Command> toggle

sec  1024D/26B6AAE1  created: 1999-06-15 expires: never
sbb  2048g/0CF8CB7A  created: 1999-06-15 expires: never
sbb  1792G/08224617  created: 1999-06-15 expires: 2002-06-14
sbb   960D/B1F423E7  created: 1999-06-15 expires: 2002-06-14
(1)  Chloe (Jester) <chloe@cyb.org>
(2)  Chloe (Plebian) <chloe@tel.net>
```

Представленная информация подобна выводимой для открытого ключа. Символы `sec` указывают секретный главный подписывающий ключ, символы `sbb` указывают секретные подчиненные ключи. Также выводятся идентификаторы пользователя из открытого ключа.

Интерактив 2 часа (фильма + обсуждение)

4. КРИПТОСИСТЕМЫ

Устройство шифров

Блочные шифры оперируют с блоками открытого текста. К ним предъявляются следующие требования:

- достаточная криптостойкость;
- простота процедур зашифрования и расшифрования;
- приемлимая надежность.

Под криптостойкостью понимают время, необходимое для раскрытия шифра при использовании наилучшего метода криптоанализа. Надежность - доля информации, дешифруемая при помощи какого-то криптоаналитического алгоритма. Само преобразование шифра должно использовать следующие принципы (по К. Шеннону):

- ***Рассеивание (diffusion)*** - т.е изменение любого знака открытого текста или ключа влияет на большое число знаков шифротекста, что скрывает статистические свойства открытого текста;
- ***Перемешивание (confusion)*** - использование преобразований, затрудняющих получение статистических зависимостей между шифротектстом и открытым текстом.

Практически все современные блочные шифры являются *композиционными* - т.е состоят из композиции простых преобразований или $F=F_1 \square F_2 o F_3 o F_4 o .. o F_n$, где **F**-преобразование шифра, F_i-простое преобразование, называемое также i-ым *циклом шифрования*. Само по себе преобразование может и не обеспечивать нужных свойств, но их цепочка позволяет получить необходимый результат. Например, стандарт DES состоит из 16 циклов. В иностранной литературе такие шифры часто называют послойными (layered). Если же используется одно и то же преобразование, т.е. F_i постоянно для ∀ i, то такой композиционный шифр называют *итерационным* шифром.

Наибольшую популярность имеют шифры, устроенные по принципу "шифра Фейстеля (Файстеля - Feistel)" (петли Фейстеля, сети Файстеля), т.е в которых:

1. входной блок для каждого преобразования разбивается на две половины: **p=(l,r)**, где **l**-левая, **r**-правая;
2. используется преобразование вида $F_i (l, r)=(r, l \square f_i (r))$, где f_i - зависящая от ключа K_i функция, а $\square$ - операция XOR или некая другая.

. Функция f_i называется *цикловой функцией*, а ключ K_i, используемый для получения функции f_i называется *цикловым ключом*. Как можно заметить, с цикловой функцией складывается только левая половина, а правая остается неизменной. Затем обе половины меняются местами. Это преобразование прокручивается несколько раз (несколько циклов) и выходом шифра является получившаяся в конце пара **(l,r)**
Графически все выглядит следующим образом:

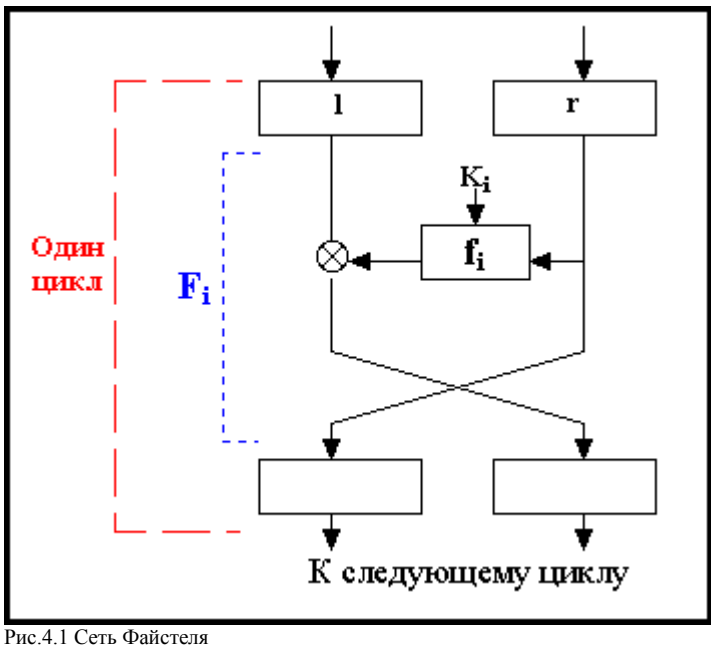


Рис.4.1 Сеть Файстеля

В качестве функции f_i выступает некая комбинация перестановок, подстановок, сдвигов, добавлений ключа и прочих преобразований. Так, при использовании подстановок информация проходит через специальные блоки, называемые *S-блоками (S-боксами, S-boxes)*, в которых значение группы битов заменяется на другое значение. По такому принципу (с небольшими отличиями) построены многие алгоритмы: DES, FEAL, серия LOKI и т.д.

В других алгоритмах используются несколько иные принципы. Так, например, алгоритмы, построенные по *SP-принципу* (*SP-сети*) осуществляют преобразование, пропуская блок через последовательность подстановок (*Substitutions*) и перстановок (*Permutations*). Отсюда и название - *SP-сети*, т.е. сети "подстановок-перстановок". Примером такого алгоритма является очень перспективная разработка Rijndael. Возможно применение в алгоритмах и каких-либо новых конструкций, но как правило, они несут в себе оперделенные ошибки (пример - FROG, HPC). Но все перечисленные алгоритмы являются композиционными. Саму идею построения криптографически стойкой системы путем последовательного применения относительно простых криптографических преобразований была высказана Шенноном (идея многократного шифрования).

Размеры блоков в каждом алгоритме свои. DES использует блоки по 64 бита (две половинки по 32 бита), LOKI97 - 128 бит. При размере выходных блоков до 8 бит шифр можно считать поточным.

Получение цикловых ключей.

Ключ имеет фиксированную длину. Однако при прокрутке хотя бы 8 циклов шифрования с размером блока, скажем, 128 бит даже при простом прибавлении посредством XOR потребуется 8*128=1024 бита ключа, поскольку нельзя добавлять в каждом цикле одно и то же значение - это ослабляет шифр. Посему для получения последовательности ключевых бит придумывают специальный алгоритм выработки

цикловых ключей (ключевое расписание - key schedule). В результате работы этого алгоритма из исходных бит ключа шифрования получается массив бит определенной длины, из которого по определенным правилам составляются цикловые ключи. Каждый шифр имеет свой алгоритм выработки цикловых ключей.
Чтобы использовать алгоритмы блочного шифрования для различных криптографических задач существует несколько режимов их работы. Наиболее часто встречающимися в практике являются следующие режимы:

- электронная кодовая книга - ECB (Electronic Code Book);
- сцепление блоков шифротекста - CBC (Cipher Block Chaining);
- обратная связь по шифротектсту - CFB (Cipher Feed Back);
- обратная связь по выходу - OFB (Output Feed Back);

Обозначим применение шифра к блоку открытого текста как $\mathbf{E_k(M)=C}$, где **k** - ключ, **M** - блок открытого текста, а **C** - получающийся шифротекст.

Электронная Кодовая Книга (ECB)

Исходный текст разбивается на блоки, равные размеру блока шифра. Затем с каждый блок шифруют независимо от других с использованием одного ключа шифрования. Графически это выглядит так:

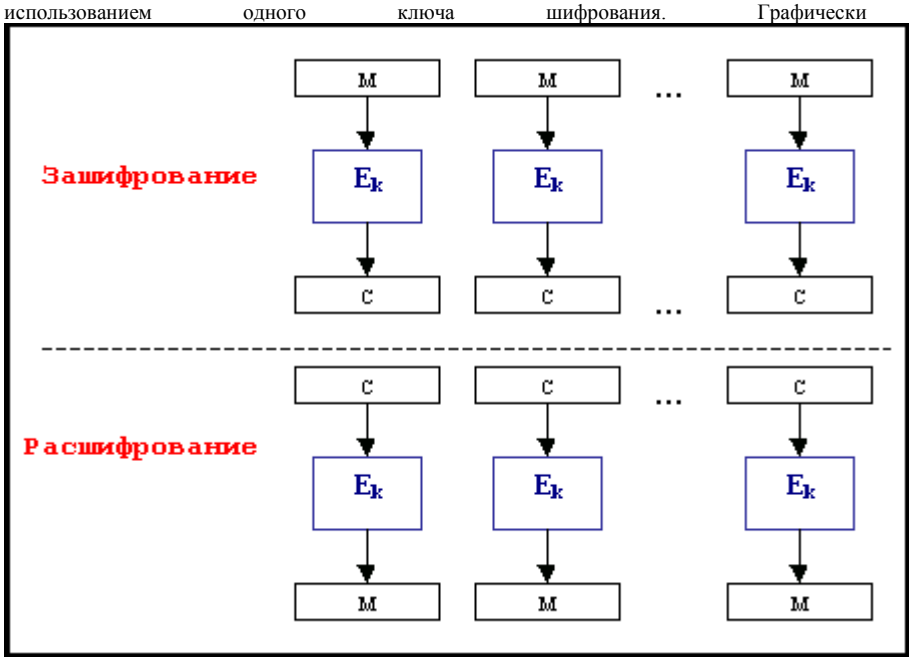


Рис. 4.2. Режим ECB.

Непосредственно этот режим применяется для шифрования небольших объемов информации, размером не более одного блока или для шифрования ключей. Это связано с тем, что одинаковые блоки открытого текста преобразуются в одинаковые блоки шифротекста, что может дать взломщику (криптоаналитику) определенную информацию о содержании сообщения. К тому же, если он предполагает наличие определенных слов в сообщении (например, слово "Здравствуйте" в начале сообщения или "До свидания" в конце), то получается, что он обладает как фрагментом открытого текста, так и соответствующего шифротекста, что может сильно облегчить задачу нахождения ключа.

Основным достоинством этого режима является простота реализации.

Сцепление блоков шифротекста (CBC)

Один из наиболее часто применимых режимов шифрования для обработки больших количеств информации. Исходный текст разбивается на блоки, а затем обрабатывается по следующей схеме:

1. Первый блок складывается побитно по модулю 2 (XOR) с неким значением IV - начальным вектором (Init Vector), который выбирается независимо перед началом шифрования.
2. Полученное значение шифруется.
3. Полученный в результате блок шифротекста отправляется получателю и одновременно служит начальным вектором IV для следующего блока открытого текста.

Расшифрование осуществляется в обратном порядке. Графически схема выглядит следующим образом:

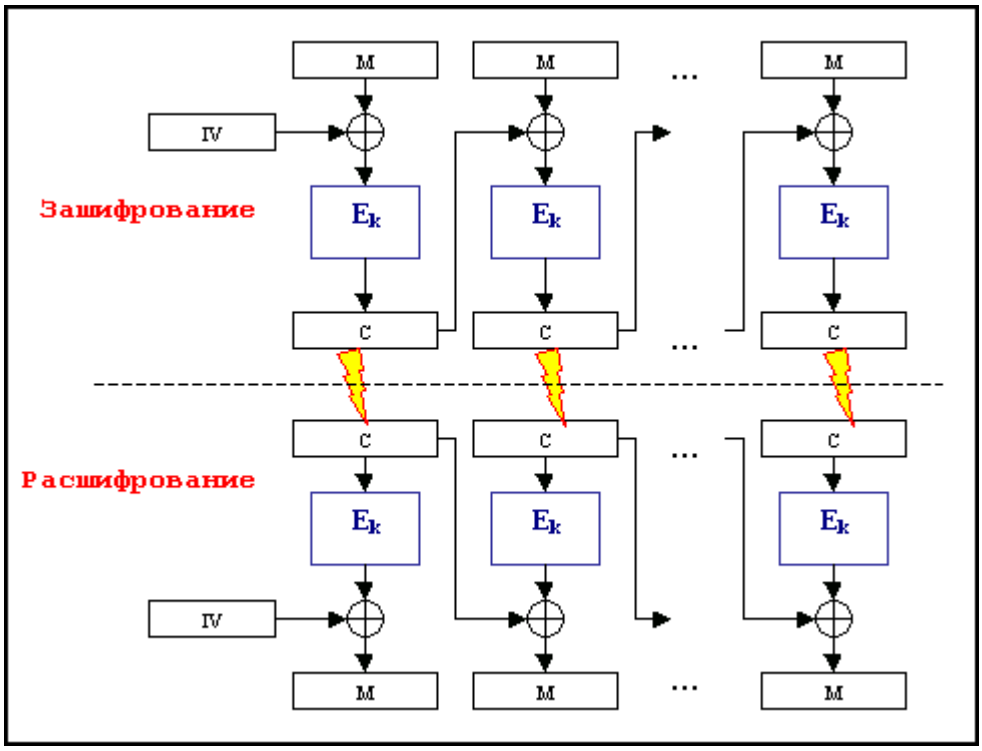


Рис. 4.3. Режим CBC.

В виде формулы, преобразование в режиме CBC можно представить как $\mathbf{C_i=E_k(M_i \oplus C_{i-1})}$, где **i** - номер соответствующего блока. Из-за использования такого сцепления блоков шифротекста с открытым текстом пропадают указанные выше недостатки режима ECB, поскольку каждый последующий блок зависит от всех предыдущих. Если во время передачи один из блоков шифротекста исказится (передастся с ошибкой), то получатель сможет корректно расшифровать последующие блоки сообщения. Проблемы возникнут только с этим "бракованным" и следующим блоками. Одним из важных свойств этого режима является "распространение ошибки" - изменение блока открытого текста меняет все последующие блоки шифротекста. Поскольку последний блок шифротекста зависит от всех блоков открытого текста, то его можно использовать для контроля целостности и аутентичности (проверки подлинности) сообщения. Его называют *кодом аутентификации сообщения (MAC - Message Authentication Code)*. Он может защитить как от случайных, так и преднамеренных изменений в сообщениях.

Обратная связь по шифротексту (CFB

Режим может использоваться для получения из поточного шифра из блочного. Размер блока в данном режиме меньше либо равен размеру блока шифра. Схема данного режима:

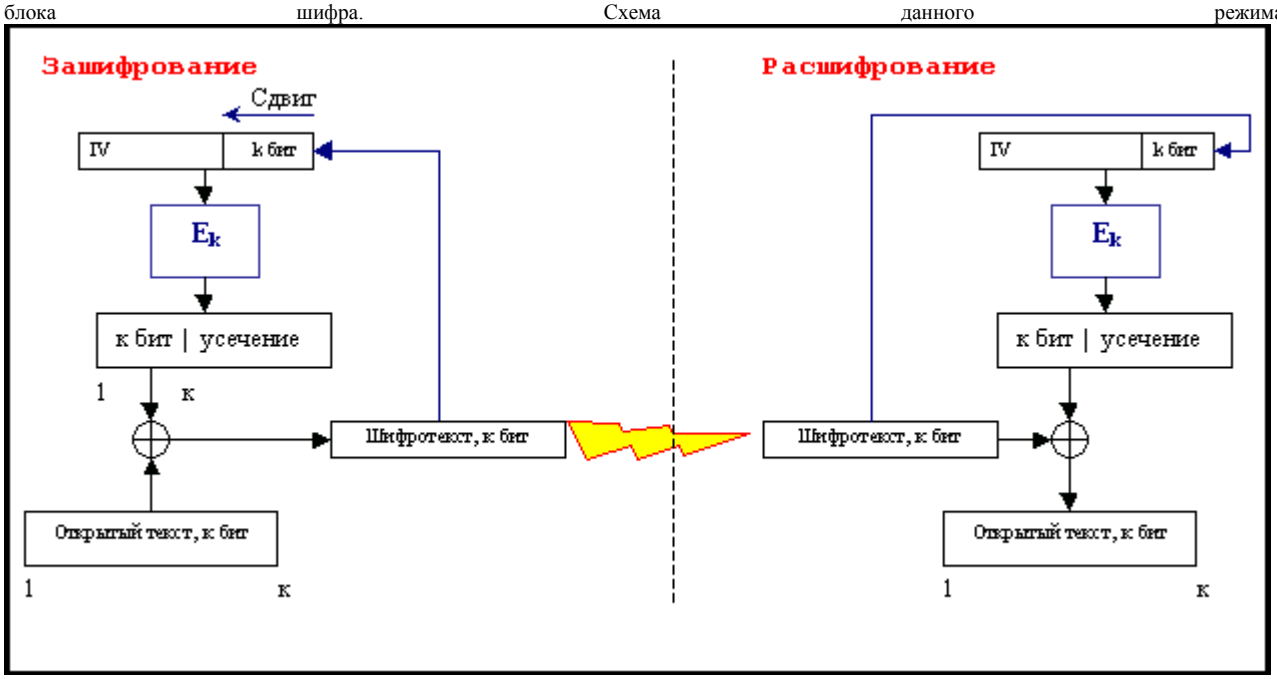


Рис. 4.3. Режим CBF

1. IV представляет собой сдвиговый регистр. Вначале IV заполняется неким значением, которое называется синхропосылкой, не является секретным и передается перед сеансом связи получателю.
2. Значение IV шифруется.
3. Берутся первые k бит зашифрованного значения IV и складываются (XOR) с k битами открытого текста получается блок шифротекста из k бит.
4. Значение IV сдвигается на k битов влево, а вместо него становится значение ш.т.
5. Затем опять 2 пункт и т.д до конца.

Расшифрование аналогично.

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

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

Обратная связь по выходу (OFB)

Данный режим примечателен тем, что позволяет получать поточный шифр в его классическом виде (см ПОТОЧНЫЕ ШИФРЫ), в отличии от режима CFB, в котором присутствует связь с шифротекстом. Принцип работы схож с принципом работы режима CFB, но сдвиговый регистр IV заполняется не битами шифротекста, а битами, выходящими из под усечения. Вот его схема:

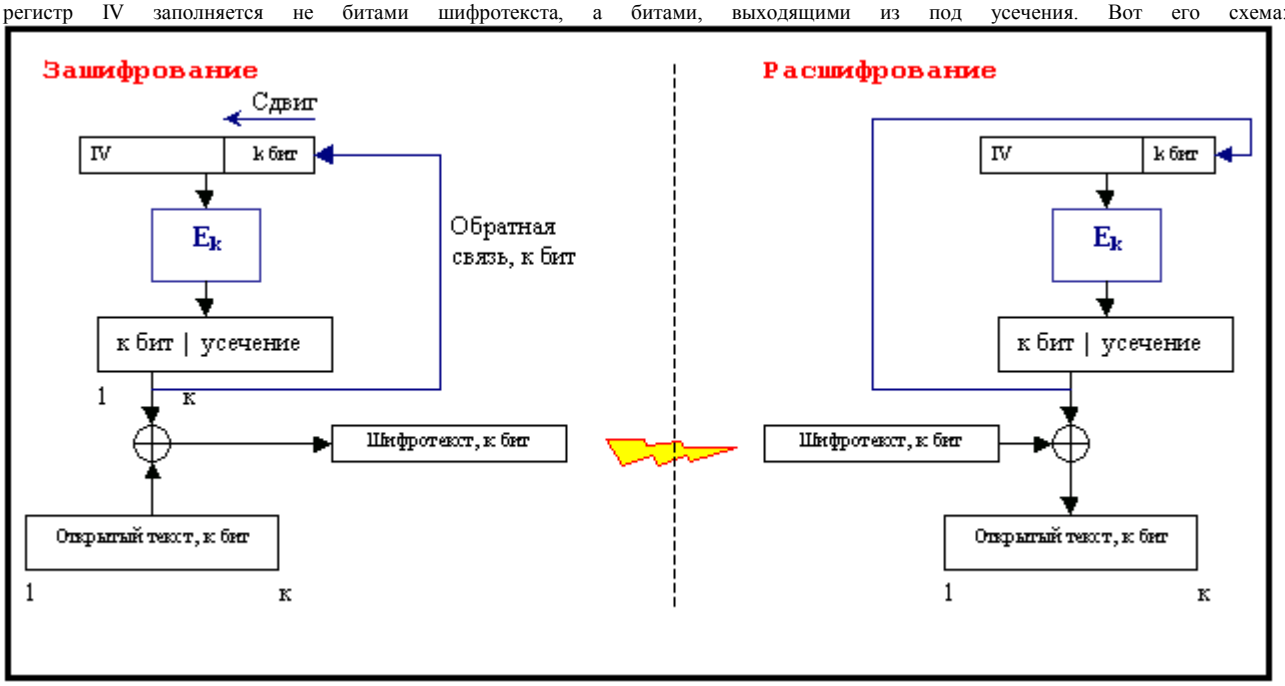


Рис. 4.2. Режим OFB.

Расшифрование осуществляется аналогично. Т.е. для любого блока длины k операция зашифрования выглядит следующим образом: $\mathbf{C_i=M_i \oplus G_i}$, где $\mathbf{G_i}$ - результат зашифрования некоторого вектора, являющегося заполнением сдвигового регистра. Главное свойство шифра - единичные ошибки не распространяются, т.к заполнение сдвигового регистра осуществляется не зависимо от шифротекста. Область применения: потоки видео, аудио или данных, для которых необходимо обеспечить оперативную доставку. Широко используется у военных наряду с поточными шифрами.

Аутентификация сообщений с помощью блочных шифров.

Настал черед еще нескольких определений:

Аутентификация (authentication) - проверка подлинности чего(или кого-)-либо. Может быть аутентификация пользователя, сообщения и т.д. Необходимо отличать ее от следующего понятия: ***Идентификация (identification)*** - некое описательное представление какого-то субъекта. Так, если кто-то заявляет, что он - Вася Иванов, то он идентифицирует себя как "Вася Иванов". Но проверить, так ли это на самом деле (провести аутентификацию) мы можем только при помощи его паспорта.

Итак, как же можно проверить подлинность сообщения с помощью блочного шифра? Довольно просто.

1. Отправитель А хочет отправить некое сообщение $(a_1,..,a_t)$. Он зашифровывает его на секретном ключе, который знает только он и получатель, в режиме CBC или CFB это сообщение, а затем из получившегося шифротекста берет последний блок b_t из κ бит (при этом κ должно быть достаточно большим).
2. Отправитель А посылает сообщение $(a_1, .. , a_t , b_t)$ получателю в открытом виде или зашифровав его на другом ключе.
3. Получатель В, получив сообщение, $(a_1, .. , a_t , b_t)$, зашифровывает $(a_1,..,a_t)$ в том же режиме, что и А (должна быть договоренность) на том же секретном ключе (который знает только он и А).
4. Сравнивая полученный результат с b_t он удостоверяется, что сообщение отправил А, что оно не было подделано на узле связи (в случае передачи в открытом виде).

В данной схеме b_t является *кодом аутентификации сообщения (MAC)*. Для российского стандарта шифрования процесс получения кода аутентификации называется работой в режиме *имитовставки*.

Некоторые комментарии:

Режим шифрования должен быть обязательно с распространением ошибок (т.е CBC или CFB). Необходимо использовать шифр с достаточной длиной блока, а то может появиться ситуация, когда из-за небольшого числа используемых бит для аутентификации возможно подменить исходное сообщение и при знании ключа получить тот же самый результат. Также очевидно, что схема опирается на то, что оба абонента имеют один и тот же секретный ключ, который получили заранее.

Поточные шифры

Устройство шифров

Шифрование в поточных шифрах осуществляется на основе сложения некоторой *ключевой последовательности* (*гаммы*) с открытым текстом сообщения. Сложение осуществляется познаково посредством XOR. Уравнение зашифрования выглядит следующим образом:

$c_i = m_i \oplus k_i$ для i=1,2,3..

где c_i - знак шифротекста, m_i - знак открытого текста, k_i - знак ключевой последовательности. Расшифрование выглядит так:

$m_i = c_i \oplus k_i$ для i=1,2,3... В качестве знаков могут выступать как отдельные биты, так и символы (байты). Таким образом, поточные шифры подходят для шифрования непрерывных потоков данных - голоса, видео и т.д. В общем виде схему шифра можно изобразить следующим образом:

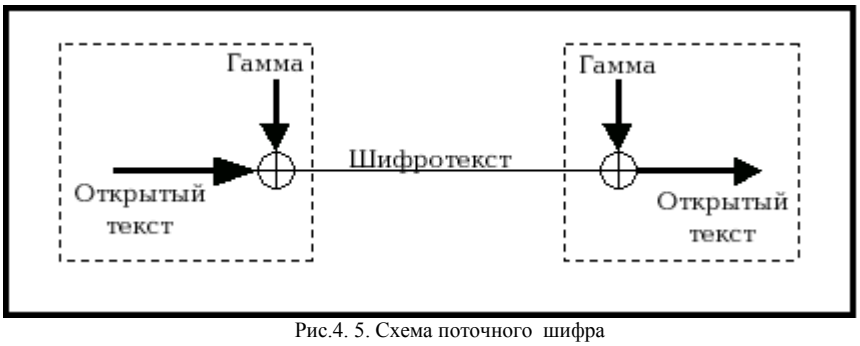


Рис.4. 5. Схема поточного шифра

Можно сказать, что шифрование осуществляется наложением *гаммы*(шифрование *гаммированием*). А сама гамма является ключом шифрования. Но иметь ключ, равный по размеру шифруемым данным представляется проблематичным. Поэтому поточные шифры и вырабатывают выходную гамму на основе некоторого секретного ключа небольшого размера, а значит основной задачей поточных шифров является выработка некоторой последовательности (*выходной гаммы*) для шифрования. Т.е. выходная гамма является ключевым потоком для сообщения. Поточные шифры классифицируют следующим образом:

- *синхронные*;
- *самосинхронизирующиеся (асинхронные)*.

Синхронные поточные шифры - ключевой поток (выходная гамма) получается независимо от исходного и шифрованного текстов. В данном случае наша иллюстрация меняется в следующую:

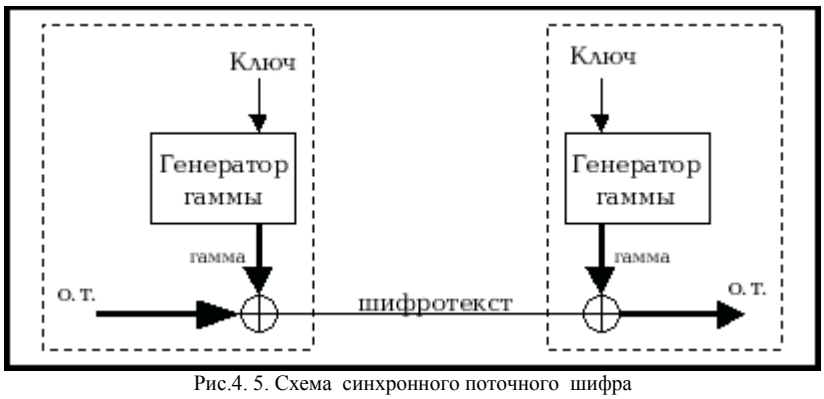


Рис.4. 5. Схема синхронного поточного шифра

Шифр вырабатывает гамму на основе секретного ключа, она складывается с открытым текстом и результат посылается другому абоненту и расшифровывается аналогично. Блок, вырабатывающий гамму называется *генератором гаммы* или *псевдослучайным генератором (гаммы)* - *PRG(Pseudo Random Generator)*. Любой блочный шифр в режиме OFB представляет собой синхронный поточный шифр.

Очевидно, что на вырабатываемую последовательность накладываются требования. Ведь если в ней есть большие последовательности нулей, то получается, что в линию при передаче передается текст сообщения с открытом виде. Или если последовательность из единиц - тот же эффект (т.к. ничто не мешает противнику попробовать "протянуть" несколько подряд идущих единиц в качестве гаммы для перехваченного сообщения. Результат - кусок открытого текста у противника). Поэтому наилучшей гаммой является случайная последовательность. Однако нельзя независимо друг от друга сгенерировать 2 одинаковые случайные последовательности.

Генераторы гаммы вырабатывают так называемые *псевдослучайные последовательности*, которые зависят от ключа шифрования, но тем не менее по своим характеристикам сходны со случайными. Задача ставится таким образом, чтобы при наличии определнного количества битов последовательности нельзя было предсказать следующие биты. Помимо этого 1 и 0 на входе должны быть равновероятны (т.е. вероятность появления 1 = вер-ти появления 0 = 0.5). Для достижения этого последовательности исследуются статистическими тестами.

Вообще, генерирование непредсказуемых псевдослучайных последовательностей является одной из важных криптографических задач на сегодняшний день (и проблем). Придумано множество генераторов, статистических тестов. Исследование псевдослучайных последовательностей мы рассмотрим далее.

Синхронные поточные шифры обладают следующими свойствами:

- *требования по синхронизации*. При использовании синхронных поточных шифров получатель и отправитель должны быть *синхронизированы* - т.е. вырабатывать одинаковые значения ключевого потока для соответствующих знаков передаваемого потока данных. Если синхронизация нарушится (например, вследствие потери знака при передаче), процесс расшифрования не даст корректного результата.
- *отсутствие размножения ошибок*. Изменение знака шифртекста при передаче не вызывает ошбок при расшифровании других знаков шифртекста.
- *свойство активной атаки*. Как следствие первого свойства, любая вставка или удаление символа в шифртекст активным противником приводит к нарушению синхронизации и обнаруживается получателем, расшифровывающим сообщение. Как следствие второго свойства, активный противник может изменять символы шифртекста и эти изменения приведут к соответствующим изменениям в открытом тексте, получаемом при расшифровании. Посему необходимы дополнительные механизмы, позволяющие предотвратить это.

Рассмотрим одну из ***атак*** на такие шифры.
$O_1O_2O_3$... - знаки открытого текста.
$K_1K_2K_3$... - знаки ключевой последовательности.
$C_1C_2C_3$... - знаки шифротекста, полученные следующим образом:

$$\begin{array}{l} O_1\ O_2\ O_3\ O_4\ \ldots \\ \oplus\ \underline{K_1\ K_2\ K_3\ K_4\ \ldots} \\ C_1\ C_2\ C_3\ C_4\ \ldots \end{array}$$

Допустим, что при повторном шифровании на том же ключе произошла вставка одного знака O':

$$\begin{array}{l} O_1\ O'\ O_2\ O_3\ \ldots \\ \oplus\ \underline{K_1\ K_2\ K_3\ K_4\ \ldots} \\ C_1\ C'_2\ C'_3\ C'_4\ \ldots \end{array}$$

Криптоаналитик противника перехватил обе последовательности $C_1C_2C_3C_4$ и $C_1C'_2C'_3C'_4$. Составив затем уравнения: $K_2 = C'_2 \oplus O'$; $O_2 = C_2 \oplus K_2$; $K_3 = C'_3 \oplus O_2 \oplus O_3 = C_3 \oplus K_3$ и т.д., и подобрав значение одного знака O' он сможет прочитать сообщение после этого знака. Поскольку в результате исследований у него будет фрагмент ключевой последовательности (гаммы), то он может попытаться восстановить всю гамму и получить сообщение целиком (если вэтом есть необходимость, конечно). Отсюда можно сделать вывод, что ***нельзя дважды использовать один и тот же ключ***.

Хотя описание атаки носит гипотетический характер, тем не менее она очень и очень реальна, Ведь в качестве вставленного знака с таким же успехом может выступать и последовательность знаков. В этом случае подбираться будет последний знак вставленной последовательности. Представим себе ситуацию шифрования 2 файлов одним ключом, причем файлы имеют одинаковый заголовок и, скажем, середину (довольно реальная ситуация!!!). Проведя описанную атаку, криптоаналитик получит либо полностью исходные файлы, либо их фрагменты, что может иметь непоправимые последствия.

Даже если 2 абсолютно различных текста шифруются на одном ключе, противник может вычислить сумму знаков шифртекстов $C_i^1 \oplus C_i^2 = O_i^1 \oplus K_i \oplus K_i \oplus O_i^2 = O_i^1 \oplus O_i^2$, где C_i^1- i-ый знак первого шифртекста, C_i^2 - i-ый знак второго, O_i^1 и O_i^2 - знаки открытых текстов соответственно. Сумма открытых текстов отнюдь не случайна и противник сможет восстановить 2 сообщения.

Самосинхронизирующиеся поточные шифры - каждый знак ключевого потока определяется фиксированным числом предшествующих знаков шифротекста. Схематично это можно изобразить так:

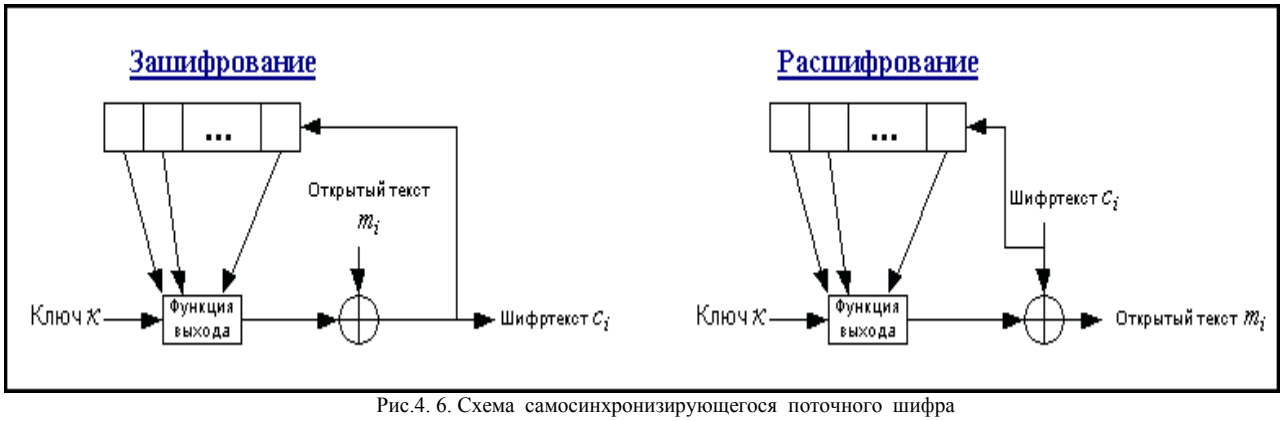


Рис.4. 6. Схема самосинхронизирующегося поточного шифра

Данному типу шифров соответствуют блочные шифры, работающие в режиме CFB.

Самосинхронизирующиеся поточные шифры обладают следующими свойствами:

- *самосинхронизация*. Самосинхронизация существует при удалении или вставке некоторых знаков шифртекста, поскольку процесс расшифрования зависит от некоторого фиксированного числа предшествующих знаков шифртектста. Это означает, что в случае удаления знака из шифртекста сначала будут ошибки при расшифровании, а затем все станет хорошо и ошибок не будет.
- *ограниченное размножение ошибок*. Предположим, что состояние шифра зависит от t предидущих знаков шифртекста. Если во время передачи один знак шифротекста был изменен или удален/вставлен, то при расшифровке будет искажено не более t знаков, после которых пойдет опять нормальный текст.
- *свойство активной атаки*. Из второго свойства следует, что любое изменение знаков шифртекста активным противником приведет к тому, что несколько знаков шифротекста расшифруются неправильно и это с большей (по сравнению с синхронными шифрами) веротностью будет замечено со стороны получателя, расшифровывающего сообщение. Однако в случае вставки или удаления знаков шифртекста (по св-ву 1) это намного труднее обнаружить (по сравнению с синхронными шифрами - там получается рассинхронизация). Поэтому необходимы дополнительные механизмы для контроля этой ситуации.
- *рассеивание статистики открытого текста*. Поскольку каждый знак открытого текста влияет на весь последующий шифртекст, статистические свойства открытого текста (ведь он далеко не случаен) не сохраняются в шифртексте.

Для исследования свойств поточных шифров очень часто используется теория конечных автоматов.

Алгоритм DES

Принципы разработки

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

DES является классической *сетью Фейштеля* с двумя ветвями. Данные шифруются 64-битными блоками, используя 56-битный ключ. Алгоритм преобразует за несколько *раундов* 64-битный вход в 64-битный выход. Длина ключа равна 56 битам. Процесс шифрования состоит из четырех этапов. На первом из них выполняется начальная перестановка (IP) 64-битного исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей. Следующий этап состоит из 16 *раундов* одной и той же функции, которая использует операции сдвига и подстановки. На третьем этапе левая и правая половины выхода последней (16-й) итерации меняются местами. Наконец, на четвертом этапе выполняется перестановка IP^{-1} результата, полученного на третьем этапе. Перестановка IP^{-1} инверсна начальной перестановке.

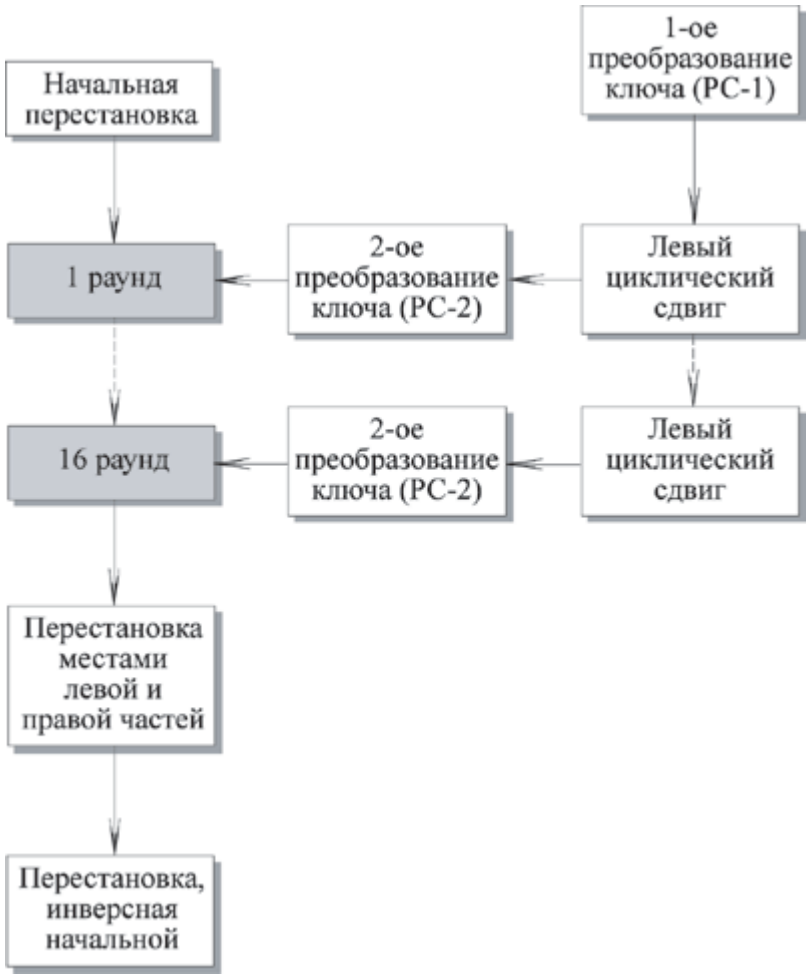


Рис.4.7 Общая схема DES

Справа на рисунке показан способ, которым используется 56-битный ключ. Первоначально ключ подается на вход функции перестановки. Затем для каждого из 16 *раундов подключ* K_i является комбинацией левого циклического сдвига и перестановки. Функция перестановки одна и та же для каждого *раунда*, но *подключи* K_i для каждого *раунда* получаются разные вследствие повторяющегося сдвига битов ключа.

Шифрование

Начальная перестановка

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

Последовательность преобразований отдельного раунда

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

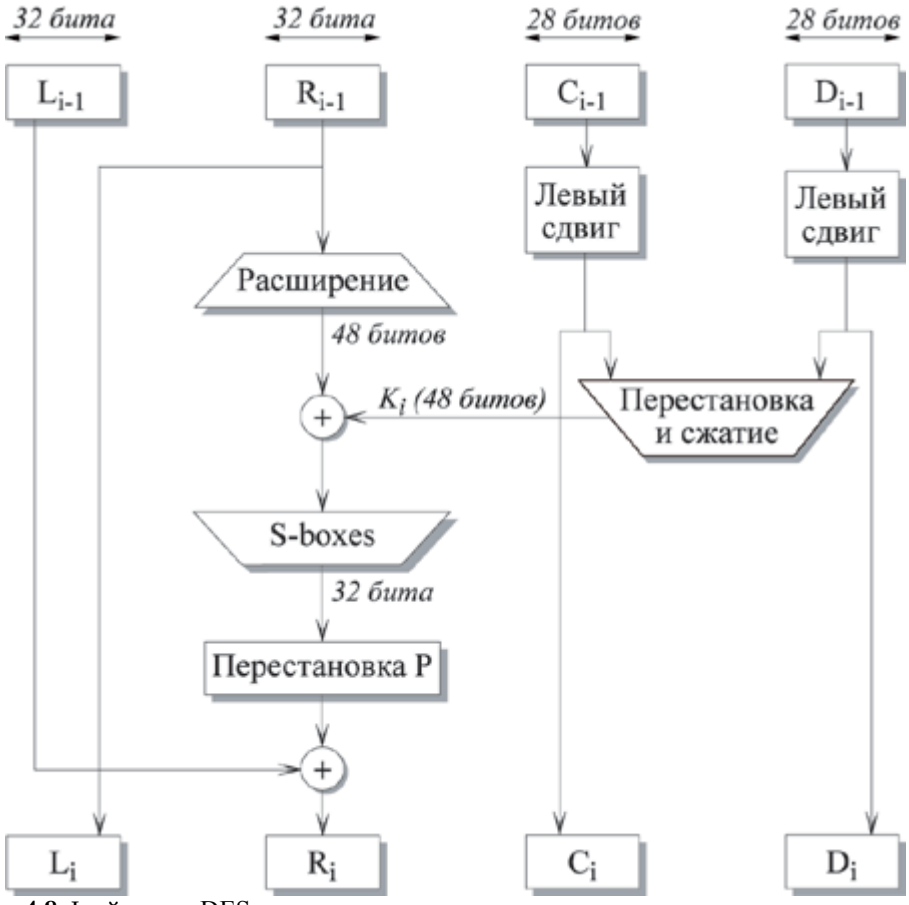


Рис. 4.8 I-ый раунд DES

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

$L_i = R_{i-1}$

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

Где $\oplus$ обозначает операцию XOR.

Таким образом, 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.

Рассмотрим функцию F более подробно.

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

. . . efgh ijkl mnop . . .

то в результате расширения получается сообщение

. . . defghi hijklm lmnopq . . .

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

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

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

Создание подключей

Ключ для отдельного *раунда* K_i состоит из 48 битов. Ключи K_i получаются по следующему алгоритму. Для 56-битного ключа, используемого на входе алгоритма, вначале выполняется перестановка в соответствии с таблицей Permuted Choice 1 (PC-1). Полученный 56-битный ключ разделяется на две 28-битные части, обозначаемые как C_0 и D_0 соответственно. На каждом *раунде* C_i и D_i независимо циклически сдвигаются влево на 1 или 2 бита, в зависимости от номера *раунда*. Полученные значения являются входом следующего *раунда*.

Они также представляют собой вход в Permuted Choice 2 (PC-2), который создает 48-битное выходное значение, являющееся входом функции $F(R_{i-1}, K_i)$.

Дешифрование

Процесс дешифрования аналогичен процессу шифрования. На входе алгоритма используется зашифрованный текст, но ключи K_i используются в обратной последовательности. K_{16} используется на первом *раунде*, K_1 используется на последнем *раунде*. Пусть выходом i-ого *раунда* шифрования будет $L_i||R_i$. Тогда соответствующий вход (16-i)-ого *раунда* дешифрования будет $R_i||L_i$.

После последнего *раунда* процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки IP^{-1} был $R_{16}||L_{16}$. Выходом этой стадии является незашифрованный текст.

Проверим корректность процесса дешифрования. Возьмем зашифрованный текст и ключ и используем их в качестве входа в алгоритм. На первом шаге выполним начальную перестановку IP и получим 64-битное значение $L^d_0||R^d_0$. Известно, что IP и IP^{-1} взаимнообратны. Следовательно

$L^d_0||R^d_0 = IP$ (зашифрованный текст)

Зашифрованный текст = $IP^{-1}(R_{16}||L_{16})$

$L^d_0||R^d_0 = IP(IP^{-1}(R_{16}||L_{16})) = R_{16}||L_{16}$

Таким образом, вход первого *раунда* процесса дешифрования эквивалентен 32-битному выходу 16-ого *раунда* процесса шифрования, у которого левая и правая части записаны в обратном порядке.

Теперь мы должны показать, что выход первого *раунда* процесса дешифрования эквивалентен 32-битному входу 16-ого *раунда* процесса шифрования. Во-первых, рассмотрим процесс шифрования. Мы видим,что

$L_{16} = R_{15}$

$R_{16} = L_{15} \oplus F(R_{15}, K_{16})$

При дешифровании:

$L^d_1 = R^d_0 = L_{16} = R_{15}$

$R^d_1 = L^d_0 \oplus F(R^d_0, K_{16}) =$

$= R_{16} \oplus F(R^d_0, K_{16}) =$

$= (L_{15} \oplus F(R_{15}, K_{16})) \oplus F(R_{15}, K_{16})$

XOR имеет следующие свойства:

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$

$D \oplus D = 0$

$E \oplus 0 = E$

Таким образом, мы имеем $L^d_1 = R_{15}$ и $R^d_1 = L_{15}$. Следовательно, выход первого *раунда* процесса дешифрования есть $L_{15}||R_{15}$, который является перестановкой входа 16-го *раунда* шифрования. Легко показать, что данное соответствие выполняется все 16 *раундов*. Мы можем описать этот процесс в общих терминах. Для i-ого *раунда* шифрующего алгоритма:

$L_i = R_{i-1}$

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

Эти равенства можно записать по-другому:

$R_{i-1} = L_i$

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

Таким образом, мы описали входы i-ого *раунда* как функцию выходов.

Выход последней стадии процесса дешифрования есть $R_0||L_0$. Чтобы входом IP^{-1} стадии было $R_0||L_0$, необходимо поменять местами левую и правую части. Но

$IP^{-1}(R_0||L_0) = IP^{-1}(IP$ (незашифрованный текст)) = незашифрованный текст

Т.е. получаем незашифрованный текст, что и демонстрирует возможность дешифрования *DES*.

Проблемы DES

Так как длина ключа равна 56 битам, существует 2^{56} возможных ключей. На сегодня такая длина ключа недостаточна, поскольку допускает успешное применение лобовых атак. Альтернативой *DES* можно считать *тройной DES*, IDEA, а также алгоритм Rijndael, принятый в качестве нового стандарта на алгоритмы симметричного шифрования.

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

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

$L_i = R_{i-1}$

$R_i = L_i \oplus f(R_{i-1}, K_i)$

Функция `F` проста. Сначала правая половина и i-ый *подключ* складываются по модулю 2^{32}. Затем результат разбивается на восемь 4-битовых значений, каждое из которых подается на вход *S-box*. *ГОСТ 28147* использует восемь различных *S-boxes*, каждый из которых имеет 4-битовый вход и 4-битовый выход. Выходы всех *S-boxes* объединяются в 32-битное слово, которое затем циклически сдвигается на 11 битов влево. Наконец, с помощью `XOR` результат объединяется с левой половиной, в результате чего получается новая правая половина.

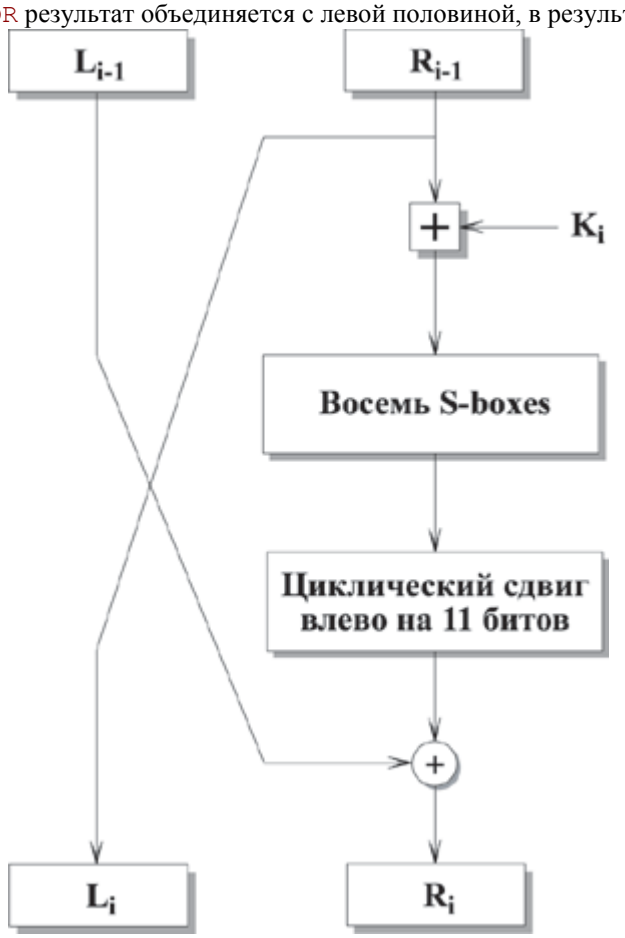


Рис. 4.9. I-ый раунд ГОСТ 28147

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

Считается, что стойкость *алгоритма ГОСТ 28147* во многом определяется структурой *S-boxes*. Долгое время структура *S-boxes* в открытой печати не публиковалась. В настоящее время известны *S-boxes*, которые используются в приложениях Центрального Банка РФ и считаются достаточно сильными. Напомню, что входом и выходом *S-box* являются 4-битные числа, поэтому каждый *S-box* может быть представлен в виде строки чисел от 0 до 15, расположенных в некотором порядке. Тогда порядковый номер числа будет являться входным значением *S-box*, а само число - выходным значением *S-box*.

1-ый S-box	0			3			4
	1		2		5		

2-ой S-box	4	1		2		3	5	0
3-ий S-box				3	0			
	4	5	2					1
4-ый S-box		3	0					5
	4			2	1			
5-ый S-box		2				5	3	
		0		4			1	
6-ой S-box		1	0					3
						2	5	4
7-ой S-box	3	1				5		
		0	4					2
8-ой S-box		5	3				0	
				4		1		2

Основные различия между DES и *ГОСТ 28147* следующие:

- DES использует гораздо более сложную процедуру создания *подключей*, чем *ГОСТ 28147*. В *ГОСТ* эта процедура очень проста.
- В DES применяется 56-битный ключ, а в *ГОСТ 28147* - 256-битный. При выборе сильных *S-boxes ГОСТ 28147* считается очень стойким.
- У *S-boxes* DES 6-битовые входы и 4-битовые выходы, а у *S-boxes ГОСТ 28147* 4-битовые входы и выходы. В обоих алгоритмах используется по восемь *S-boxes*, но размер *S-box ГОСТ 28147* существенно меньше размера *S-box* DES.
- В DES применяются нерегулярные перестановки P, в *ГОСТ 28147* используется 11-битный циклический сдвиг влево. Перестановка DES увеличивает лавинный эффект. В *ГОСТ 28147* изменение одного входного бита влияет на один *S-box* одного *раунда*, который затем влияет на два *S-boxes* следующего *раунда*, три *S-boxes* следующего и т.д. В *ГОСТ 28147* требуется 8 *раундов* прежде, чем изменение одного входного бита повлияет на каждый бит результата; DES для этого нужно только 5 *раундов*.
- В DES 16 *раундов*, в *ГОСТ 28147* - 32 *раунда*, что делает его более стойким к дифференциальному и линейному криптоанализу.

Алгоритм ГОСТ 28147-89 - Режим гаммирования

Зашифрование данных

Криптосхема, реализующая алгоритм зашифрования данных в режиме гаммирования показана на схеме.

Открытые данные, разбитые на **64**-разрядные блоки $\mathbf{To^{(1)}, To^{(2)},..., To^{(M-1)}, To^{(M)}}$, зашифровываются в режиме гаммирования путем поразрядного суммирования по модулю **2** в сумматоре $\mathbf{CM_5}$ с гаммой шифра **Гш**, которая вырабатывается блоками по **64** бита:

$\mathbf{Гш = (Гш^{(1)}, Гш^{(2)},...,Гш^{(M-1)}, Гш^{(M)})}$

где **M** - определяется объемом шифруемых данных.

В **КЗУ** вводятся **256** бит ключа. **В** накопителе $\mathbf{N_1}$, $\mathbf{N_2}$ вводится **64**-разрядная двоичная последовательность (синхропосылка) $\mathbf{S=(S_1, S_2,..., S_{64})}$, являющаяся исходным заполнением этих накопителей для последующей выработки **M** блоков гаммы шифра.

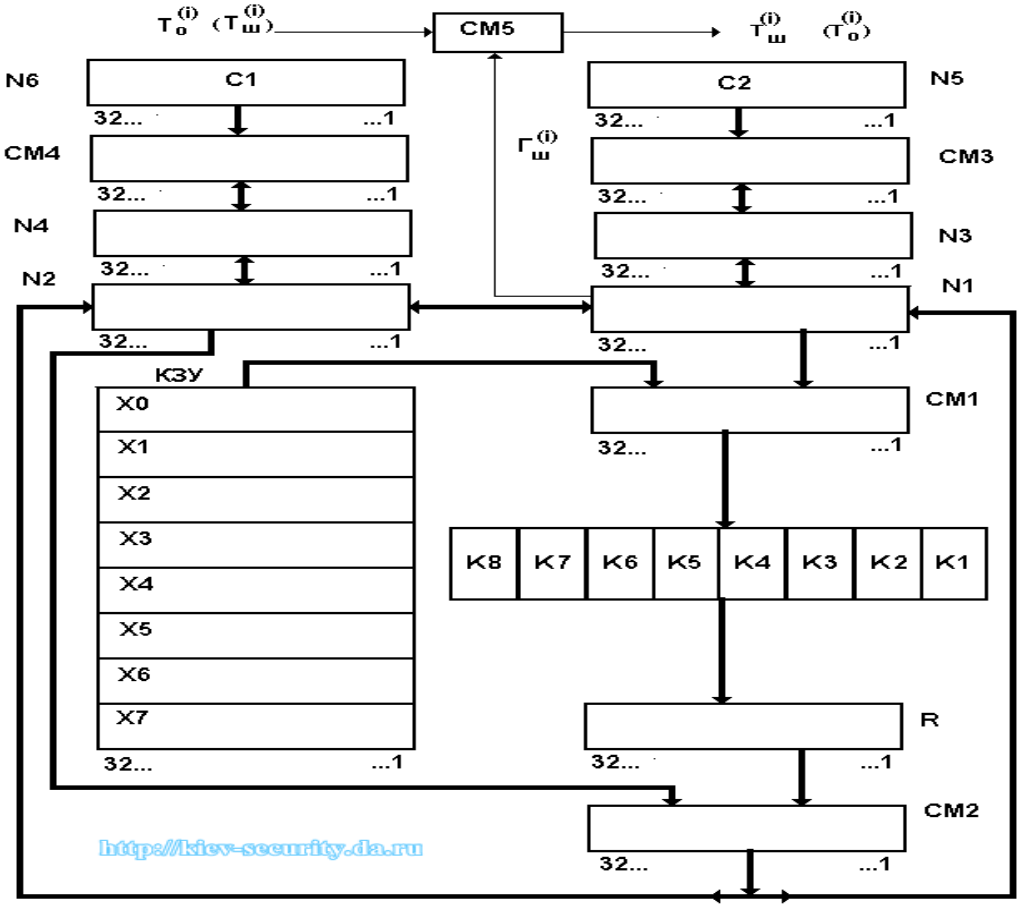


Рис.4.10 Структурная схема зашифрования в режиме гаммирования

Исходное заполнение накопителей N_1 и N_2 (синхропосылка S) зашифровывается в режиме простой замены. Результат зашифрования **A(S) = (Y_0, Z_0)** переписывается в **32**-разрядные накопители N_3 и N_4. Заполнение накопителя N_4 суммируется по модулю $(2^{32}-1)$ в сумматоре CM_4 с **32**-разрядной константой C_1 из накопителя N_6, результат записывается в N_4. Заполнение накопителя N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с **32**-разрядной константой C_2 из накопителя N_5, результат записывается в N_3. Заполнение N_3 переписывается в N_1, а заполнение N_4 переписывается в N_2, при этом заполнение N_3, N_4 сохраняется.

Заполнение N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1, N_2 образует первый **64**-разрядный блок гаммы шифра $Гш^{(1)}$, который суммируется поразрядно по модулю **2** в сумматоре CM_5 с первым **64**-разрядным блоком открытых данных. В результате суммирования получается **64**-разрядный блок зашифрованных данных. Аналогичным образом зашифровываются остальные блоки открытых данных.

В канал связи или память ЭВМ передаются синхропосылка **S** и блоки зашифрованных данных.

Расшифрование данных

При расшифровании криптосхема имеет тот же вид, что и при зашифровании открытых данных в режиме гаммирования. В **КЗУ** вводятся **256** бит ключа, с помощью которого осуществлялось зашифрование данных. В накопители N_1 и N_2 вводится синхропосылка S. Процесс выработки M блоков гаммы шифра осуществляется совершенно аналогично описанному выше. Блоки зашифрованных данных суммируются поразрядно по модулю **2** в сумматоре CM_5 с блоками гаммы шифра, в результате получаются блоки открытых данных.

5. СОВРЕМЕННЫЕ КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ

Предварительные математические понятия

Практически все операции *Rijndael* определяются на уровне байта. Байты можно рассматривать как элементы конечного поля GF (2^8). Некоторые операции определены в терминах четырехбайтных слов. Введем основные математические понятия, необходимые для обсуждения алгоритма.

Поле GF(2^8)

Элементы конечного поля могут быть представлены несколькими различными способами. Для любой степени простого числа существует единственное конечное поле, поэтому все представления GF (2^8) являются изоморфными. Несмотря на подобную эквивалентность, представление влияет на сложность реализации. Выберем классическое полиномиальное представление.

Байт b, состоящий из битов b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0, представляется в виде полинома с коэффициентами из {0, 1}:

$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$

Пример: байт с шестнадцатеричным значением '57' (двоичное 01010111) соответствует полиному

$x^6 + x^4 + x^2 + x + 1$

Сложение

В полиномиальном представлении сумма двух элементов является полиномом с коэффициентами, которые равны сумме по модулю 2 (т.е. 1 + 1 = 0) коэффициентов слагаемых.

Пример: '57' + '83' = 'DA' или в полиномиальной нотации:

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) =$

$x^7 + x^6 + x^4 + x^2$

В бинарной нотации мы имеем: 01010111 + 10000011 = 11011010. Очевидно, что сложение соответствует простому XOR (обозначается как $\oplus$) на уровне байта.

Выполнены все необходимые условия Абелевой группы: операция сложения (каждой паре элементов сопоставляется третий элемент группы, называемый их суммой), ассоциативность, нулевой элемент ('00'), обратный элемент (относительно операции сложения) и коммутативность.

Умножение

В полиномиальном представлении умножение в GF (2^8) соответствует умножению полиномов по модулю неприводимого двоичного полинома степени 8. Полином является неприводимым, если он не имеет делителей, кроме 1 и самого себя. Для *Rijndael* такой полином называется m(x) и определяется следующим образом:

$m(x) = x^8 + x^4 + x^3 + x + 1$

или '11B' в шестнадцатеричном представлении.

Пример: '57' • '83' = 'C1'

Или

$(x^6 + x^4 + x^2 + x + 1)\ (x^7 + x + 1) =$

$= x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 =$

$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

$(x^8 + x^4 + x^3 + x + 1)\ (x^5 + x^3) + x^7 + x^6 + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

Следовательно,

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^8 + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$

Ясно, что результат является двоичным полиномом не выше 8 степени. В отличие от сложения, простой операции умножения на уровне байтов не существует.

Умножение, определенное выше, является ассоциативным, и существует единичный элемент ('01'). Для любого двоичного полинома b(x) не выше 8-й степени можно использовать расширенный алгоритм Евклида для вычисления полиномов a(x) и c(x) таких, что

$b(x)\ a(x) + m(x)\ c(x) = 1$

Следовательно,

$a(x) \bullet b(x) \bmod m(x) = 1$

или

$b^{-1}(x) = a(x) \bmod m(x)$

Более того, можно показать, что

$a(x) \bullet (b(x) + c(x)) =$

$a(x) \bullet b(x) + a(x) \bullet c(x)$

Из всего этого следует, что множество из 256 возможных значений байта образует конечное поле GF (2^8) с XOR в качестве сложения и умножением, определенным выше.

Умножение на x

Если умножить b(x) на полином x, мы будем иметь:

$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$

$x \bullet b(x)$ получается понижением предыдущего результата по модулю m(x). Если $b_7 = 0$, то данное понижение является тождественной операцией. Если $b_7 = 1$, m(x) следует вычесть (т.е. XORed). Из этого

следует, что умножение на x может быть реализовано на уровне байта как левый сдвиг и последующий побитовый XOR с '1B'. Данная операция обозначается как b = xtime (a).

Полиномы с коэффициентами из GF

Полиномы могут быть определены с коэффициентами из GF(2^8). В этом случае четырехбайтный вектор соответствует полиному степени 4.

Полиномы могут быть сложены простым сложением соответствующих коэффициентов. Как сложение в GF(2^8) является побитовым XOR, так и сложение двух векторов является простым побитовым XOR.

Умножение представляет собой более сложное действие. Предположим, что мы имеем два полинома в GF(2^8).

$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$
$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$

c(x) = a(x) b(x) определяется следующим образом:

$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$

$c_0 = a_0 \bullet b_0$

$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$

$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$

$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$

$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$

$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$

$c_6 = a_3 \bullet b_3$

Ясно, что в таком виде c(x) не может быть представлен четырехбайтным вектором. Понижая c(x) по модулю полинома 4-й степени, результат может быть полиномом степени ниже 4. В *Rijndael* это сделано с помощью полинома

$M(x) = x^4 + 1$

так как

$x^j \bmod (x^4 + 1) = x^{j \bmod 4}$

Модуль, получаемый из a(x) и b(x), обозначаемый d(x) = a(x) $\otimes$ b(x), получается следующим образом:

$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$

$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$

$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$

$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$

Операция, состоящая из умножения фиксированного полинома a(x), может быть записана как умножение матрицы, где матрица является циклической. Мы имеем

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Замечание: $x^4 + 1$ не является несократимым полиномом в GF (2^8), следовательно, умножение на фиксированный полином необязательно обратимо. В алгоритме *Rijndael* выбран фиксированный полином, который имеет обратный.

Умножение на x

При умножении b(x) на полином x будем иметь:

$b_3x^4 + b_2x^3 + b_1x^2 + b_0x$

x $\otimes$ b(x) получается понижением предыдущего результата по модулю $1 + x^4$.

Это дает

$b_2x^3 + b_1x^2 + b_0x + b_3$

Умножение на x эквивалентно умножению на матрицу, как описано выше со всеми a_i = '00' за исключением a_1 = '01'. Имеем:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Следовательно, умножение на `x` соответствует циклическому сдвигу байтов внутри вектора.

Обоснование разработки

При разработке алгоритма учитывались следующие три критерия:

- противодействие всем известным атакам;
- скорость и компактность кода для широкого круга платформ;
- простота разработки.

В большинстве алгоритмов шифрования преобразование каждого раунда имеет структуру *сети Фейштеля* . В этом случае обычно часть битов в каждом промежуточном состоянии просто перемещается без изменения в другую половину. Преобразование *раунда* алгоритма *Rijndael* не имеет структуру *сети Фейштеля*. Вместо этого преобразование каждого *раунда* состоит из четырех различных преобразований, называемых слоями.

Каждый слой разрабатывался с учетом противодействия линейному и дифференциальному криптоанализу. В основу каждого слоя положена своя собственная функция:

1. Нелинейный слой состоит из параллельного применения *S-boxes* для оптимизации нелинейных свойств в наихудшем случае.
2. Слой линейного перемешивания строк гарантирует высокую степень диффузии для нескольких *раундов*.
3. Слой линейного перемешивания столбцов также гарантирует высокую степень диффузии для нескольких *раундов*.
4. Дополнительный слой ключа состоит из простого `XOR` промежуточного состояния с ключом *раунда*.

Перед первым *раундом* применяется дополнительное *забеливание* с использованием ключа. Причина этого состоит в следующем. Любой слой после последнего или до первого добавления ключа может быть просто снят без знания ключа и тем самым не добавляет безопасности в алгоритм (например, начальная и конечная перестановки в DES). Начальное или конечное добавление ключа применяется также в некоторых других алгоритмах, например IDEA, SAFER и Blowfish.

Для того чтобы сделать структуру алгоритма более простой, слой линейного перемешивания последнего *раунда* отличается от слоя перемешивания других *раундов*. Можно показать, что это в любом случае не повышает и не понижает безопасность. Это аналогично отсутствию операции swap в последнем *раунде* DES.

Спецификация алгоритма

Rijndael является блочным алгоритмом шифрования с переменной длиной блока и переменной длиной ключа. Длина блока и длина ключа могут быть независимо установлены в 128, 192 или 256 бит.

Состояние, ключ шифрования и число раундов

Различные преобразования выполняются над промежуточным результатом, называемым состоянием.

Состояние можно рассматривать как двумерный массив байтов. Этот массив имеет четыре строки и различное число столбцов, обозначаемое как `Nb`, равное длине блока, деленной на `32`. Ключ также можно рассматривать как двумерный массив с четырьмя строками. Число столбцов *ключа шифрования*, обозначаемое как `Nk`, равно длине ключа, деленной на `32`.

В некоторых случаях эти блоки также рассматриваются как одномерные массивы четырехбайтных векторов, где каждый вектор состоит из соответствующего столбца. Такие массивы имеют длину 4, 6 или 8 соответственно, и индексы в диапазонах 0 … 3, 0 … 5 или 0 … 7. Четырехбайтные вектора иногда мы будем называть словами.

Если необходимо указать четыре отдельных байта в четырехбайтном векторе, будет использоваться нотация `(a, b, c, d)`, где `a`, `b`, `c` и `d` являются байтами в позициях 0, 1, 2 и 3, соответственно, в рассматриваемом столбце, векторе или слове.

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$	$A_{0,4}$	$A_{0,5}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$	$A_{1,5}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$	$A_{2,5}$
$A_{3,0}$	$A_{3,1}$	$A_{3,1}$	$A_{3,3}$	$A_{3,4}$	$A_{3,5}$

$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$

Рис. 5.1. Пример состояния (с Nb = 6) и ключа шифрования (с Nk = 4)

Входы и выходы *Rijndael* считаются одномерными массивами из 8 байтов, пронумерованными от `0` до `4* Nb - 1`. Следовательно, эти блоки имеют длину 16, 24 или 32 байта, и массив индексируется в диапазонах `0 … 15`, `0 … 23` или `0 … 31`. Ключ считается одномерным массивом 8-битных байтов, пронумерованных от `0` до

4* Nk – 1. Следовательно, эти блоки имеют длину 16, 24 или 32 байта, и массив индексируется в диапазонах 0 … 15, 0 … 23 или 0 … 31.

Входные байты алгоритма отображаются в байты состояния в следующем порядке: $A_{0,0}$, $A_{1,0}$, $A_{2,0}$, $A_{3,0}$, $A_{0,1}$, $A_{1,1}$, $A_{2,1}$, $A_{3,1}$, … Байты *ключа шифрования* отображаются в массив в следующем порядке: $K_{0,0}$, $K_{1,0}$, $K_{2,0}$, $K_{3,0}$, $K_{0,1}$, $K_{1,1}$, $K_{2,1}$, $K_{3,1}$, … После выполнения операции шифрования выход алгоритма получается из байтов состояния аналогичным образом.

Следовательно, если одноразмерный индекс байта в блоке есть n, и двухмерный индекс есть (i,j), то мы имеем:

I = n mod 4

J = ⌊ n / 4 ⌋

N = i + 4*j

Более того, индекс i является также номером байта в четырехбайтном векторе или слове, j является индексом вектора или слова во вложенном блоке.

Число *раундов* обозначается Nr и зависит от значений Nb и Nk, что показано в следующей таблице.

Таблица 6.1 – Число раундов как функция от длины блока и длины ключа

Nr	**Nb = 4**	**Nb = 6**	**Nb = 8**
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Преобразование раунда

Преобразование *раунда* состоит из четырех различных преобразований. В нотации на псевдо С это можно записать следующим образом:

```
Round (State, RoundKey)
{
  ByteSub (State);
  ShiftRow (State);
  MixColumn (State);
  AddRoundKey (State, RoundKey);
}
```

Заключительный *раунд* алгоритма немного отличается и выглядит следующим образом:

```
FinalRound (State, RoundKey)
{
  ByteSub (State);
  ShiftRow (State);
  AddRoundKey (State, RoundKey);
}
```

Как мы видим, заключительный *раунд* эквивалентен остальным, за исключением того, что отсутствует слой MixColumn.

Преобразование ByteSub

Преобразование ByteSub является нелинейной байтовой подстановкой, выполняющейся для каждого байта состояния независимо. Таблица подстановки является обратимой и сконструирована в виде композиции двух преобразований:

1. Во-первых, берется мультипликативная инверсия в GF (2^8) с определенным выше представлением. '00' отображается сам в себя.
2. Затем применяется афинное (в GF (2)) преобразование, определяемое следующим образом:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Применение описанного *S-box* ко всем байтам состояния обозначается как ByteSub (State). На рисунке 6.2 показан результат применения преобразования `ByteSub` к `State`.

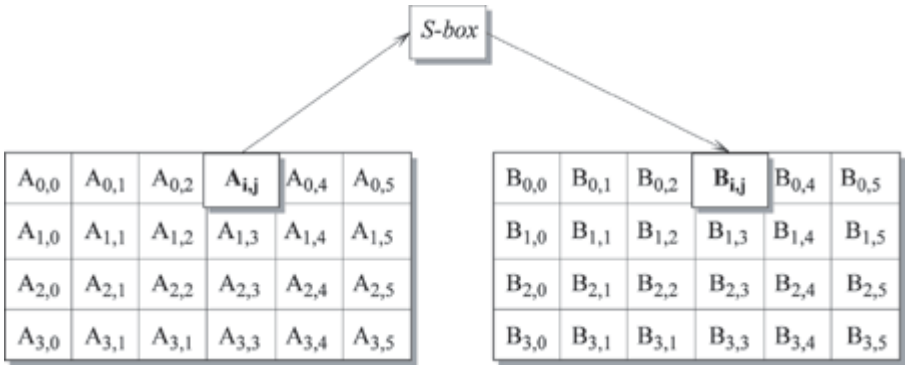


Рис. 5.2. Применение ByteSub для каждого байта в State

Инверсия `ByteSub` есть применение байтовой подстановки в соответствии с инверсной таблицей. Это получается инверсией афинного отображения и мультипликативной инверсией в GF (2^8).

Преобразование ShiftRow

В `ShiftRow` строки состояния циклически сдвигаются на различные значения. Нулевая строка не сдвигается. Строка 1 сдвигается на C_1 байтов, строка 2 на C_2 байтов, строка 3 на C_3 байтов. Величины C_1, C_2 и C_3 зависят от `Nb`. Значения приведены в следующей таблице.

Таблица 6.2 – Величина сдвига в зависимости от длины блока

b	1	2	3

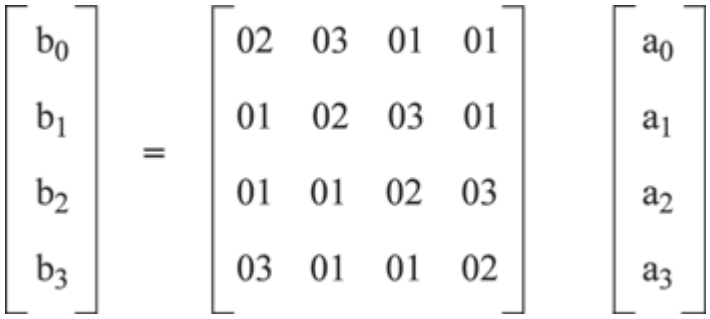
Операция сдвига строк на указанные значения обозначается как ShiftRow (State). Инверсией для `ShiftRow` является циклический сдвиг трех нижних строк соответственно на `Nb - C1`, `Nb - C2` и `Nb - C3` байт, чтобы байт в позиции `j` в строке `i` перемещался в позиции `(j + Nb - Ci) mod Nb`.

Преобразование MixColumn

В `MixColumn` столбцы состояния рассматриваются как полиномы в GF (2^8) и умножаются по модулю $x^4 + 1$ на фиксированный полином:

$c(x) = '03' x^3 + '01' x^2 + '01' x + '02'$

Данный полином является взаимнопростым с $x^4 + 1$ и, следовательно, инвертируем. Как было описано выше, это может быть записано в виде умножения матрицы. Пусть $b(x) = c(x) \otimes a(x)$



Применение данной операции ко всем столбцам состояния обозначается как

MixColumn (State)

Инверсия `MixColumn` является аналогичным `MixColumn`. Каждый столбец преобразуется умножением его на полином `d(x)`, определяемый следующим образом:

('03' x^3 + '01' x^2 + '01' x + '02') ⊗ d(x) = '01'

В результате получаем

d(x) = '0B' x^3 + '0D' x^2 + '09' x + '0E'

Сложение с ключом раунда

Выполняется операция побитового `XOR` ключа *раунда* с текущим состоянием. Длина ключа *раунда* равна длине блока `Nb`. Данное преобразование обозначается как

AddRoundKey (State, RoundKey)

`AddRoundKey` является инверсией самого себя.

Создание ключей раунда

Ключи *раунда* получаются из *ключа шифрования* с помощью преобразования, состоящего из двух компонентов: расширение ключа и выбор ключа *раунда*. Основной принцип состоит в следующем:

- Общее число битов ключа *раунда* равно длине блока, умноженной на количество *раундов* плюс 1. Например, для длины блока 128 бит и 10 *раундов* необходимо 1408 битов ключа *раунда*.
- *Ключ шифрования* расширяется в `ExpandedKey`.
- Ключи *раунда* получаются из этого `ExpandedKey` следующим способом: первый ключ *раунда* состоит из первых `Nb` слов, второй состоит из следующих `Nb` слов и т.д.

Расширение ключа

`Expanded Key` является линейным массивом четырехбайтных слов и обозначается как `W [Nb * (Nr + 1)]`. Первые `Nk` слов состоят из *ключа шифрования*. Остальные слова определяются рекурсивно. Функция расширения ключа зависит от значения `Nk`: существует версия функции для `Nk`, равным или меньшим 6, и версия для `Nk` больше 6.

Для `Nk` ≤6 мы имеем:

```
KeyExpansion (byte Key [4*Nk]
          word W[Nb * (Nr + 1)])
{
 for (i = 0; i < Nk; i++)
  W[i] =(Key [4*i], Key [4*i+1],
       Key [4*i+2], Key [4*i+3]);
 for (i = Nk; i < Nb * (Nr + 1); i++) {
  temp = W [i - 1];
  if (i % Nk == 0)
   temp = SubByte (RotByte (temp)) ^
        Rcon [i / Nk];
  W [i] = W [i- Nk] ^ temp;
 }
}
```

В данном случае `SubByte (W)` является функцией, которая возвращает четырехбайтное слово, в котором каждый байт является результатом применения *S-box Rijndael* к байту в соответствующей позиции во входном слове. Функция `RotByte (W)` возвращает слово, в котором байты циклически переставлены таким образом, что для входного слова `(a, b, c, d)` создается выходное слово `(b, c, d, a)`.

Можно заметить, что первые `Nk` слов заполняются *ключом шифрования*. Каждое следующее слово `W[i]` равно `XOR` предыдущего слова `W[i-1]` и позиций слова `Nk` до `W[i - Nk]`. Для слов в позициях, которые кратны `Nk`, сначала применяется преобразование `XOR` к `W[i-1]` и константой *раунда*. Данное преобразование состоит из циклического сдвига байтов в слове `RotByte`, за которым следует применение табличной подстановки для всех четырех байтов в слове `(SubByte)`.

Для `Nk > 6` мы имеем:

```
KeyExpansion (byte Key [4*Nk]
          word W [Nb* (Nr+1)])
{
 for (i=0; i < Nk; i++)
  W[i]= (key [4*i], key [4*i+1],
       key [4*i+2], key [4*i+3]);
 for (i = Nk; i < Nb * (Nr + 1); i++) {
  temp = W [i-1];
  if (i % Nk == 0)
```

```
    temp = SubByte (RotByte (temp)) ^
         Rcon [i / Nk];
  else if (i % Nk == 4)
    temp = SubByte (temp);
  W[i] = W[i - Nk] ^ temp;
 }
}
```

Отличие в схеме для `Nk ≤6` состоит в том, что для `i-4` кратных `Nk`, `SubByte` применяется для `W[i-1]` перед `XOR`.

Константы *раунда* не зависят от `Nk` и определяются следующим образом:

```
Rcon [i] = (RC [i], '00', '00', '00')
```

`RC [i]` являются элементами в GF (2^8) со значением $x^{(i-1)}$ таким, что:

```
RC [1] = 1 (т.е. '01')
RC [i] = x (т.е. '02') • (RC [i-1]) = x(i-1)
```

Выбор ключа раунда

Ключ *раунда* `i` получается из слов буфера ключа *раунда* `W [Nb * i] до W [Nb * (i+1)]`.

Алгоритм шифрования

Алгоритм шифрования `Rijndael` состоит из

- начального сложения с ключом;
- `Nr - 1` *раундов*;
- заключительного *раунда*.

В С-подобном представлении это выглядит так:

```
Rijndael (State, CipherKey)
{
  KeyExpansion (CipherKey, ExpandedKey);
  AddRoundKey (State, ExpandedKey);
  for (i=1; i < Nr; i++)
    Round (State, ExpandedKey + Nb*i);
  FinalRound (State, ExpandedKey + Nb*Nr)
}
```

Расширение ключа может быть выполнено заранее, и `Rijndael` может быть специфицирован в терминах расширенного ключа.

```
Rijndael (State, ExpandedKey)
{
  AddRoundKey (State, ExpandedKey);
  for (i=1; i < Nr; i++)
    Round (State, ExpandedKey + Nb*i);
  FinalRound (State, ExpandedKey + Nb*Nr)
}
```

Замечание: расширенный ключ всегда получается из *ключа шифрования* и никогда не специфицируется непосредственно. Тем не менее, на выбор самого *ключа шифрования* ограничений не существует.

Преимущества алгоритма

Преимущества, относящиеся к аспектам реализации:

- *Rijndael* может выполняться быстрее, чем обычный блочный алгоритм шифрования. Выполнена оптимизация между размером таблицы и скоростью выполнения.
- *Rijndael* можно реализовать в смарт-карте в виде кода, используя небольшой RAM и имея небольшое число циклов. Выполнена оптимизация размера ROM и скорости выполнения.
- Преобразование *раунда* допускает параллельное выполнение, что является важным преимуществом для будущих процессоров и специализированной аппаратуры.
- Алгоритм шифрования не использует арифметические операции, поэтому тип архитектуры процессора не имеет значения.

Простота разработки:

- Алгоритм шифрования полностью "самоподдерживаемый". Он не использует других криптографических компонентов, *S-box*'ов, взятых из хорошо известных алгоритмов, битов, полученных из специальных таблиц, чисел типа p и тому подобных уловок.
- Алгоритм не основывает свою безопасность или часть ее на неясностях или плохо понимаемых итерациях арифметических операций.
- Компактная разработка алгоритма не дает возможности спрятать люки.

Переменная длина блока:

- Длины блоков от 192 до 256 бит позволяют создавать хэш-функции без коллизий, использующие *Rijndael* в качестве функции сжатия. Длина блока 128 бит сегодня считается для этой цели недостаточной.

Расширения:

- Разработка позволяет специфицировать варианты длины блока и длины ключа в диапазоне от 128 до 256 бит с шагом в 32 бита.
- Хотя число *раундов Rijndael* зафиксировано в данной спецификации, в случае возникновения проблем с безопасностью он может модифицироваться и иметь число *раундов* в качестве параметра.

Расширения

Различная длина блока и ключа шифрования

Обработка ключа поддерживает длину ключа, которая была бы кратна 4 байтам. Единственным параметром, который необходим для определения другой длины ключа, отличной от 128, 192 или 256 бит, является число *раундов* алгоритма.

Структура алгоритма допускает произвольную длину блока, кратную 4 байтам, с минимумом в 16 байтов. Добавление ключа и `ByteSub` и `MixColumn` преобразования не зависят от длины блока. Единственным

преобразованием, которое зависит от длины блока, является ShiftRow. Для каждой длины блока должен быть определен специальный массив C_1, C_2, C_3.

Можно определить расширение *Rijndael*, которое также поддерживает длины блока и ключа между 128 и 256 битами с приращением в 32 бита. Число *раундов* определяется так:

Nr = max (Nk, Nb) + 6

Это расширяет правило для количества *раундов* для альтернативных длин блока и ключа.

Дополнительные значения C_1, C_2 и C_3 определены в следующей таблице.

Таблица 6.3 – Величина сдвига в зависимости от длины блока

Величина сдвига в зависимости от длины блока			
Nb	**C1**	**C2**	**C3**
5	1	2	3
7	1	2	4

6.ТЕОРИЯ СЛОЖНОСТИ ВЫЧИСЛЕНИЙ.

Сложность вычислений.

Сложность вычисления (т. е. работы алгоритма на данном входном слове) определяется *ресурсами*, требующимися для этого вычисления. Два важнейших ресурса - *время* и *память* (наибольший номер ячейки, над которым побывала головка Машина Тьюринга в процессе вычисления). Итак, ресурс, требуемый для конкретного вычисления, - это число.

Теория сложности сообщает время, за которое может быть взломан алгоритм.

Сложность алгоритмов.

При оценке сложности алгоритма учитывается работа алгоритма на всех выходах или крайняя оценка – "в наихудшем случае". Обычно вычислительная сложность характеризуется O ("O большое"), таким образом временная сложность не зависит от реализации.

Классификация алгоритмов, при ремере входных данных n :

- **Постоянные –** сложность не зависит от n : O(1);
- **Линейные –** сложность : O(n);
- **Полиномиальные –** сложность : $O(n^m)$, где *m* это константа;
- **Экспоненциальные –** сложность : $O(t^{f(n)})$, где *t* константа большая 1, а f(n) – полиномиальная функция;
- **Суперполиномиальные–** сложность : $O(c^{f(n)})$, где c константа, а f(n) – функция возрастающая быстрее постоянной, но медленнее линейной;

1. 3. Сложность задач.

Как отличить простые задачи от сложных?

Простые задачи (решаемые) – задачи, решаемые за полиномиальное время (например решение СЛУ в рациональных числах)

Сложные задачи (трудные, не решаемые) – задачи, которые не решаются за полиномиальное время, либо алгоритм решения за полиномиальное время не найден.

Помимо этого, существуют *принципиально неразрешимые* задачи, доказано А.Тьюрингом.

Сложность задач не определяется по сложности наилучшего алгоритма, ее решающего. Для оценки сложности вводится классификация по сложности функций, вычисление которых возможно при задаваемых ограничениях на потребляемые ресурсы:

a. **a.** Класс **P –** класс задач, которые можно решить за полиномиальное время.

b. **b.** Класс **NP –** класс задач, которые можно решить за полиномиальное время только на недетерминированной Машине Тьюринга: в отличии от обычной МТ может делать предположения. Это задачи у которых есть ответ, найти который трудно, но проверить можно за полиномиальное время.

Замечание: Ясно, что **P** входит **NP**, но вот проверить, что (**P <> NP)** или (**P = NP)** до сих пор не удалось. Это одна из важнейших проблем XXI века.

c. **c.** Класс **NP-полных** задач – класс задач, не менее сложных, чем любая **NP** задача. Т.е. решив такие задачи за полиномиальное время можно доказать **(P = NP).** К этому классу относится *задача выполнимости* - Задано логическое выражение. Существует ли набор правильных значений, на которых данное выражение будет истиной?(Пока не доказано.)

d. **d.** Класс **PSPACE –** класс задач, которые могут быть решены в полиномиальном пространстве, но не обязательно за полиномиальное время.

e. **e.** Класс **PSPACE-полных** задач **–** класс задач, которые обладают следующим свойством: если любая из них **NP**, то **PSPACE=NP**, и если любая из них является **P** проблемой, то **PSPACE=NP.**

f. **f.** Класс **EXPTIME –** класс задач, которые решаются за экспоненциальное время.

g. g. Класс **EXPTIME-полных** задач - класс задач, которые не решаются за детерминированное полиномиальное время. Известно, что **P<>EXPTIME.**

7.АЛГОРИТМ RSA. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ АЛГОРИТМА. СТОЙКОСТЬ АЛГОРИТМА.

Алгоритм RSA

Диффи и Хеллман определили новый подход к шифрованию, что вызвало к жизни разработку алгоритмов шифрования, удовлетворяющих требованиям систем с *открытым ключом*. Одним из первых результатов был алгоритм, разработанный в 1977 году Роном Ривестом, Ади Шамиром и Леном Адлеманом и опубликованный в 1978 году. С тех пор алгоритм Rivest-Shamir-Adleman (*RSA*) широко применяется практически во всех приложениях, использующих криптографию с *открытым ключом*.

Алгоритм основан на использовании того факта, что задача ***факторизации*** является трудной, т.е. легко перемножить два числа, в то время как не существует полиномиального алгоритма нахождения простых сомножителей большого числа.

Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные являются целыми между 0 и n −1 для некоторого n.

Описание алгоритма

Алгоритм, разработанный Ривестом, Шамиром и Адлеманом, использует выражения с экспонентами. Данные шифруются блоками, каждый блок рассматривается как число, меньшее некоторого числа n. Шифрование и дешифрование имеют следующий вид для некоторого незашифрованного блока M и зашифрованного блока C.

$C = M^e \pmod n$

$M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n$

Как отправитель, так и получатель должны знать значение n. Отправитель знает значение e, получатель знает значение d. Таким образом, *открытый ключ* есть KU = {e, n} и *закрытый ключ* есть KR = {d, n}. При этом должны выполняться следующие условия:

1. Возможность найти значения e, d и n такие, что $M^{ed} = M \bmod n$ для всех $M < n$.
2. Относительная легкость вычисления M^e и C^d для всех значений $M < n$.
3. Невозможность определить d, зная e и n.

Сначала рассмотрим первое условие. Нам необходимо выполнение равенства:

$M^{ed} = M \pmod n$

Рассмотрим некоторые математические понятия, свойства и теоремы, которые позволят нам определить e, d и n.

1. Если $(a \cdot b) \equiv (a \cdot c) \bmod n$, то $b \equiv c \bmod n$, если a и n взаимнопростые, т.е $\gcd(a, n) = 1$.
2. Обозначим Z_p - все числа, взаимнопростые с p и меньшие p. Если p - простое, то Z_p - это все остатки. Обозначим w^{-1} такое число, что $w \cdot w^{-1} \equiv 1 \bmod p$.

Тогда $\forall w \in Z_p \ \exists z: w \cdot z \equiv 1 \bmod p$

Доказательство этого следует из того, что т.к. w и p взаимнопростые, то при умножении всех элементов Z_p на w остатками будут все элементы Z_p, возможно, переставленные. Таким образом, хотя бы один остаток будет равен 1.

3. Определим функцию Эйлера следующим образом: $\Phi(n)$ - число положительных чисел, меньших n и взаимнопростых с n. Если p - простое, то $\Phi(p) = p-1$.

Если p и q - простые, то $\Phi(p \cdot q) = (p-1) \cdot (q-1)$.

В этом случае $Z_{p \cdot q} = \{0, 1, j, (p \cdot q - 1)\}$.

Перечислим остатки, которые не являются взаимнопростыми с $p \cdot q$:

$\{p, 2 \cdot p, j, (q-1) \cdot p\}$

$\{q, 2 \cdot q, j, (p-1) \cdot q\}$

0

Таким образом

$\Phi(p \cdot q) = p \cdot q - [(q-1) + (p-1) + 1] = p \cdot q - (p+q) + 1 = (p-1) \cdot (q-1)$.

4. ***Теорема Ферма.***

$a^{n-1} \equiv 1 \bmod n$, если n - простое.

Если все элементы Z_n умножить на a по модулю n, то в результате получим все элементы Z_n, быть может, в другом порядке. Рассмотрим следующие числа:

$\{a \bmod n, 2 \cdot a \bmod n, j, (n-1) \cdot a \bmod n\}$ являются числами $\{1, 2, j, (n-1)\}$, быть может, в некотором другом порядке. Теперь перемножим по модулю n числа из этих двух множеств.

$[(a \bmod n) \cdot (2a \bmod n) \cdot \ldots \cdot (n-1)a \bmod n]$

$\bmod n \equiv (n-1)! \bmod n \quad (n-1)! \, a^{n-1} \equiv (n-1)! \bmod n$

n и $(n-1)!$ являются взаимнопростыми, если n - простое, следовательно, $a^{n-1} \equiv 1 \bmod n$.

5. ***Теорема Эйлера.***

$a^{\Phi(n)} \equiv 1 \bmod n$ для всех взаимнопростых a и n.

Это верно, если n - простое, т.к. в этом случае $\Phi(n) = n-1$. Рассмотрим множество $R = \{x_1, x_2, j, x\Phi(n)\}$. Теперь умножим по модулю n каждый элемент этого множества на a. Получим множество $S = \{a \cdot x_1 \bmod n, a \cdot x_2 \bmod n, j, a \cdot x\Phi(n) \bmod n\}$. Это множество является перестановкой множества R по следующим причинам.

Так как a является взаимнопростым с n и x_i являются взаимнопростыми с n, то $a \cdot x_i$ также являются взаимнопростыми с n. Таким образом, S - это множество целых, меньших n и взаимнопростых с n. В S нет дублей, т.к. если $a \cdot x_i \bmod n = a \cdot x_j \bmod n \Rightarrow x_i = x_j$. Следовательно, перемножив элементы множеств S и R, получим:

$$\prod_{i=1}^{\Phi(n)} (a \cdot x_i \bmod n) \equiv \prod_{i=1}^{\Phi(n)} x_i \bmod n$$

$$\left(\prod_{i=1}^{\Phi(n)} a \cdot x_i \equiv \prod_{i=1}^{\Phi(n)} x_i\right) \bmod n$$

$$\left(a^{\Phi(n)} \cdot \prod_{i=1}^{\Phi(n)} x_i \equiv \prod_{i=1}^{\Phi(n)} x_i\right) \bmod n$$

$$\left(a^{\Phi(n)} \equiv 1\right) \bmod n$$

Теперь рассмотрим сам *алгоритм RSA*. Пусть p и q - простые.

$n = p \cdot q$.

Надо доказать, что $\forall M < n: M^{\Phi(n)} = M^{(p-1) \cdot (q-1)} \equiv 1 \bmod n$

Если $\gcd(M, n) = 1$, то соотношение выполняется. Теперь предположим, что $\gcd(M, n) \neq 1$, т.е. $\gcd(M, p \cdot q) \neq 1$. Пусть $\gcd(M, p) \neq 1$, т.е. $M = c \cdot p \Rightarrow \gcd(M, q) = 1$, так как в противном случае $M = c \cdot p$ и $M = l \cdot q$, но по условию $M < p \cdot q$.

Следовательно,

$$M^{\Phi(q)} \equiv 1 \bmod q$$

$$(M^{\Phi(q)})(p) \equiv 1 \bmod q$$

$$M^{\Phi(n)} \equiv 1 \bmod q$$

По определению модуля это означает, что $M^{\Phi(n)} = 1 + k \cdot q$. Умножим обе части равенства на $M = c \cdot p$. Получим

$M^{\Phi(n)+1} = c \cdot p + k \cdot q \cdot c \cdot p.$

$$M^{\Phi(n)} \equiv 1 \bmod n$$

Или

$$M^{\Phi(n)+1} \equiv M \bmod n$$

Таким образом, следует выбрать e и d такие, что $e \cdot d \equiv 1 \bmod (n)$

Или $e \equiv d^{-1} \bmod \Phi(n)$

e и d являются взаимнообратными по умножению по модулю $\Phi(n)$. Заметим, что в соответствии с правилами модульной арифметики, это верно только в том случае, если d (и следовательно, e) являются взаимнопростыми с $\Phi(n)$. Таким образом, $\gcd(\Phi(n), d) = 1$.

Теперь рассмотрим все элементы *алгоритма RSA*.

p, q - два простых целых числа	- открыто, вычисляемо.
$n = p \cdot q$	- закрыто, вычисляемо.
$d, \gcd(\Phi(n), d) = 1;$	- открыто, выбираемо.
$1 < d < \Phi(n)$	
$e \equiv d^{-1} \bmod \Phi(n)$	- закрыты, выбираемы.

Закрытый ключ состоит из $\{d, n\}$, *открытый ключ* состоит из $\{e, n\}$. Предположим, что пользователь A опубликовал свой *открытый ключ*, и что пользователь B хочет послать пользователю A сообщение M. Тогда B вычисляет $C = M^e \pmod n$ и передает C. При получении этого зашифрованного текста пользователь A дешифрует вычислением $M = C^d \pmod n$.

Суммируем *алгоритм RSA*:

Создание ключей

Выбрать простые p и q

Вычислить $n = p \cdot q$

Выбрать d $\gcd(\Phi(n), d) = 1; 1 < d < \Phi(n)$

Вычислить e $e = d^{-1} \bmod \Phi(n)$

Открытый ключ $KU = \{e, n\}$

Закрытый ключ $KR = \{d, n\}$

Шифрование

Незашифрованный текст: $M < n$

Зашифрованный текст: $C = M^e \pmod n$

Дешифрование

Зашифрованный текст: C

Незашифрованный текст: $M = C^d \pmod n$

Рассмотрим конкретный пример:

Выбрать два простых числа: $p = 7,\ q = 17$.

Вычислить $n = p \cdot q = 7 \cdot 17 = 119$.

Вычислить $\Phi(n) = (p - 1) \cdot (q - 1) = 96$.

Выбрать e так, чтобы e было взаимнопростым с $\Phi(n) = 96$ и меньше, чем $\Phi(n)$: $e = 5$.

Определить d так, чтобы $d \cdot e \equiv 1 \bmod 96$ и $d < 96$.

$d = 77$, так как $77 \cdot 5 = 385 = 4 \cdot 96 + 1$.

Результирующие ключи *открытый* $KU = \{5, 119\}$ и *закрытый* $KR = \{77, 119\}$.

Например, требуется зашифровать сообщение $M = 19$.

$19^5 = 66 \pmod{119}$; $C = 66$.

Для дешифрования вычисляется $66^{77} \pmod{119} = 19$.

Вычислительные аспекты

Рассмотрим сложность вычислений в *алгоритме RSA* при создании ключей и при шифровании/дешифровании.

Шифрование/дешифрование

Как шифрование, так и дешифрование включают возведение целого числа в целую степень по модулю n. При этом промежуточные значения будут громадными. Для того, чтобы частично этого избежать, используется следующее свойство модульной арифметики:

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

Другая оптимизация состоит в эффективном использовании показателя степени, так как в случае *RSA* показатели степени очень большие. Предположим, что необходимо вычислить x^{16}. Прямой подход требует 15 умножений. Однако можно добиться того же конечного результата с помощью только четырех умножений, если использовать квадрат каждого промежуточного результата: x^2, x^4, x^8, x^{16}.

Создание ключей

Создание ключей включает следующие задачи:

1. Определить два простых числа p и q.
2. Выбрать e и вычислить d.

Прежде всего, рассмотрим проблемы, связанные с выбором p и q. Так как значение $n = p \cdot q$ будет известно любому потенциальному противнику, для предотвращения раскрытия p и q эти простые числа должны быть выбраны из достаточно большого множества, т.е. p и q должны быть большими числами. С другой стороны, метод, используемый для поиска большого простого числа, должен быть достаточно эффективным.

В настоящее время неизвестны алгоритмы, которые создают произвольно большие простые числа. Процедура, которая используется для этого, выбирает случайное нечетное число из требуемого диапазона и проверяет, является ли оно простым. Если число не является простым, то опять выбирается случайное число до тех пор, пока не будет найдено простое.

Были разработаны различные тесты для определения того, является ли число простым. Это тесты вероятностные, то есть тест показывает, что данное число вероятно является простым. Несмотря на это они могут выполняться таким образом, что сделают вероятность близкой к 1. Если n "проваливает" тест, то оно не является простым. Если n "пропускает" тест, то n может как быть, так и не быть простым. Если n пропускает много таких тестов, то можно с высокой степенью достоверности сказать, что n является простым. Это достаточно долгая процедура, но она выполняется относительно редко: только при создании новой пары (KU, KR).

На сложность вычислений также влияет то, какое количество чисел будет отвергнуто перед тем, как будет найдено простое число. Результат из теории чисел, известный как теорема простого числа, говорит, что простых чисел, расположенных около n в среднем одно на каждые $\ln(n)$ чисел. Таким образом, в среднем требуется проверить последовательность из $\ln(n)$ целых, прежде чем будет найдено простое число. Так как все четные числа могут быть отвергнуты без проверки, то требуется выполнить приблизительно $\ln(n)/2$ проверок. Например, если простое число ищется в диапазоне величин 2^{200}, то необходимо выполнить около $\ln(2^{200}) / 2 = 70$ проверок.

Выбрав простые числа p и q, далее следует выбрать значение e так, чтобы $\gcd(\Phi(n), e) = 1$ и вычислить значение d, $d = e^{-1} \bmod \Phi(n)$. Существует единственный алгоритм, называемый расширенным алгоритмом Евклида, который за фиксированное время вычисляет наибольший общий делитель двух целых и если этот общий делитель равен единице, определяет инверсное значение одного по модулю другого. Таким образом, процедура состоит в генерации серии случайных чисел и проверке каждого относительно $\Phi(n)$ до тех пор, пока не будет найдено число, взаимнопростое с $\Phi(n)$. Возникает вопрос, как много случайных чисел

придется проверить до тех пор, пока не найдется нужное число, которое будет взаимнопростым с Φ(n). Результаты показывают, что вероятность того, что два случайных числа являются взаимнопростыми, равна 0.6.

Криптосистема Эль-Гамаля.

Данная система является альтернативой RSA и при равном значении ключа обеспечивает ту же криптостойкость [12].

В отличие от RSA метод Эль-Гамаля основан на проблеме дискретного логарифма. Этим он похож на *алгоритм Диффи-Хелмана*. Если возводить число в степень в конечном поле достаточно легко, то восстановить аргумент по значению (то есть найти логарифм) довольно трудно.

Основу системы составляют параметры p и g - числа, первое из которых - простое, а второе - целое.

Александр генерирует секретный ключ a и вычисляет открытый ключ $y = g^a \bmod p$. Если Борис хочет послать Александру сообщение m, то он выбирает случайное число k, меньшее p и вычисляет

$y_1 = g^k \bmod p$ и

$y_2 = m\, y^k$,

где означает побитовое сложение по модулю 2. Затем Борис посылает (y_1,y_2) Александру.

Александр, получив зашифрованное сообщение, восстанавливает его:

$m = (y_1^a \bmod p)\, y_2$.

Алгоритм цифровой подписи DSA, разработанный NIST (National Institute of Standard and Technology) и являющийся частью стандарта DSS частично опирается на рассмотренный метод.

8.ЭЛЕКТРОННАЯ ПОДПИСЬ.

Общая схема цифровой подписи

Прежде всего стоит сказать, что система цифровой подписи использует *криптоситемы*, которые базируются на концепции открытого ключа. О них уже рассказывалось на страницах МК в статьях Владимира (Людена) Ю. Некрасова ╚Чтоб никто не догадался╝ и моей собственной ╚Открытый ключ к закрытой информации╝, поэтому в дальнейшем будем считать, что криптоситемы, базирующиеся на концепции открытого ключа, нам уже известны, и углубляться в них мы в дальнейшем не будем. Также следует сказать, что основной задачей систем цифровой подписи является обеспечение достоверности и конфиденциальности соответствующего сообщения.

Из чего же состоит такая система?

Вероятностный алгоритм или система генерирования ключей. Каждый абонент А получает пару ключей (КА, КА1), где КА — открытый ключ, а КА1 — тайный.

Алгоритм подписывания SIGN, который, используя сообщение М и тайный ключ КА1, выдает некоторое слово S = SIGN(M, KA1). Слово S называется *подписью абонента* А на сообщении М. В случае, если абонент А хочет послать сообщение М с заверением того, что оно послано именно им, то он отсылает пару (M, S).

Алгоритм проверки подписи CHECK, которым может воспользоваться любой желающий проверить факт, что подпись S на сообщении М принадлежит именно абоненту А — владельцу открытого ключа КА. Если CHECK(M, S, KA) = 1, то проверка подписи считается успешной.

Следует отметить, что для алгоритмов SIGN и CHECK для любого сообщения М и пары ключей КА и КА1 должно выполнятся условие CHECK(KA, M, SIGN(KA1, M)) = 1. Это соотношение определяет корректность системы цифровой подписи.

Надежность системы цифровой подписи обеспечивается тем, что только законный владелец тайного ключа КА1 может для сообщения М сделать такую подпись S, которая прошла бы проверку.

Предложенная схема цифровой подписи была развита американскими математиками *Диффи и Гелманом*. При этом они утверждают, что любую криптосистему с открытым ключом можно превратить в систему цифровой подписи следующим образом. Пускай E и D — алгоритмы шифрования и дешифрования соответственно, К и К1 — открытый и тайный ключи. Тогда цифровая подпись ставится по такому правилу: SIGN(M, K1) = DK1(M), а проверку подписи нужно проводить следующим образом: если EK(S) = M, то CHECK(K, S) = 1, во всех других случаях CHECK(K, S) = 0.

Итак, рассмотрев общую схему построения цифровой подписи, приступим к описанию конкретных систем цифровой подписи.

Аналогично криптосистемам системы цифровой подписи для обеспечения конфиденциальности сообщений допускают *вероятностную модификацию*. В вероятностных системах цифровой подписи алгоритм SIGN будет не детерминированным, а вероятностным. Таким образом, подпись S будет случайной величиной.

Рассмотрим цифровую подпись в системе Эль Гамала, которая базируется на той же идее, что и соответствующая криптосистема.

Процесс генерирования ключей начинается с выбора большого простого числа p, а также числа g (1 < g < p – 1), причем желательно чтобы число g имело достаточно большой порядок. Числа p и g, не будучи тайными, пребывают во всеобщем использовании абонентов сети. Каждый абонент выбирает случайное число a (1 < a < p – 1) и вычисляет h = g^a mod p. Открытым ключом в данном случае будут числа p, g и h, а тайным — число a.

Для того чтобы поставить свою подпись S на открытом сообщении M, абонент A должен выполнить следующие шаги:

1. Выбирается случайное число r (0 < r < q – 1).
2. Вычисляется S1 = g^r mod p.
3. Вычисляется r1 = r^(-1) mod (p –1).
4. Вычисляется S2 = (M – a*S1)*r1 mod (p – 1).
5. В качестве подписи S ставится пара чисел: S = (S1, S2).

Чтобы проверить правильность подписи, абонент B должен проверить выполнение следующего соотношения: g^M = (h^S1)*(S1^S2) mod p.

Стандарт цифровой подписи DSS

Национальный институт стандартов и технологии США (NIST) разработал федеральный стандарт *цифровой подписи DSS*. Для создания *цифровой подписи* используется алгоритм *DSA* (Digital Signature Algorithm). В качестве хэш-алгоритма стандарт предусматривает использование алгоритма SHA-1 (Secure Hash Algorithm). *DSS* первоначально был предложен в 1991 году и пересмотрен в 1993 году в ответ на публикации, касающиеся безопасности его схемы.

Подход DSS

DSS использует алгоритм, который разрабатывался для использования только в качестве *цифровой подписи*. В отличие от RSA, его нельзя использовать для шифрования или обмена ключами. Тем не менее, это технология открытого ключа.

Рассмотрим отличия подхода, используемого в *DSS* для создания *цифровых подписей*, от применения таких алгоритмов как RSA.

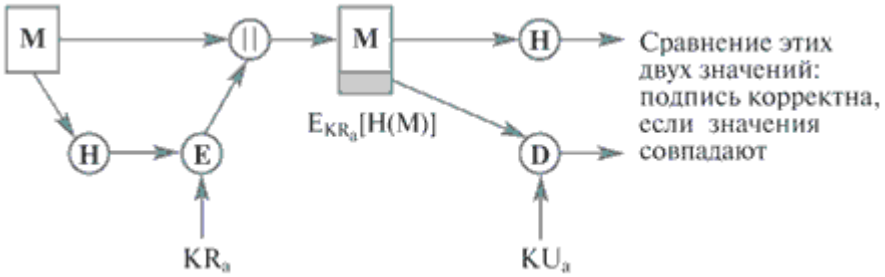


Рис. 8.1. Создание и проверка подписи с помощью алгоритма RSA

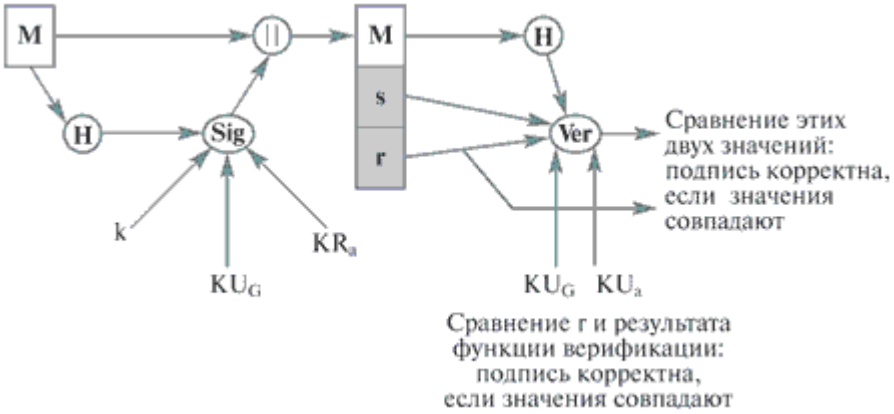


Рис. 8.2. Создание и проверка подписи с помощью стандарта DSS

В подходе RSA подписываемое сообщение подается на вход сильной хэш-функции, которая создает хэш-код фиксированной длины. Для создания подписи этот хэш-код шифруется с использованием закрытого ключа отправителя. Затем сообщение и подпись пересылаются получателю. Получатель вычисляет хэш-код сообщения и проверяет подпись, используя открытый ключ отправителя. Если вычисленный хэш-код равен дешифрованной подписи, то считается, что подпись корректна.

Подход *DSS* также использует сильную хэш-функцию. Хэш-код является входом функции подписи вместе со случайным числом k, созданным для этой конкретной подписи. Функция подписи также зависит от закрытого ключа отправителя KR_a и множества параметров, известных всем участникам. Можно считать, что это множество состоит из глобального открытого ключа KU_G. Результатом является подпись, состоящая из двух компонент, обозначенных как s и r.

Для проверки подписи получатель также создает хэш-код полученного сообщения. Этот хэш-код вместе с подписью является входом в функцию верификации. Функция верификации зависит от глобального открытого ключа KU_G и от открытого ключа отправителя KU_a. Выходом функции верификации является значение, которое должно равняться компоненте r подписи, если подпись корректна. Функция подписи такова, что только отправитель, знающий закрытый ключ, может создать корректную подпись.

Теперь рассмотрим детали алгоритма, используемого в *DSS*.

Алгоритм цифровой подписи

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

Общие компоненты группы пользователей

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

160-битное простое число q, т.е. $2^{159} < q < 2^{160}$.

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

$g = h^{(p-1)/q} \bmod p$, где h является целым между 1 и $(p-1)$ и g должно быть больше, чем $1,10$.

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

Закрытый ключ отправителя

Закрытый ключ x должен быть числом между 1 и $(q-1)$ и должен быть выбран случайно или псевдослучайно.

x - случайное или псевдослучайное целое,
$0 < x < q$,

Открытый ключ отправителя

Открытый ключ вычисляется из закрытого ключа как $y = g^x \bmod p$. Вычислить y по известному x довольно просто. Однако, имея открытый ключ y, вычислительно невозможно определить x, который является дискретным логарифмом y по основанию g.

$y = g^x \bmod p$

Случайное число, уникальное для каждой подписи.

k - случайное или псевдослучайное целое, $0 < k < q$, уникальное для каждого подписывания.

Подписывание

Для создания подписи отправитель вычисляет две величины, r и s, которые являются функцией от компонент открытого ключа (p, q, g), закрытого ключа пользователя (x), хэш-кода сообщения H (M) и целого k, которое должно быть создано случайно или псевдослучайно и должно быть уникальным при каждом подписывании.

$r = (g^k \bmod p) \bmod q$
$s = [\, k^{-1} (H(M) + xr) \,] \bmod q$
Подпись = (r, s)

Проверка подписи

Получатель выполняет проверку подписи с использованием следующих формул. Он создает величину v, которая является функцией от компонент общего открытого ключа, открытого ключа отправителя и хэш-кода полученного сообщения. Если эта величина равна компоненте r в подписи, то подпись считается действительной.

$w = s^{-1} \bmod q$
$u_1 = [\, H(M)\, w \,] \bmod q$
$u_2 = r\, w \bmod q$
$v = [\, (g^{u1}\, y^{u2}) \bmod p \,] \bmod q$
подпись корректна, если $v = r$

Докажем, что v = r в случае корректной подписи.

Лемма 1. Для любого целого t, если
$g = h^{(p-1)/q} \bmod p$
то $g^t \bmod p = g^{t \bmod q} \bmod p$

По теореме Ферма, так как h является взаимнопростым с p, то $h^{p-1} \bmod p = 1$. Следовательно, для любого неотрицательного целого n

$$g^{nq} \bmod p = (h^{(p-1)/q} \bmod p)^{nq} \bmod p$$
$$= h^{((p-1)/q)\, nq} \bmod p$$
$$= h^{(p-1)n} \bmod p$$
$$= ((h^{(p-1)} \bmod p)^n) \bmod p$$
$$= 1^n \bmod p = 1$$

Таким образом, для неотрицательных целых n и z мы имеем

$$g^{nq+z} \bmod p = (g^{nq}\, g^z) \bmod p$$
$$= ((g^{nq} \bmod p)\ (g^z \bmod p)) \bmod p$$
$$= g^z \bmod p$$

Любое неотрицательное целое t может быть представлено единственным образом как t = nq + z, где n и z являются неотрицательными целыми и 0 < z < q. Таким образом z = t mod q.

Лемма 2. Для неотрицательных чисел a и b: $g^{(a \bmod q + b \bmod q)} \bmod p = g^{(a+b) \bmod q} \bmod p$.

По лемме 1 мы имеем

$g^{(a \bmod q + b \bmod q) \bmod q} \bmod p$
$= g^{(a \bmod q + b \bmod q) \bmod q} \bmod p$
$= g^{(a + b) \bmod q} \bmod p$

Лемма 3. $y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p$

По определению $y = g^x \bmod p$. Тогда:

$y^{(rw) \bmod q} \bmod p$
$= (g^x \bmod p)^{(rw) \bmod q} \bmod p$ по правилам
$= g^{x\,((rw) \bmod q)} \bmod p$ модульной арифметики
$= g^{(x\,((rw \bmod q))) \bmod q} \bmod p$ по лемме 1
$= g^{(xrw) \bmod q} \bmod p$

Лемма 4. $((H(M) + xr)\ w) \bmod q = k$

По определению $s = (k^{-1}\ (H(M) + xr)) \bmod q$. Кроме того, так как q является простым, любое неотрицательное целое меньшее q имеет мультипликативную инверсию. Т.е. $(k\ k^{-1}) \bmod q = 1$. Тогда:

$(ks) \bmod q = (k((k^{-1}(H(M) + xr)) \bmod q)) \bmod q$
$= (k\ (k^{-1}(H(M) + xr))) \bmod q$
$= ((k^{k-1}) \bmod q)\ ((H(M) + xr) \bmod q) \bmod q$
$= (H(M) + xr) \bmod q$

По определению $w = s^{-1} \bmod q$, следовательно, $(ws) \bmod q = 1$. Следовательно:

$((H(M) + xr)\, w) \bmod q$
$= (((H(M) + xr) \bmod q)\ (w \bmod q)) \bmod q$
$= (((ks) \bmod q)\ (w \bmod q)) \bmod q$
$= (kws) \bmod q$
$= (k \bmod q)\ ((ws) \bmod q)) \bmod q$
$= k \bmod q$

Так как 0 < k < q, то k mod q = k.

Теорема. Используя определения для v и r, докажем, что v=r.

$v = ((g^{u1}\, y^{u2}) \bmod p) \bmod q$
$= ((g^{(H(M)\, w) \bmod q}\, y^{(rw) \bmod q}) \bmod p) \bmod q$
$= ((g^{(H(M)\, w) \bmod q}\, g^{(xrw) \bmod q}) \bmod p) \bmod q$
$= ((g^{(H(M)\, w) \bmod q + (xrw) \bmod q}) \bmod p) \bmod q$
$= ((g^{(H(M)\, w + xrw) \bmod q}) \bmod p) \bmod q$

$= ((g^{w\,(H(M)\,+\,xr)\ mod\ q)}\ mod\ p)\ mod\ q$
$= (g^{k\ mod\ p)}\ mod\ q$
= r

Электронная подпись на основе алгоритма RSA

Наиболее простым и распространенным инструментом электронной подписи является уже знакомый алгоритм RSA. Ниже оно будет рассмотрена в качестве примера. Кроме этого существуют еще десятки других схем цифровой подписи.

Предположим, что

d,*p*,*q* - секретные, а *e*, *n*=*pq* - открытые.

Замечания.

1. Разложение по *n* дает*: (n)=(p-1)(q-1); зная (n) и e, можно найти d.*
2. Из *e* и *d* можно найти кратность *(n); кратность (n) позволяет определить делители n.*

Пусть DATA - передаваемое Александром Борису сообщение.

Александр подписывает DATA для Бориса при передаче :

$Ee_B,n_B\{Ed_A,n_A\{DATA\}\}$.

При этом он использует:

* закрытый ключ Ed_A,n_A Александра,
* открытый ключ Ee_B,n_B Бориса.

Борис может читать это подписанное сообщение сначала при помощи закрытого ключа Ed_B,n_B Бориса с целью получения

$Ed_A,n_A\{DATA\} = Ed_B,n_B\{Ee_B,n_B\{Ed_A,n_A\{DATA\}\}\}$

и затем - открытого ключа Ee_A,n_A Александра для получения

$DATA = Ee_A,n_A\{Ed_A,n_A\{DATA\}\}$.

Таким образом, у Бориса появляется сообщение DATA, посланное ему Александром.

Очевидно, что данная схема позволяет защититься от нескольких видов нарушений.

Александр не может отказаться от своего сообщения, если он признает, что секретный ключ известен только ему.

Нарушитель без знания секретного ключа не может ни сформировать, ни сделать осмысленное изменение сообщения, передаваемого по линии связи.

Данная схема позволяет при решении многих конфликтных ситуаций обходиться без посредников.

Иногда нет необходимости зашифровывать передаваемое сообщение, но нужно его скрепить электронной подписью. В этом случае текст шифруется закрытым ключом отправителя и полученная цепочка символов прикрепляется к документу. Получатель с помощью открытого ключа отправителя расшифровывает подпись и сверяет ее с текстом.

В 1991 г. Национальный институт стандартов и технологии (NIST) предложил для появившегося тогда алгоритма цифровой подписи DSA (Digital Signature Algorithm) стандарт DSS (Digital Signature Standard), в основу которого положены алгоритмы Эль-Гамаля и RSA. [13]

9. ХЕШ-ФУНКЦИИ.

Требования к хэш-функциям

Хэш-функцией называется односторонняя функция, предназначенная для получения *дайджеста* или "отпечатков пальцев" файла, сообщения или некоторого блока данных.

Хэш-код создается функцией H:

h = H (M)

Где M является сообщением произвольной длины и h является *хэш-кодом* фиксированной длины.

Рассмотрим требования, которым должна соответствовать *хэш-функция* для того, чтобы она могла использоваться в качестве аутентификатора сообщения. Рассмотрим очень простой пример *хэш-функции*. Затем проанализируем несколько подходов к построению *хэш-функции*.

Хэш-функция H, которая используется для аутентификации сообщений, должна обладать следующими свойствами:

1. *Хэш-функция* H должна применяться к блоку данных любой длины.
2. *Хэш-функция* H создает выход фиксированной длины.
3. H (M) относительно легко (за полиномиальное время) вычисляется для любого значения M.
4. Для любого данного значения *хэш-кода* h вычислительно невозможно найти M такое, что H (M) = h.
5. Для любого данного x вычислительно невозможно найти y $\neq$x, что H (y) = H (x).
6. Вычислительно невозможно найти произвольную пару (x, y) такую, что H (y) = H (x).

Первые три свойства требуют, чтобы *хэш-функция* создавала *хэш-код* для любого сообщения.

Четвертое свойство определяет требование односторонности *хэш-функции*: легко создать *хэш-код* по данному сообщению, но невозможно восстановить сообщение по данному *хэш-коду*. Это свойство важно, если аутентификация с использованием *хэш-функции* включает секретное значение. Само секретное значение может не посылаться, тем не менее, если *хэш-функция* не является односторонней, противник может легко раскрыть секретное значение следующим образом. При перехвате передачи атакующий получает сообщение M и *хэш-код* C = H (S_{AB} || M). Если атакующий может инвертировать *хэш-функцию*, то, следовательно, он может получить S_{AB} || M = H^{-1} (C). Так как атакующий теперь знает и M и S_{AB} || M, получить S_{AB} совсем просто.

Пятое свойство гарантирует, что невозможно найти другое сообщение, чье значение *хэш-функции* совпадало бы со значением *хэш-функции* данного сообщения. Это предотвращает подделку аутентификатора при использовании зашифрованного *хэш-кода*. В данном случае противник может читать сообщение и, следовательно, создать его *хэш-код*. Но так как противник не владеет секретным ключом, он не имеет возможности изменить сообщение так, чтобы получатель этого не обнаружил . Если данное свойство не выполняется, атакующий имеет возможность выполнить следующую последовательность действий: перехватить сообщение и его зашифрованный *хэш-код*, вычислить *хэш-код* сообщения, создать альтернативное сообщение с тем же самым *хэш-кодом*, заменить исходное сообщение на поддельное. Поскольку *хэш-коды* этих сообщений совпадают, получатель не обнаружит подмены.

Хэш-функция, которая удовлетворяет первым пяти свойствам, называется ***простой или слабой хэш-функцией***. Если кроме того выполняется шестое свойство, то такая функция называется ***сильной хэш-функцией***. Шестое свойство защищает против класса атак, известных как атака "*день рождения*".

Простые хэш-функции

Все *хэш-функции* выполняются следующим образом. Входное значение (сообщение, файл и т.п.) рассматривается как последовательность n-битных блоков. Входное значение обрабатывается последовательно блок за блоком, и создается m-битное значение *хэш-кода*.

Одним из простейших примеров *хэш-функции* является побитный XOR каждого блока:

$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{ik}$

Где

C_i - i-ый бит *хэш-кода*, $1 \leq i \leq n$.

k - число n-битных блоков входа.

b_{ij} - i-ый бит в j-ом блоке.

$\oplus$- операция XOR.

В результате получается *хэш-код* длины n, известный как продольный избыточный контроль. Это эффективно при случайных сбоях для проверки целостности данных.

Часто при использовании подобного продольного избыточного контроля для каждого блока выполняется однобитный циклический сдвиг после вычисления *хэш-кода*. Это можно описать следующим образом.

- Установить n-битный *хэш-код* в ноль.
- Для каждого n-битного блока данных выполнить следующие операции:
  - сдвинуть циклически текущий *хэш-код* влево на один бит;

- выполнить операцию XOR для очередного блока и *хэш-кода*.

Это даст эффект "случайности" входа и уничтожит любую регулярность, которая присутствует во входных значениях.

Хотя второй вариант считается более предпочтительным для обеспечения целостности данных и предохранения от случайных сбоев, он не может использоваться для обнаружения преднамеренных модификаций передаваемых сообщений. Зная сообщение, атакующий легко может создать новое сообщение, которое имеет тот же самый *хэш-код*. Для этого следует подготовить альтернативное сообщение и затем присоединить n-битный блок, который является *хэш-кодом* нового сообщения, и блок, который является *хэш-кодом* старого сообщения.

Хотя простого XOR или ротационного XOR (RXOR) недостаточно, если целостность обеспечивается только зашифрованным *хэш-кодом*, а само сообщение не шифруется, подобная простая функция может использоваться, когда все сообщение и присоединенный к нему *хэш-код* шифруются. Но и в этом случае следует помнить о том, что подобная *хэш-функция* не может проследить за тем, чтобы при передаче последовательность блоков не изменилась. Это происходит в силу того, что данная *хэш-функция* определяется следующим образом: для сообщения, состоящего из последовательности 64-битных блоков X_1, X_2,..., X_N, определяется *хэш-код* C как поблочный XOR всех блоков, который присоединяется в качестве последнего блока:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Затем все сообщение шифруется, включая *хэш-код*, в режиме CBC для создания зашифрованных блоков Y_1, Y_2, ..., Y_{N+1}. По определению CBC имеем:

$$X_1 = IV \oplus D_K [Y_1]$$

$$X_i = Y_{i-1} \oplus D_K [Y_i]$$

$$X_{N+1} = Y_N \oplus D_K [Y_{N+1}]$$

Но X_{N+1} является *хэш-кодом*:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N =$$

$$(IV \oplus D_K [Y_1]) \oplus (Y_1 \oplus D_K [Y_2]) \oplus \ldots \oplus$$

$$(Y_{N-1} \oplus D_K [Y_N])$$

Так как сомножители в предыдущем равенстве могут вычисляться в любом порядке, следовательно, *хэш-код* не будет изменен, если зашифрованные блоки будут переставлены.

Первоначальный стандарт, предложенный NIST, использовал простой XOR, который применялся к 64-битным блокам сообщения, затем все сообщение шифровалось, используя режим CBC.

"Парадокс дня рождения"

Прежде чем рассматривать более сложные *хэш-функции*, необходимо проанализировать одну конкретную атаку на простые *хэш-функции*.

Так называемый "***парадокс дня рождения***" состоит в следующем. Предположим, количество выходных значений *хэш-функции* H равно n. Каким должно быть число k, чтобы для конкретного значения X и значений Y_1, …, Y_k вероятность того, что хотя бы для одного Y_i выполнялось равенство

$$H(X) = H(Y)$$

была бы больше 0,5.

Для одного Y вероятность того, что $H(X) = H(Y)$, равна $1/n$.

Соответственно, вероятность того, что $H(X) \neq H(Y)$, равна $1 - 1/n$.

Если создать k значений, то вероятность того, что ни для одного из них не будет совпадений, равна произведению вероятностей, соответствующих одному значению, т.е. $(1 - 1/n)^k$.

Следовательно, вероятность, по крайней мере, одного совпадения равна

$$1 - (1 - 1/n)^k$$

По формуле бинома Ньютона

$$(1 - a)^k =$$

$$1 - ka + (k(k-1)/2!)a^2 - \ldots \approx 1 - ka$$

$$1 - (1 - k/n) = k/n = 0,5$$

$$k = n/2$$

Таким образом, мы выяснили, что для m-битового *хэш-кода* достаточно выбрать 2^{m-1} сообщений, чтобы вероятность совпадения *хэш-кодов* была больше 0,5.

Теперь рассмотрим следующую задачу: обозначим $P(n, k)$ вероятность того, что в множестве из k элементов, каждый из которых может принимать n значений, есть хотя бы два с одинаковыми значениями. Чему должно быть равно k, чтобы $P(n, k)$ была бы больше 0,5?

Число различных способов выбора элементов таким образом, чтобы при этом не было дублей, равно

$$n(n-1) \ldots (n-k+1)n!/(n-k)!$$

Всего возможных способов выбора элементов равно

$$n^k$$

Вероятность того, что дублей нет, равна

$$n!/(n-k)!n^k$$

Вероятность того, что есть дубли, соответственно равна

$$1 - n!/(n-k)!n^k$$

$$P(n, k) = 1 - n! / ((n-k)! \times n^k) =$$

$$1 - (n \times (n-1) \times \ldots \times (n-k-1)) / n^k =$$

$$1 - [(n-1)/n \times (n-2)/n \times \ldots \times (n-k+1)/n] =$$

$1 - [(1- 1/n) \times (1 - 2/n) \times \ldots \times (1 - (k-1)/n)]$

Известно, что

$1 - x \leq e^{-x}$

$P(n, k) > 1 - [e^{-1/n} \times e^{-2}/n \times \ldots \times e^{-k}/n]$

$P(n, k) > 1 - e^{-k(k-1)/n}$

$1/2 = 1 - e^{-k(k-1)/n}$

$2 = e^{k(k-1)/n}$

$\ln 2 = k(k-1) / 2n$

$k(k-1) \approx k^2$

$k = (2n \times \ln 2)^{1/2} = 1{,}17\, n^{1/2} \approx n^{1/2}$

Если *хэш-код* имеет длину m бит, т.е. принимает 2^m значений, то

$k = \sqrt{2m} = 2^{m/2}$

Подобный результат называется "парадоксом дня рождения", потому что в соответствии с приведенными выше рассуждениями для того, чтобы вероятность совпадения дней рождения у двух человек была больше $0,5$, в группе должно быть всего 23 человека. Этот результат кажется удивительным, возможно, потому, что для каждого отдельного человека в группе вероятность того, что с его днем рождения совпадет день рождения кого-то другого в группе, достаточно мала.

Вернемся к рассмотрению свойств *хэш-функций*. Предположим, что используется 64-битный *хэш-код*. Можно считать, что это вполне достаточная и, следовательно, безопасная длина для *хэш-кода*. Например, если зашифрованный *хэш-код* C передается с соответствующим незашифрованным сообщением M, то противнику необходимо будет найти M' такое, что

$H(M') = H(M)$

для того, чтобы подменить сообщение и обмануть получателя. В среднем противник должен перебрать 2^{63} сообщений для того, чтобы найти такое, у которого *хэш-код* равен перехваченному сообщению.

Тем не менее, возможны различного рода атаки, основанные на "парадоксе дня рождения". Возможна следующая стратегия:

1. Противник создает $2^{m/2}$ вариантов сообщения, каждое из которых имеет некоторый определенный смысл. Противник подготавливает такое же количество сообщений, каждое из которых является поддельным и предназначено для замены настоящего сообщения.
2. Два набора сообщений сравниваются в поисках пары сообщений, имеющих одинаковый *хэш-код*. Вероятность успеха в соответствии с "парадоксом дня рождения" больше, чем $0,5$. Если соответствующая пара не найдена, то создаются дополнительные исходные и поддельные сообщения до тех пор, пока не будет найдена пара.
3. Атакующий предлагает отправителю исходный вариант сообщения для подписи. Эта подпись может быть затем присоединена к поддельному варианту для передачи получателю. Так как оба варианта имеют один и тот же *хэш-код*, будет создана одинаковая подпись. Противник будет уверен в успехе, даже не зная ключа шифрования.

Таким образом, если используется 64-битный *хэш-код*, то необходимая сложность вычислений составляет порядка 2^{32}.

В заключение отметим, что длина *хэш-кода* должна быть достаточно большой. Длина, равная 64 битам, в настоящее время не считается безопасной. Предпочтительнее, чтобы длина составляла порядка 100 битов.

Использование цепочки зашифрованных блоков

Существуют различные *хэш-функции*, основанные на создании цепочки зашифрованных блоков, но без использования секретного ключа. Одна из таких *хэш-функций* была предложена Рабином. Сообщение M разбивается на блоки фиксированной длины $M_1, M_2, \ldots, M_N$ и используется алгоритм симметричного шифрования, например DES, для вычисления *хэш-кода* G следующим образом:

H_0 = начальное значение

$H_i = E_{Mi}[H_{i-1}]$

$G = H_N$

Это аналогично использованию шифрования в режиме CBC, но в данном случае секретного ключа нет. Как и в случае любой *простой хэш-функции*, этот алгоритм подвержен "атаке дня рождения", и если шифрующим алгоритмом является DES и создается только 64-битный *хэш-код*, то система считается достаточно уязвимой.

Могут осуществляться другие атаки типа "дня рождения", которые возможны даже в том случае, если противник имеет доступ только к одному сообщению и соответствующему ему зашифрованному *хэш-коду* и не может получить несколько пар сообщений и зашифрованных *хэш-кодов*. Возможен следующий сценарий: предположим, что противник перехватил сообщение с аутентификатором в виде зашифрованного *хэш-кода*, и известно, что незашифрованный *хэш-код* имеет длину m битов. Далее противник должен выполнить следующие действия:

- Используя описанный выше алгоритм, вычислить незашифрованный *хэш-код* G.
- Создать поддельное сообщение в виде $Q_1, Q_2, \ldots, Q_{N-2}$.
- Вычислить $H_i = E_{Qi}[H_{i-1}]$ для $1 \leq i \leq N-2$.
- Создать $2^{m/2}$ случайных блока X и для каждого такого блока X вычислить $E_X[H_{N-2}]$. Создать дополнительно $2^{m/2}$ случайных блока Y и для каждого блока Y вычислить $D_Y[G]$, где D - дешифрующая функция, соответствующая E. Основываясь на "парадоксе дня рождения" можно сказать, что с высокой степенью вероятности эта последовательность будет содержать блоки X и Y такие, что $E_X[H_{N-2}] = D_Y[Y]$.
- Создать сообщение $Q_1, Q_2, \ldots, Q_{N-2}, X, Y$. Это сообщение имеет *хэш-код* G и, следовательно, может быть использовано вместе с зашифрованным аутентификатором.

Эта форма атаки известна как атака "встреча посередине". В различных исследованиях предлагаются более тонкие методы для усиления подхода, основанного на цепочке блоков. Например, Девис и Прайс описали следующий вариант:

$H_i = E_{Mi}[H_{i-1}] \oplus H_{i-1}$

Возможен другой вариант:

$H_i = E_{Hi-1}[M_i] \oplus M_i$

Однако обе эти схемы также имеют уязвимости при различных атаках. В более общем случае, можно показать, что некоторая форма "атаки дня рождения" имеет успех при любом хэш-алгоритме, включающем использование цепочки шифрованных блоков без применения секретного ключа.

Дальнейшие исследования были направлены на поиск других подходов к созданию функций хэширования.

Односторонняя (однонаправленная) функция (one way function) - это функция f осуществляющая отображение **X**->**Y**, где **X** и **Y** - произвольные множества, и удовлетворяющая следующим условиям:

1. $x \in$ **X** (области опеределения) легко вычислить $y=f(x)$, $y \in$ **Y.**
2. Почти для любого $y \in$ **Y** (области значения) найти $f^{-1}(y)$, т.е. x, для которого $y=f(x)$, вычислительно невозможно.

Почему в определнии стоит "почти для любого"? Потому, что если взять некоторый x и вычислить для него $y=f(x)$, то мы уже будем знать, что полученному y соответствует взятый нами x. Сохраним эти 2 значения и если когда-нибудь мы столкнемся с таким y, то мы спокойно найдем x.

Примером односторонней функции может служить вычисление $a^x \bmod n$, где a и n - некоторые числа. Такая задача называется задачей дискретного логарифмирования. В настоящее время нет эффективных алгоритмов, решающих эту задачу для больших чисел за приемлимое время. Вообще, приведенный пример можно назвать односторонней функцией с некоторой натяжкой, поскольку если появится такой алгоритм или вдруг несказано увеличатся вычислительные мощности, то такая задача становится решаемой !!! Поэтому поиск действительно односторонних функций или даже доказательство их существования является одной из важных задач криптографии.

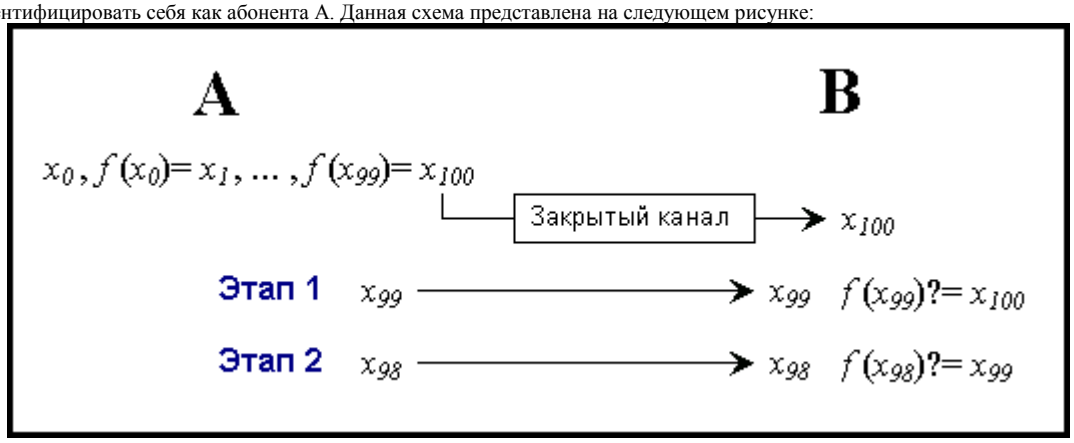
Примером применения односторонней функции может служить следующая схема идентифкации.

Абонент А вырабатывает следующую последовательность: x_0, $f(x_0)=x_1$, ..., $f(x_{99})=x_{100}$.

Затем x_{100} передается по секретному каналу (или при встрече) абоненту В.

Когда А необходимо идентифицировать себя, он передает по открытому каналуВ x_{99} . В проверяет, $f(x_{99})=?x_{100}$.

В следующий раз А передаст x_{99} и В проверит $f(x_{98})=?x_{99}$ и т.д. Перехват сообщений на i-ом этапе в открытом канале ничего не даст злоумышленнику, т.к. он не сможет получить соответсвующее значение x_{i-1}(из-за односторонней функции), чтобы в следующий раз идентифицировать себя как абонента А. Данная схема представлена на следующем рисунке:



Такие схемы применяются для идентифкации "свой/чужой".

Односторонней функцией с секретом (trapdoor one way function) называют функцию f_k осуществляющая отображение **X**->**Y**, где **X** и **Y** - произвольные множества, и удовлетворяющая следующим условиям:

1. $x \in$ **X** (области опеределения) легко вычислить $y=f_k(x)$, $y \in$ **Y.**
2. При известном k $y \in$ **Y** легко вычислить $x=f_k^{-1}(y)$,$x \in$ **X.**
3. Почти для всех k и почти для всех y нахождение $x=f_k^{-1}(y)$ вычислительно невозможно без знания параметра k.

На основе односторонних функций с секретом и строятся асимметричные криптосистемы. Так, алгоритм зашифрования с открытым ключом можно рассматривать как одностороннюю функцию с секретом, а секретом для этой функции является секретный ключ, используя который можно расшифровать сообщение. В качестве примера такой функции можно привести используемую в криптосистеме RSA модульную экспоненту (см. криптосистему RSA).

10. АУТЕНТИФИКАЦИЯ И ОБМЕН КЛЮЧАМИ.

Алгоритмы распределения ключей с использованием третьей доверенной стороны Понятие мастер-ключа

При симметричном шифровании два участника, которые хотят обмениваться конфиденциальной информацией, должны иметь один и тот же ключ. Частота изменения ключа должна быть достаточно большой, чтобы у противника не хватило времени для полного перебора ключа. Следовательно, сила любой криптосистемы во многом зависит от технологии распределения ключа. Этот термин означает передачу ключа двум участникам, которые хотят обмениваться данными, таким способом, чтобы никто другой не мог ни подсмотреть, ни изменить этот ключ. Для двух участников A и B распределение ключа может быть выполнено одним из следующих способов.

1. Ключ может быть создан A и физически передан B.
2. Третья сторона может создать ключ и физически передать его A и B.
3. A и B имеют предварительно созданный и недолго используемый ключ, один участник может передать новый ключ другому, применив для шифрования старый ключ.
4. Если A и B каждый имеют безопасное соединение с третьим участником C, C может передать ключ по этому безопасному каналу A и B.

Первый и второй способы называются ручным распределением ключа. Это самые надежные способы распределения ключа, однако во многих случаях пользоваться ими неудобно и даже невозможно. В распределенной системе любой хост или сервер должен иметь возможность обмениваться конфиденциальной информацией со многими аутентифицированными хостами и серверами. Таким образом, каждый хост должен иметь набор ключей, поддерживаемый динамически. Проблема особенно актуальна в больших распределенных системах.

Количество требуемых ключей зависит от числа участников, которые должны взаимодействовать. Если выполняется шифрование на сетевом или IP-уровне, то ключ необходим для каждой пары хостов в сети. Таким образом, если есть N хостов, то необходимое число ключей [N (N - 1)]/2. Если шифрование выполняется на прикладном уровне, то ключ нужен для каждой пары прикладных процессов, которых гораздо больше, чем хостов.

Третий способ распределения ключей может применяться на любом уровне стека протоколов, но если атакующий получает возможность доступа к одному ключу, то вся последовательность ключей будет раскрыта. Более того, все равно должно быть проведено первоначальное распространение большого количества ключей.

Поэтому в больших автоматизированных системах широко применяются различные варианты четвертого способа. В этой схеме предполагается существование так называемого центра распределения ключей (Key Destribution Center - KDC), который отвечает за распределение ключей для хостов, процессов и приложений. Каждый участник должен разделять уникальный ключ с KDC.

Использование центра распределения ключей основано на использовании иерархии ключей. Как минимум используется два типа ключей: мастер-ключи и ключи сессии.

Для обеспечения конфиденциальной связи между конечными системами используется временный ключ, называемый ключом сессии. Обычно ключ сессии используется для шифрования транспортного соединения и затем уничтожается. Каждый ключ сессии должен быть получен по сети из центра распределения ключей. Ключи сессии передаются в зашифрованном виде, используя мастер-ключ, который разделяется между центром распределения ключей и конечной системой.

Эти мастер-ключи также должны распределяться некоторым безопасным способом. Однако при этом существенно уменьшается количество ключей, требующих ручного распределения. Если существует N участников, которые хотят устанавливать соединения, то в каждый момент времени необходимо [N (N - 1)]/2 ключей сессии. Но требуется только N мастер-ключей, по одному для каждого участника.

Время жизни ключа сессии как правило равно времени жизни самой сессии.

Чем чаще меняются ключи сессии, тем более безопасными они являются, так как противник имеет меньше времени для взламывания данного ключа сессии. С другой стороны, распределение ключей сессии задерживает начало любого обмена и загружает сеть. Политика безопасности должна сбалансировать эти условия для определения оптимального времени жизни конкретного ключа сессии.

Если соединение имеет долгое время жизни, то должна существовать возможность периодически менять ключ сессии.

Для протоколов, не поддерживающих соединение, таких как протокол, ориентированный на транзакции, нет явной инициализации или прерывания соединения. Следовательно, неясно, как часто надо менять ключ сессии. Большинство подходов основывается на использовании нового ключа сессии для каждого нового обмена. Наиболее часто применяется стратегия использования ключа сессии только для фиксированного периода времени или только для определенного количества транзакций.

Протоколы аутентификации

Рассмотрим основные протоколы, обеспечивающие как взаимную аутентификацию участников, так и аутентификацию только одного из участников.

Взаимная аутентификация

Данные протоколы применяются для взаимной аутентификации участников и для обмена ключом сессии.

Основной задачей таких протоколов является обеспечение конфиденциального распределения ключа сессии и гарантирование его своевременности, то есть протокол не должен допускать повторного использования

старого ключа сессии. Для обеспечения конфиденциальности ключи сессии должны передаваться в зашифрованном виде. Вторая задача, обеспечение своевременности, важна, потому что существует угроза перехвата передаваемого сообщения и повторной его пересылки. Такие повторения в худшем случае могут позволять взломщику использовать скомпрометированный ключ сессии, при этом успешно подделываясь под другого участника. Успешное повторение может, как минимум, разорвать операцию аутентификации участников.

Такие повторы называются replay-атаками. Рассмотрим возможные примеры подобных replay-атак:

1. Простое повторение: противник просто копирует сообщение и повторяет его позднее.
2. Повторение, которое не может быть определено: противник уничтожает исходное сообщение и посылает скопированное ранее сообщение.

Один из возможных подходов для предотвращения replay-атак мог бы состоять в присоединении последовательного номера (sequence number) к каждому сообщению, используемому в аутентификационном обмене. Новое сообщение принимается только тогда, когда его последовательный номер правильный. Трудность данного подхода состоит в том, что каждому участнику требуется поддерживать значения sequence number для каждого участника, с которым он взаимодействует в данный момент. Поэтому обычно sequence number не используются для аутентификации и обмена ключами. Вместо этого применяется один из следующих способов:

1. Отметки времени: участник А принимает сообщение как не устаревшее только в том случае, если оно содержит отметку времени, которая, по мнению А, соответствует текущему времени. Этот подход требует, чтобы часы всех участников были синхронизированы.
2. Запрос/ответ: участник А посылает в запросе к В случайное число (nonce - number only once) и проверяет, чтобы ответ от В содержал корректное значение этого nonce.

Считается, что подход с отметкой времени не следует использовать в приложениях, ориентированных на соединение, потому что это технически трудно, так как таким протоколам, кроме поддержки соединения, необходимо будет поддерживать синхронизацию часов различных процессоров. При этом возможный способ осуществления успешной атаки может возникнуть, если временно будет отсутствовать синхронизация часов одного из участников. В результате различной и непредсказуемой природы сетевых задержек распределенные часы не могут поддерживать точную синхронизацию. Следовательно, процедуры, основанные на любых отметках времени, должны допускать окно времени, достаточно большое для приспособления к сетевым задержкам, и достаточно маленькое для минимизации возможности атак.

С другой стороны, подход запрос/ответ не годится для приложений, не устанавливающих соединения, так как он требует предварительного рукопожатия перед началом передач, тем самым отвергая основное свойство транзакции без установления соединения. Для таких приложений доверие к некоторому безопасному серверу часов и постоянные попытки каждой из частей синхронизировать свои часы с этим сервером может быть оптимальным подходом.

Использование симметричного шифрования

Для обеспечения аутентификации и распределения ключа сессии часто используется двухуровневая иерархия ключей симметричного шифрования. В общих чертах эта стратегия включает использование доверенного центра распределения ключей (KDC). Каждый участник разделяет секретный ключ, называемый также мастер-ключом, с KDC. KDC отвечает за создание ключей, называемых ключами сессии, и за распределение этих ключей с использованием мастер-ключей. Ключи сессии применяются в течение короткого времени для шифрования только данной сессии между двумя участниками.

Большинство алгоритмов распределения секретного ключа с использованием KDC, включает также возможность аутентификации участников.

Протокол Нидхэма и Шредера

Предполагается, что секретные мастер-ключи K_A и K_B разделяют соответственно А и KDC и В и KDC. Целью протокола является безопасное распределение ключа сессии KS между А и В. Протокол представляет собой следующую последовательность шагов:

1. $A \quad KDC: \ ID_A \,||\, ID_B \,||\, N_1$

2. $KDC \quad A: \ E_{Ka}\,[K_S \,||\, ID_B \,||\, N_1 \,||\, E_{Kb}\,[K_S \,||\, ID_A]\,]$

3. $A \quad B: \ E_{Kb}\,[K_S \,||\, ID_A]$

4. $B \quad A: \ E_{KS}\,[N_2]$

5. $A \quad B: \ E_{KS}\,[f\,(N_2)]$

1. А запрашивает у KDC ключ сессии для установления защищенного соединения с В. Сообщение включает идентификацию А и В и уникальный идентификатор данной транзакции, который обозначен как N_1 и называется nonce. Nonce может быть временной меткой, счетчиком или случайным числом; минимальное требование состоит в том, чтобы он отличался для каждого запроса. Кроме того, для предотвращения подделки желательно, чтобы противнику было трудно предугадать nonce. Таким образом, случайное число является лучшим вариантом для nonce.
2. KDC отвечает сообщением, зашифрованным ключом K_A. Таким образом, только А может расшифровать сообщение, и А уверен, что оно получено от KDC, так как предполагается, что кроме А и KDC этот ключ не знает никто. Это сообщение включает следующие элементы, предназначенные для А:
   - Одноразовый ключ сессии.
   - Идентификатор В.
   - nonce, который идентифицирует данную сессию .

А должен убедиться, что полученный nonce равен значению nonce из первого запроса. Это доказывает, что ответ от KDC не был модифицирован при пересылке и не является повтором некоторого предыдущего запроса. Кроме того, сообщение включает два элемента, предназначенные для В:

- Одноразовый ключ сессии K_S.
- Идентификатор A ID_A.

Эти два последних элемента шифруются мастер-ключом, который KDC разделяет с B. Они посылаются B при установлении соединения и доказывают идентификацию A.

3. A сохраняет у себя ключ сессии и передает B информацию от KDC, предназначенную B: E_{Kb} [K_S || ID_A]. Так как эта информация зашифрована K_B, она защищена от просмотра. Теперь B знает ключ сессии (K_S), знает, что другим участником является A, (ID_A) и что начальная информация передана от KDC, т.к. она зашифрована с использованием K_B.

В этой точке ключ сессии безопасно передан от A к B, и они могут начать безопасный обмен. Тем не менее, существует еще два дополнительных шага:

4. Используя созданный ключ сессии, B пересылает A nonce N_2.
5. Также используя K_S, A отвечает f (N_2), где f - функция, выполняющая некоторую модификацию N_2.

Эти шаги гарантируют B, что сообщение, которое он получил, не изменено и не является повтором предыдущего сообщения.

Заметим, что реальное распределение ключа включает только шаги 1 - 3, а шаги 4 и 5, как и 3, выполняют функцию аутентификации.

A безопасно получает ключ сессии на шаге 2. Сообщение на шаге 3 может быть дешифровано только B. Шаг 4 отражает знание B ключа K_S, и шаг 5 гарантирует B знание участником A ключа K_S и подтверждает, что это не устаревшее сообщение, так как используется nonce N_2. Шаги 4 и 5 призваны предотвратить общий тип replay-атак. В частности, если противник имеет возможность захватить сообщение на шаге 3 и повторить его, то это должно привести к разрыву соединения.

Разрывая рукопожатие на шагах 4 и 5, протокол все еще уязвим для некоторых форм атак повторения. Предположим, что противник X имеет возможность скомпрометировать старый ключ сессии. Маловероятно, чтобы противник мог сделать больше, чем просто копировать сообщение шага 3. Потенциальный риск состоит в том, что X может заставить взаимодействовать A и B, используя старый ключ сессии. Для этого X просто повторяет сообщение шага 3, которое было перехвачено ранее и содержит скомпрометированный ключ сессии. Если B не запоминает идентификацию всех предыдущих ключей сессий с A, он не сможет определить, что это повтор. Далее X должен перехватить сообщение рукопожатия на шаге 4 и представиться A в ответе на шаге 5.

Протокол Деннинга

Деннинг предложил преодолеть эту слабость модификацией протокола Нидхэма и Шредера, которая включает дополнительную отметку времени на шагах 2 и 3:

1. A -> KDC: ID_A || ID_B
2. KDC -> A: E_{Ka} [K_S || ID_B || T || E_{Kb} [K_S || ID_A || T]]
3. A -> B: E_{Kb} [K_S || ID_A || T]
4. B -> A: E_{KS} [N_1]
5. A -> B: E_{KS} [f (N_1)]

T - это отметка времени, которая гарантирует A и B, что ключ сессии является только что созданным. Таким образом, и A, и B знают, что распределенный ключ не является старым. A и B могут верифицировать временную отметку проверкой, что

$|Clock - T| < \Delta t_1 + \Delta t_2$

где Δt_1 - оцениваемое нормальное расхождение между часами KDC и локальными часами (у A или B) и t_2 - ожидаемая сетевая задержка времени. Каждый участник может установить свои часы, ориентируясь на определенный доверенный источник. Поскольку временная отметка T шифруется с использованием секретных мастер-ключей, взломщик, даже зная старый ключ сессии, не сможет достигнуть цели повторением шага 3 так, чтобы B не заметил искажения времени.

Шаги 4 и 5 не были включены в первоначальное представление, но были добавлены позднее. Эти шаги подтверждают A, что B получил ключ сессии.

Протокол Деннинга обеспечивает большую степень безопасности по сравнению с протоколом Нидхэма и Шредера. Однако данная схема требует доверия к часам, которые должны быть синхронизированы в сети. В этом есть определенный риск, который состоит в том, что распределенные часы могут рассинхронизироваться в результате диверсии или повреждений. Проблема возникает, когда часы отправителя спешат по отношению к часам получателя. В этом случае противник может перехватить сообщение от отправителя и повторить его позднее, когда отметка времени в сообщении станет равной времени на узле получателя. Это повторение может иметь непредсказуемые последствия.

Один способ вычисления атак повторения состоит в требовании, чтобы участники регулярно сверяли свои часы с часами KDC. Другая альтернатива, при которой нет необходимости всем синхронизировать часы, состоит в доверии протоколам рукопожатия, использующим nonce.

Протокол аутентификации с использованием билета

Данный протокол пытается преодолеть проблемы, возникшие в предыдущих двух протоколах. Он выглядит следующим образом:

1. A -> B: ID_A || N_a
2. B -> KDC: ID_B || N_b || E_{Kb} [ID_A || N_a || T_b]
3. KDC -> A: EK_a [ID_B || N_a || K_S || T_b] || E_{Kb} [ID_A || K_S || T_b] || N_b
4. A -> B: E_{Kb} [ID_A || K_S || T_b] || E_{KS} [N_b]

1. A инициализирует аутентификационный обмен созданием nonce N_a и посылкой его и своего идентификатора к B в незашифрованном виде. Этот nonce вернется к A в зашифрованном сообщении, включающем ключ сессии, гарантируя A, что ключ сессии не старый.
2. B сообщает KDC, что необходим ключ сессии. Это сообщение к KDC включает идентификатор B и nonce N_b. Данный nonce вернется к B в зашифрованном сообщении, которое включает ключ сессии, гарантируя B, что ключ сессии не устарел. Сообщение B к KDC также включает блок, зашифрованный секретным ключом, разделяемым B и KDC. Этот блок используется для указания KDC, когда

заканчивается время жизни данного ключа сессии. Блок также специфицирует намеченного получателя и содержит nonce, полученный от A. Этот блок является своего рода "верительной грамотой" или "билетом" для A.

3. KDC получил nonces от A и B и блок, зашифрованный секретным ключом, который B разделяет с KDC. Блок служит билетом, который может быть использован A для последующих аутентификаций. KDC также посылает A блок, зашифрованный секретным ключом, разделяемым A и KDC. Этот блок доказывает, что B получил начальное сообщение A (ID_B), что в нем содержится допустимая отметка времени и нет повтора (N_a). Этот блок обеспечивает A ключом сессии (K_S) и устанавливает ограничение времени на его использование (T_b).

4. A посылает полученный билет B вместе с nonce B, зашифрованным ключом сессии. Этот билет обеспечивает B ключом сессии, который тот использует для дешифрования и проверки nonce. Тот факт, что nonce B расшифрован ключом сессии, доказывает, что сообщение пришло от A и не является повтором.

Данный протокол аутентифицирует A и B и распределяет ключ сессии. Более того, протокол предоставляет в распоряжение A билет, который может использоваться для его последующей аутентификации, исключая необходимость повторных контактов с аутентификационным сервером. Предположим, что A и B установили сессию с использованием описанного выше протокола и затем завершили эту сессию. Впоследствии, но до истечения лимита времени, установленного протоколом, A может создать новую сессию с B. Используется следующий протокол:

1. A -> B: E_{Kb} [ID_A || K_S || T_b], N_a'
2. B -> A: N_b', E_S [N_a']
3. A -> B: E_S [N_b']

Когда B получает сообщение на шаге 1, он проверяет, что билет не просрочен. Заново созданные nonces N_a' и N_b' гарантируют каждому участнику, что не было атак повтора. Время T_b является временем относительно часов B. Таким образом, эта временная метка не требует синхронизации, потому что B проверяет только им самим созданные временные отметки.

Использование шифрования с открытым ключом

Протокол аутентификации с использованием аутентификационного сервера.

Рассмотрим протокол, использующий отметки времени и аутентификационный сервер:

1. A -> AS: ID_A || ID_B
2. AS -> A: E_{KRas} [ID_A || KU_a || T] ||
   E_{KRas} [ID_B || KU_b || T]
3. A -> B: E_{KRas} [ID_A || KU_a || T] ||
   E_{KRas} [ID_B || KU_b || T] ||
   E_{KUb} [E_{KRa} [K_S || T]]

В данном случае третья доверенная сторона является просто аутентификационным сервером AS, потому что третья сторона не создает и не распределяет секретный ключ. AS просто обеспечивает сертификацию открытых ключей участников. Ключ сессии выбирается и шифруется A, следовательно, не существует риска, что AS взломают и заставят распределять скомпрометированные ключи сессии. Отметки времени защищают от повтора скомпрометированных ключей сессии.

Данный протокол компактный, но, как и прежде, требует синхронизации часов.

Протокол аутентификации с использованием KDC

Другой подход использует nonces. Этот протокол состоит из следующих шагов:

1. A -> KDC: ID_A || ID_B
2. KDC -> A: E_{KRkdc} [ID_B || KU_b]
3. A -> B: E_{KUb} [N_a || ID_A]
4. B -> KDC: ID_B || ID_A || E_{KUkdc} [N_a]
5. KDC -> B: E_{KRkdc} [ID_A || KU_a] ||
   E_{KUb} [E_{KRkdc} [N_a || K_S || ID_B]]
6. B -> A: E_{KUa} [E_{KRkdc} [N_a || K_S || ID_B] || N_b]
7. A -> B: E_{KS} [N_b]

На первом шаге A информирует KDC, что хочет установить безопасное соединение с B. KDC возвращает A сертификат открытого ключа B (шаг 2). Используя открытый ключ B, A информирует B о создании защищенного соединения и посылает nonce N_a (шаг 3). На 4-м шаге B спрашивает KDC о сертификате открытого ключа A и запрашивает ключ сессии. B включает nonce A, чтобы KDC мог пометить ключ сессии этим nonce. Nonce защищен использованием открытого ключа KDC. На 5-м шаге KDC возвращает B сертификат открытого ключа A плюс информацию {N_a, K_S, ID_B}. Эта информация означает, что K_S является секретным ключом, созданным KDC в интересах B и связан с N_a. Связывание K_S и N_a гарантирует A, что K_S не устарел. Эта тройка шифруется с использованием закрытого ключа KDC, это гарантирует B, что тройка действительно получена от KDC. Она также шифруется с использованием открытого ключа B, чтобы никто другой не мог подсмотреть ключ сессии и использовать эту тройку для установления соединения с A. На шаге 6 тройка {N_a, K_S, ID_B}, зашифрованная закрытым ключом KDC, передается A вместе с nonce N_b, созданным B. Все сообщение шифруется открытым ключом A. A восстанавливает ключ сессии K_S, использует его для шифрования N_b, который возвращает B. Это последнеее сообщение гарантирует B, что A знает ключ сессии.

Это достаточно безопасный протокол при различного рода атаках. Однако авторы предложили пересмотренную версию данного алгоритма:

1. A -> KDC: ID_A || ID_B
2. KDC -> A: E_{KRauth} [ID_B || KU_b]
3. A -> B: E_{KUb} [N_a || ID_A]
4. B -> KDC: ID_B || ID_A || E_{KUauth} [N_a]
5. KDC -> B: E_{KRauth} [ID_A || KU_a] || E_{KUb} [
   E_{KRauth} [N_a || K_S || ID_A || ID_B]]
6. B -> A: E_{KUa} [E_{KRauth} [N_a || K_S ||
   ID_A || ID_B] || N_b]
7. A -> B: E_{KS} [N_b]

Добавляется идентификатор A ID_A к данным, зашифрованных с использованием закрытого ключа KDC на шагах 5 и 6 для идентификации обоих участников сессии. Это включение ID_A приводит к тому, что значение

nonce N_a должно быть уникальным только среди всех nonces, созданных A, но не среди nonces, созданных всеми участниками. Таким образом, пара $\{ID_A, N_a\}$ уникально идентифицирует соединение, созданное A.

Односторонняя аутентификация

Существует специфическое приложение - электронная почта, для которого шифрование также имеет большое значение. Особенность e-mail состоит в том, что отправителю и получателю нет необходимости быть на связи в одно и то же время. Вместо этого сообщения направляются в почтовый ящик получателя, где они хранятся до тех пор, пока у того не появится возможность получить их.

Заголовок сообщения должен быть незашифрованным, чтобы сообщение могло пересылаться протоколами e-mail, такими как X.400 или SMTP. Однако желательно, чтобы протоколы управления почтой не имели бы доступа к самому сообщению. Соответственно, e-mail сообщение должно быть зашифровано так, чтобы система управления почтой могла бы не знать ключ шифрования.

Вторым требованием является аутентификация сообщения. Обычно получателю нужна определенная гарантия того, что сообщение пришло от законного отправителя.

Использование симметричного шифрования

При использовании симметричного шифрования сценарий централизованного распределения ключей в полном объеме непригоден. Эти схемы требуют, чтобы в двух заключительных шагах отправитель посылал запрос получателю, ожидая ответа с созданным ключом сессии, и только после этого отправитель может послать сообщение.

С учетом перечисленных ограничений протоколы использования KDC являются возможными кандидатами для шифрования электронной почты. Для того чтобы избежать требования к получателю B находиться на связи в то же самое время, когда и отправитель A, шаги 4 и 5 должны быть опущены. Таким образом, остается последовательность шагов:

1. A -> KDC: $ID_A \| ID_B \| N_1$
2. KDC -> A: $E_{Ka} [K_S \| ID_B \| N_1 \| E_{Kb} [K_S \| ID_A]]$
3. A -> B: $E_{Kb} [K_S , ID_A] \| E_{KS} [M]$

Данный подход гарантирует, что только требуемый получатель сообщения сможет прочитать его. Это также обеспечивает определенный уровень аутентификации, что отправителем является A. Очевидно, что протокол не защищает от атак повтора. Некоторая мера защиты может быть обеспечена включением отметки времени в сообщение. Однако поскольку существуют потенциальные задержки в процессе передачи e-mail сообщений, такие временные отметки имеют ограниченный срок действия.

Использование шифрования с открытым ключом

При использовании шифрования с открытым ключом требуется, чтобы отправитель знал открытый ключ получателя (для обеспечения конфиденциальности), получатель знал открытый ключ отправителя (для обеспечения аутентификации), или и то, и другое (для обеспечения конфиденциальности и аутентификации).

Если требуется конфиденциальность, то может быть использована следующая схема:

A -> B: $E_{KUb} [K_S] \| E_{KS} [M]$

В этом случае сообщение шифруется одноразовым секретным ключом. A шифрует этот одноразовый ключ открытым ключом B. Только B имеет соответствующий закрытый ключ для получения одноразового ключа и использования этого ключа для дешифрования сообщения. Эта схема более эффективна, чем простое шифрование всего сообщения открытым ключом B.

Если требуется аутентификация, то цифровая подпись может быть создана по такой схеме:

A -> B: $M \| E_{KRa} [H (M)]$

Этот метод гарантирует, что A не сможет впоследствии отвергнуть полученное сообщение. Однако данная технология открыта для другого типа подделок. Например, можно получить доступ к почтовой очереди перед доставкой, вырезать подпись отправителя, вставить свою и опять поставить сообщение в очередь на доставку.

Чтобы этого не допустить, сообщение и подпись можно зашифровать ключом симметричного шифрования, который в свою очередь шифруется открытым ключом получателя:

A -> B: $E_{KS} [M \| E_{KRa} [H (M)]] \| E_{KUb} [K_S]$

Следующая схема не требует, чтобы B знал открытый ключ A. В этом случае должен использоваться сертификат открытого ключа:

A -> B: $M \| E_{KRa} [H (M)] \| E_{KRas} [T \| ID_A \| KU_a]$

В конце сообщения A посылает B подпись, зашифрованную закрытым ключом A, и сертификат A, зашифрованный закрытым ключом аутентификационного сервера. Получатель применяет сертификат для получения открытого ключа отправителя и затем использует открытый ключ отправителя для проверки самого сообщения. Конфиденциальность может быть добавлена аналогично предыдущей схеме.

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование лабораторной работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**3.**	Программирование арифметических алгоритмов	5	Работа в малых группах (2 часа)
2	**3.**	Программирование алгебраических алгоритмов	5	Работа в малых группах (1 часа)
3	**2.**	Защита от закладок при разработке программ	5	Работа в малых группах (1 часа)
4	**4.**	Программирование алгоритмов криптосистем с открытым ключом	5	Работа в малых группах (1 часа)
5	**2.**	Профилактика заражения вирусами компьютерных систем	4	Работа в малых группах (1 часа)
		ИТОГО	**24**	**6**

4.4. Практические занятия

№ п/п	*Номер раздела дисциплины*	*Наименование практической работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**4.**	Криптографические методы защиты	2	Работа в малых группах (1 часа)
2	**4.**	Шифрование методом IDEA	2	Работа в малых группах (1 часа)
3	**4.**	Шифрование методом RC6	2	Работа в малых группах (1 часа)
4	**4.**	Шифрование методом Джиффорда	2	Работа в малых группах (1 часа)
5	**4.**	Шифрование методом аналитических преобразований	2	-
6	**4.**	Сокрытие информации методом стеганографии	2	-
		ИТОГО	**12**	**4**

4.5. Контрольные мероприятия: контрольная работа

Цель: Развить навыки студентов по использованию приобретённых знаний в ответах на конкретные вопросы.

Структура: Каждое индивидуальное задание предполагает ответ студента на десять вопросов по 5 темам:

- Основные понятия информационной безопасности.
- Правовое обеспечение защиты информации
- Организационное обеспечение защиты информации
- Инженерно-техническое обеспечение защиты информации
- Программно-аппаратные методы защиты информации

Основная тематика: Информационная безопасность сетей и систем.

Рекомендуемый объем: Пояснительная записка объемом 15 - 20 страниц должна содержать титульный лист, задание, ответ на заданные вопросы, список используемой литературы.

Выдача задания, прием кр проводиться в соответствии с календарным учебным графиком

Оценка	Критерии оценки контрольной работы
Зачтено	Во время защиты контрольной работы студент демонстрирует знание всех поставленных вопросов, уверенное использование источников, владение достаточным уровнем понимания материала, и способностью самостоятельно высказать мысль на научно-техническом языке.
Не зачтено	Во время защиты контрольной работы студент показал не полное понимание материала или неправильно ответил на поставленные вопросы.

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*		*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ОПК	*ПК*				
		1	*17*				
1	2	3	4	5	6	7	8
1. Информационная безопасность.	6	+	+	2	3	Лк, СРС	ЭКЗАМЕН, кр
2. Защита информации при помощи криптографии.	16	+	+	2	8	Лк, ЛР, СРС	ЭКЗАМЕН, кр
3. Классификация шифров.	17	+	+	2	8,5	Лк, ЛР, СРС	ЭКЗАМЕН, кр
4. Криптосистемы	25	+	+	2	12,5	Лк, ЛР,ПЗ, СРС	ЭКЗАМЕН, кр
5. Современные криптографические системы	8	+	+	2	4	Лк, СРС	ЭКЗАМЕН, кр
6. Теория сложности вычислений	6	+	+	2	2	Лк, СРС	ЭКЗАМЕН, кр
7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма	7	+	+	2	3,5	Лк, СРС	ЭКЗАМЕН, кр
8. Электронная подпись.	8	+	+	2	4	Лк, СРС	ЭКЗАМЕН, кр
9. Хеш-функции	7	+	+	2	3,5	Лк, СРС	ЭКЗАМЕН, кр
10. Аутентификация и обмен ключами.	8	+	+	2	4	Лк, СРС	ЭКЗАМЕН, кр
всего часов	**108**	**54**	**54**	**2**	**54**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Степанов, Е. А. Информационная безопасность и защита информации : учебное пособие / Е. А. Степанов, И. К. Корнеев. – Москва : Инфра-М, 2001. – 304 с. (страницы 134-170)
2. Информационные технологии : учебник / Под ред. В. В. Трофимова. – Москва : Юрайт, 2011. – 624 с. (страницы 200-350)

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	***Вид занятия*** (*Лк, ЛР, ПЗ, КП, КР, кр*)	***Количество экземпляров в библиотеке*, шт.**	***Обеспеченность*, (экз./ чел.)**
1	**2**	**3**	**4**	**5**
Основная литература				
1.	Правовое обеспечение информационной безопасности : учебное пособие для вузов / Под ред. С. Я. Казанцева. – 2-е изд., испр. И доп. – Москва : Академия, 2007. – 240 с.	ЛК, кр	15	1
2.	Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.	ЛК, ЛР, ПЗ, кр	31	1
3	Нестеров, С.А. Основы информационной безопасности : учебное пособие / С.А. Нестеров ; Министерство образования и науки Российской Федерации, Санкт-Петербургский государственный политехнический университет. – СПб. : Издательство Политехнического университета, 2014. – 322 с. : схем., табл., ил. – ISBN 978-5-7422-4331-1 ; То же [Электронный ресурс]. – URL: //biblioclub.ru/index.php?page=book&id=363040 (11.04.2017).	ЛК	ЭР	1
4	Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.	ЛК, ЛР, ПЗ, кр	16	1
Дополнительная литература				
5	Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.	ЛК, ЛР, ПЗ, кр	5	0,3
6	Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.	ЛК, ЛР, ПЗ, кр	5	0,3

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО – ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1.Электронный каталог библиотеки БрГУ http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=.

2. Электронная библиотека БрГУ
http://ecat.brstu.ru/catalog .

3. Электронно-библиотечная система «Университетская библиотека online» http://biblioclub.ru .

4. Электронно-библиотечная система «Издательство «Лань»
http://e.lanbook.com .

5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru .

6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru .

7. Университетская информационная система РОССИЯ (УИС РОССИЯ) https://uisrussia.msu.ru/ .

8. Национальная электронная библиотека НЭБ
http://xn--90ax2c.xn--p1ai/how-to-search /.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

9.1. Методические указания для обучающихся по выполнению лабораторных работ/практических работ

Лабораторная работа №1

Программирование арифметических алгоритмов

Цель работы:

Исследование и разработка основных методов симметричных криптосистем.

Задание:

1. Изучить основы криптографии
2. Познакомиться с некоторыми арифметическими алгоритмами криптографии.

Порядок выполнения:

1. Изучить теоретические основы.
2. Изучить шифр перестановки.
3. Изучить шифр одиночной перестановки.
4. Изучить шифр двойной перестановки.
5. Изучить шифрование при помощи магического квадрата.

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. В чем заключается главная задача криптографии?
2. В чем особенность шифра перестановки?
3. В чем особенность шифра одиночной перестановки?
4. В чем особенность шифра двойной перестановки?
5. В чем особенность шифрования при помощи магического квадрата?

Лабораторная работа №2

Программирование алгебраических алгоритмов

Цель работы:

Исследование и разработка классических методов симметричных криптосистем

Задание:

1. Изучить принцип работы симметричных криптосистем.
2. Познакомиться с некоторыми алгебраическими способами криптографии..

Порядок выполнения:

1. Изучить теоретические основы
2. Изучить шифр простой замены. Система шифрования Цезаря.
3. Изучить шифр сложной замены. Шифр Гронсфельда.
4. Изучить шифр многоалфавитной замены.
5. Изучить способ Гаммирование

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Шифр Гронсфельда.
2. Шифры двойной перестановки. Шифрование с помощью магического квадрата.
3. Шифр многоалфавитной замены и алгоритм его реализации.

Лабораторная работа №3

Защита от закладок при разработке программ

Цель работы:

Исследование и анализ служебных программ Windows для повышения эффективности работы компьютера.

Задание:

1. Установите проверку подлинности доступа к ресурсам компьютера из локальной сети. Запретите доступ к ресурсам вашего компьютера из Интернета.
2. Разрешить удаленный доступ к ресурсам вашего компьютера.
3. Использование удаленного доступа к сетевым ресурсам.
4. Защита и восстановление данных на компьютере.

Порядок выполнения:

1. Установите проверку подлинности доступа к ресурсам компьютера из локальной сети. Запретите доступ к ресурсам вашего компьютера из Интернета.
2. Разрешить удаленный доступ к ресурсам вашего компьютера.
3. Использование удаленного доступа к сетевым ресурсам.
4. Защита и восстановление данных на компьютере.

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая

линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

10. Почему при эксплуатации компьютерной системы важно знать ее параметры?
11. Какие стандартные средства Windows XP обеспечивают пользователю возможность определения параметров компьютерной системы?
12. Почему обеспечение бесперебойной работы дисковой системы компьютера является одной из основных мер обеспечения информационной безопасности?
13. Опишите причины нарушений в работе магнитных дисков.
14. Почему необходима процедура очистки диска?
15. Что такое фрагментация файла? Почему она возникает и как влияет на скорость операций чтения информации с диска?
16. В каких случаях рекомендуется выполнить дефрагментацию диска?
17. С какой целью выполняется архивация данных компьютера?
18. Что такое дискета аварийного восстановления? Какой программой она создается?

Лабораторная работа №4

Программирование алгоритмов криптосистем с открытым ключом

Цель работы:

Исследование и анализ основных методов асимметричных криптосистем

Задание:

1. Изучить шифрования Эль Гамаля.
2. Изучить шифрование данных RSA.

Порядок выполнения:

1. Зашифровать простое сообщение способом Эль Гамаля.
2. Зашифровать текст при помощи шифрования данных RSA.

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая

линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Алгоритм шифрации двойным квадратом. Шифр Enigma.
2. Алгоритм шифрования DES.
3. Алгоритм шифрования ГОСТ 28147-89.
4. Алгоритм шифрования RSA.
5. Алгоритм шифрования Эль Гамаля.
6. Задачи и алгоритмы электронной подписи.
7. Задачи распределения ключей.

Лабораторная работа №5

Профилактика заражения вирусами компьютерных систем

Цель работы:

Анализ и исследование антивирусных программ.

Задание:

1. Познакомиться с работой антивирусных программ.

Порядок выполнения:

1. Изучить теоретические основы.
2. Ознакомится с энциклопедией компьютерных вирусов на сайте лаборатории Касперского.
3. Проверить на присутствие вирусов всех критических областей компьютера.
4. Поиск вирусов на вашем компьютере с тщательной проверкой всех подключенных дисков, памяти, файлов .
5. Проверка на присутствие вирусов объектов, загрузка которых осуществляется при старте операционной системы.
6. Поиск на компьютере руткитов, обеспечивающих сокрытие вредоносных программ в операционной системе.
7. Изучить дополнительные возможности программы Norton AntiVirus по защите данных.

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Что такое компьютерный вирус? Какими свойствами обладают компьютерные вирусы?
2. По каким признакам классифицируют компьютерные вирусы? Перечислите типы вирусов.
3. Какие вирусы называются резидентными и в чем особенность таких вирусов?
4. Каковы отличия вирусов-репликаторов, стелс – вирусов, мутантов и «троянских» программ?
5. Опишите схему функционирования загрузочного вируса.
6. Опишите схему функционирования файлового вируса.
7. Опишите схему функционирования загрузочно-файловых вирусов.
8. Что такое полиморфный вирус? Почему этот тип вирусов считается наиболее опасным?

Практическое занятие №1

Криптографические методы защиты

Цель работы:

Познакомиться с криптографическими методами защиты.

Задание:

1. Изучить классификацию криптографических методов защиты.
2. Изучить основные определения.

Порядок выполнения:

Изучить теоретические данные. Изучить классификацию криптографических методов защиты. Изучить основные определения.

Форма отчетности:

Отчет не предусмотрен.

Задания для самостоятельной работы:

Изучение новейших методов криптографической защиты информации.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах :

учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Что такое шифрование?
2. Чем отличается открытый ключ от закрытого?
3. Дать определение стеганографии.

Практическое занятие №2
Шифрование методом IDEA

Цель работы:
Приобрести навыки работы с методом шифрования IDEA.

Задание:
1. Изучить методом шифрования IDEA.

Порядок выполнения:
Изучить теоретические данные. Изучить описание алгоритма. Познакомиться с блок-схемой метода.

Форма отчетности:
Отчет не предусмотрен.

Задания для самостоятельной работы:
Найти современные области применения данного способа шифрования.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Что является входной информацией для шифра IDEA?
2. Сколько нужно циклов шифрования для достаточной степени сокрытия информации?
3. На сколько блоков делиться входная информация?

Практическое занятие №3
Шифрование методом RC6

Цель работы:
Приобрести навыки работы с методом шифрования RC6..

Задание:
1. Познакомиться с шифрованием методом RC6.

Порядок выполнения:
Изучить теоретические данные. Изучить описание алгоритма. Познакомиться с блок-схемой метода. Изучить процедуры: шифрования, дешифрования и генерации ключа.

Форма отчетности:
Отчет не предусмотрен.

Задания для самостоятельной работы:
Найти современные области применения данного способа шифрования.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Что является входной информацией для шифра RC6?
2. Сколько нужно циклов шифрования для достаточной степени сокрытия информации?
3. На сколько блоков делиться входная информация?

Практическое занятие №4

Шифрование методом Джиффорда.

Приобрести навыки работы с методом шифрования Джиффорда

Задание:

1. Познакомиться с шифрованием методом Джиффорда.

Порядок выполнения:
Изучить теоретические данные. Изучить описание алгоритма. Познакомиться с блок-схемой метода.

Форма отчетности:
Отчет не предусмотрен.

Задания для самостоятельной работы:
Найти современные области применения данного способа шифрования.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Принцип действия метода шифрования Джиффорда?
2. Для каких целей использовался данный метода шифрования?
3. В каком году он был взломан?.

Практическое занятие №5

Шифрование методом аналитических преобразований.

Цель работы:

Познакомиться с методами аналитически преобразования для сокрытия информации..

Задание:

1. Познакомиться с методами аналитических преобразований.

Порядок выполнения:

Изучить теоретические данные. Изучить описание алгоритмов.

Форма отчетности:

Отчет не предусмотрен.

Задания для самостоятельной работы:

Найти современные области применения данного способа шифрования.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. Дать определение аналитическим преобразованиям.

2. В каких случаях могут быть использованы аналитические преобразования?
3. К какой информации этот вид шифрования может быть применим?

Практическое занятие №6

Сокрытие информации методом стеганографии

Цель работы:

Познакомиться с методами стеганографии.

Задание:

1. Познакомиться с методами аналитических преобразований.

Порядок выполнения:

Изучить теоретические данные. Изучить описание современных методов стеганографии.

Форма отчетности:

Отчет не предусмотрен.

Задания для самостоятельной работы:

Найти современные области применения данного способа шифрования.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
2. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

Контрольные вопросы для самопроверки

1. В чем главный принцип стенографии?
2. В каких «контейнерах» может быть спрятано зашифрованное сообщение?
3. Привести примеры стеганографии древних времён.

19. Методические указания по выполнению контрольной работы

Работа посвящена более глубокому изучению дисциплины. Для этого студентам предлагает ответить на ряд вопросов по следующим темам:

1. Основные понятия информационной безопасности. Основы защиты информации.
2. Правовое обеспечение защиты информации.

3. Организационное обеспечение защиты информации.
4. Инженерно-техническое обеспечение защиты информации.
5. Программно-аппаратные методы защиты информации

Выбор контрольных вопросов производится по варианту студента. Каждому студенту необходимо ответить на 10 вопросов.

Основная литература

1. Правовое обеспечение информационной безопасности : учебное пособие для вузов / Под ред. С. Я. Казанцева. – 2-е изд., испр. И доп. – Москва : Академия, 2007. – 240 с.
2. Иванов, М. Ю. Информационные технологии: методы криптографии : учебное пособие / М. Ю. Иванов. – Братск : БрГУ, 2010. – 100 с. – Б. ц.
3. Новожилов, О. П. Информатика : учебное пособие / О. П. Новожилов. – 2-е изд., испр. И доп. – М. : Юрайт, 2012. – 564 с.

Дополнительная литература

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учебное пособие / П. Н. Девянин. – М. : Горячая линия- Телеком, 2012. – 320 с.
2. Малюк, А. А. Введение в защиту информации в автоматизированных системах : учебное пособие / А. А. Малюк. – 4-е изд., стереотип. – М. : Горячая линия- Телеком, 2011. – 146 с.

20. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Информационно-коммуникационные технологии (ИКТ) – преподаватель использует для:

- получения информации при подготовке к занятиям,
- создания презентационного сопровождения лекций;
- интерактивного общения;
- пакет прикладных программ (Microsoft,)
- ОС Windows; OpenOffice; LibreOffice и д.р.;

21. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	*Наименование аудитории*	*Перечень основного оборудования*	*№ ЛР или ПЗ*
1	3	4	5
ЛР	Лекционный кабинет/дисплейный класс	AMD Athlon 64 (5GHz/250Gb/2Gb/DD-RW), 2 ядра	ЛР 1-5
ПЗ	Лекционный кабинет/дисплейный класс	AMD Athlon 64 (5GHz/250Gb/2Gb/DD-RW), 2 ядра	ПЗ 1-6
кр	Лекционный кабинет/дисплейный класс	AMD Athlon 64 (5GHz/250Gb/2Gb/DD-RW), 2 ядра	-
СР	ЧЗЗ	-	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-1	Способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной тайны.	**1.**. Информационная безопасность	1.1. Введение в информационную безопасность	Экзаменационный билет
		2. Защита информации при помощи криптографии	2.1. Защита передаваемых и хранимых секретных данных от разглашения и искажения	Экзаменационный билет
		3. Классификация шифров	3.1. Перестановочные шифры	Экзаменационный билет
		4. Криптосистемы	4.1. Устройство шифров	Экзаменационный билет
			4.3. Алгоритм DES	Экзаменационный билет
		5. Современные криптографические системы	5.1. Предварительные математические понятия	Экзаменационный билет
			5.3. Спецификация алгоритма	Экзаменационный билет
		6. Теория сложности вычислений	6.1. Сложность вычислений.	Экзаменационный билет
		7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма	7.1. Алгоритм RSA	Экзаменационный билет
		8. Электронная подпись.	8.1. Общая схема цифровой подписи	Экзаменационный билет
			8.3. Электронная подпись на основе алгоритма RSA	Экзаменационный билет
		9. Хеш-функции	9.1. Требования к хэш-функциям	Экзаменационный билет
		10. Аутентификация и обмен ключами.	10.1. Алгоритмы распределения ключей с использованием третьей доверенной стороны Понятие мастер-ключа	Экзаменационный билет
			10.3. Использование шифрования с открытым ключом	Экзаменационный билет
ПК-17	Способность применять современные теоретические и эксперимental	**1.** Информационная безопасность	1.2. Модель сетевой безопасности .Классификация сетевых атак	Экзаменационный билет
		2. Защита информации при	2.2. Задача подтверждения авторства сообщения	Экзаменационный билет

	ьные методы исследования с целью создания новых перспективных средств электросвязи и информатики	помощи криптографии		
		3. Классификация шифров	3.2. Классификация методов дешифрования. Модель предполагаемого противника. Правила Керкхоффа	Экзаменационный билет
		4. Криптосистемы	4.2. Поточные шифры	Экзаменационный билет
			4.4. Алгоритм ГОСТ 28147	Экзаменационный билет
		5. Современные криптографические системы	5.2. Обоснование разработки	Экзаменационный билет
			5.4. Преимущества алгоритма	Экзаменационный билет
		6. Теория сложности вычислений	6.2. Сложность алгоритмов	Экзаменационный билет
		7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма	7.2. Криптосистема Эль-Гамаля	Экзаменационный билет
		8. Электронная подпись.	8.2. Стандарт цифровой подписи DSS	Экзаменационный билет
		9. Хеш-функции	9.2. Простые хэш-функции	Экзаменационный билет
		10. Аутентификация и обмен ключами.	10.2. Протоколы аутентификации	Экзаменационный билет

2. Экзаменационные вопросы

№ п/п	**Компетенции**		**ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ**	**№ и наименование раздела** (
	Код	**Определение**		
1	2	3	4	5
1	ОПК-1	Способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать	**1.** Введение в информационную безопасность	**1.**. Информационная безопасность
			1. Защита передаваемых и хранимых секретных данных от разглашения и искажения	**2.** Защита информации при помощи криптографии
			1. Перестановочные шифры	**3.** Классификация шифров
			1. Устройство шифров	**4.** Криптосистемы
			2. Алгоритм DES	
			1. Предварительные математические понятия	**5.** Современные криптографические системы
			2. Спецификация алгоритма	
			1. Сложность вычислений.	**6.** Теория сложности вычислений

		основные требования информационной безопасности, в том числе защиты государственной тайны.	**1.** Алгоритм RSA	**7.** Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма
			1. Общая схема цифровой подписи	**8.** Электронная подпись.
			2. Электронная подпись на основе алгоритма RSA	
			1. Требования к хэш-функциям	**9.** Хеш-функции
			1. Алгоритмы распределения ключей с использованием третьей доверенной стороны Понятие мастер-ключа	**10.** Аутентификация и обмен ключами.
			2. Использование шифрования с открытым ключом	
2	ПК-17	Способность применять современные теоретические и экспериментальные методы исследования с целью создания новых перспективных средств электросвязи и информатики	**1.** Модель сетевой безопасности .Классификация сетевых атак	**1.**. Информационная безопасность
			1. Задача подтверждения авторства сообщения	**2.** Защита информации при помощи криптографии
			1. Классификация методов дешифрования. Модель предполагаемого противника. Правила Керкхоффа	**3.** Классификация шифров
			1. Поточные шифры	**4.** Криптосистемы
			2. Алгоритм ГОСТ 28147	
			1. Обоснование разработки	**5.** Современные криптографические системы
			2. Преимущества алгоритма	
			1. Сложность алгоритмов.	**6.** Теория сложности вычислений
			1. Криптосистема Эль-Гамаля.	**7.** Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма
			1. Стандарт цифровой подписи DSS	**8.** Электронная подпись.
			1. Простые хэш-функции	**9.** Хеш-функции
			1. Протоколы аутентификации	**10.** Аутентификация и обмен ключами.

3. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать (ОПК-1): – основные закономерности передачи информации в инфокоммуникационных системах, основные виды сигналов, используемых в телекоммуникационных системах, особенности передачи различных сигналов по каналам и тракам телекоммуникационных системсистем; (ПК-17): - основы цифровой вычислительной техники, структуры и функционирование локальных вычислительных сетей и глобальной сети Интернет; **Уметь** (ОПК-1): – формулировать основные технические требования к телекоммуникационным сетям и систем; (ПК-17): – оценивать основные проблемы, связанные с эксплуатацией и внедрением новой телекоммуникационной техники **Владеть** (ОПК-1): - начальными навыками разработки программного обеспечения сигнальных процессов и микроконтроллеров. (ПК-17): – начальными навыками отладки с использованием соответствующих отладочных средств программного обеспечения сигнальных процессов и микроконтроллеров;	**отлично**	Студент должен во время ответа показать знания: технологии программной защиты в интернете, основные критерии защищённости, основные способы шифрования информации, основных терминов используемые в научно-технической литературе по программной защиты в интернете. Студент должен иметь навыки владения: использования универсальных программных продуктов на ПК, понимания материала и способности высказывания мыслей на научно-техническом языке. Студент во время ответа должен продемонстрировать умения: использования навыков анализа основных способов шифрования.
	хорошо	Ответ содержит неточности. Дополнительные вопросы требуется, но студент с ними справляется отлично.
	удовлетворительно	Ответил только на один вопрос, либо слабо ответил на оба вопроса. На дополнительные вопросы отвечает неуверенно.
	неудовлетворительно	На оба вопроса студент отвечает неубедительно. На дополнительные вопросы преподавателя также не может ответить.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина технологии программной защиты в интернете направлена на ознакомление со способами информационной защиты в интернете, и их практическим применением в современных системах телекоммуникаций; на получение теоретических знаний и практических навыков различных работы с различными способами шифрования и организации систем информационной безопасности для их дальнейшего использования в практической деятельности.

Изучение дисциплины информатика предусматривает:

- лекции,
- лабораторные работы,
- практические занятия,
- контрольную работу,
- самостоятельную работу студента,

– экзамен.

В ходе освоения раздела 1 «Информационная безопасность» студенты должны изучить: введение в информационную безопасность, модель сетевой безопасности , классификация сетевых атак.

В ходе освоения раздела 2 «Защита информации при помощи криптографии» студенты должны изучить: защита передаваемых и хранимых секретных данных от разглашения и искажения, задача подтверждения авторства сообщения.

В ходе освоения раздела 3 «Классификация шифров» студенты должны изучить: перестановочные шифры, классификация методов дешифрования, модель предполагаемого противника, правила Керкхоффа.

В ходе освоения раздела 4 «Криптосистемы» студенты должны изучить: устройство шифров, поточные шифры, алгоритм DES, алгоритм ГОСТ 28147.

В ходе освоения раздела 5 «Современные криптографические системы» студенты должны изучить: предварительные математические понятия, обоснование разработки, спецификация алгоритма, преимущества алгоритма.

В ходе освоения раздела 6 «Классификация шифров» студенты должны изучить: сложность вычислений, сложность алгоритмов.

В ходе освоения раздела 7 «Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма» студенты должны изучить: алгоритм RSA, криптосистема Эль-Гамаля.

В ходе освоения раздела 8 «Электронная подпись» студенты должны изучить: общая схема цифровой подписи, стандарт цифровой подписи DSS, электронная подпись на основе алгоритма RSA

В ходе освоения раздела 9 «Хеш-функции» студенты должны изучить: требования к хэш-функциям, простые хэш-функции.

В ходе освоения раздела 10 «Классификация шифров» студенты должны изучить: алгоритмы распределения ключей с использованием третьей доверенной стороны, понятие мастер-ключа, протоколы аутентификации, использование шифрования с открытым ключом

В процессе проведения лабораторных работ происходит закрепление знаний, формирование умений и навыков реализации представления об различных способах шифрования информации.

В процессе проведения практических работ происходит закрепление знаний, формирование умений и навыков использования продвинутых методов шифрования и сокрытия информации

При подготовке к экзамену рекомендуется особое внимание уделить следующим вопросам: устройство шифров, общая схема цифровой подписи.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературой. Дополнительные сведения по изучаемым темам можно найти в периодической печати и Интернете.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины
Технологии программной защиты в Интернете

1. Цель и задачи дисциплины

Целью изучения дисциплины является формирование у обучающихся профессиональных компетенций в области построения и функционирования сетей передачи данных, базовых технологий организации локальных и территориальных компьютерных сетей, стека протоколов TCP/IP, принципов расчета характеристик отдельных участков сетей передачи данных, методы защиты от ошибок при передачи данных.

Задачей изучения дисциплины является ознакомление обучающихся с вопросами функционирования и настройки аппаратного и программного обеспечения сетевых технологий передачи данных.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: Лк – 24 часов, ЛР – 24 часов, ПЗ – 12 часов, СРС – 48 часов.
Общая трудоемкость дисциплины составляет 144 часов, 4 зачетных единиц

2.2 Основные разделы дисциплины:

1. Информационная безопасность.
2. Защита информации при помощи криптографии.
3. Классификация шифров.
4. Криптосистемы.
5. Современные криптографические системы.
6. Теория сложности вычислений.
7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма.
8. Электронная подпись.
9. Хеш-функции.
10. Аутентификация и обмен ключами.

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующей компетенции:

ОПК-1 - Способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной.

ПК-17 - Способность применять современные теоретические и экспериментальные методы исследования с целью создания новых перспективных средств электросвязи и информатики

4. Вид промежуточной аттестации: экзамен

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:
_____
_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____
_____
_____

Протокол заседания кафедры №____ от «___» _____ 20 __ __ г.,
(разработчик)

Заведующий кафедрой ___ _____ _____
(подпись) *(Ф.И.О.)*

Приложение 4

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-1	Способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной.	**1.**. Информационная безопасность	1.1. Введение в информационную безопасность	Контрольная работа
		2. Защита информации при помощи криптографии	2.1. Защита передаваемых и хранимых секретных данных от разглашения и искажения	Отчеты по лабораторным работам, контрольная работа
		3. Классификация шифров	3.1. Перестановочные шифры	Отчеты по лабораторным работам, контрольная работа
		4. Криптосистемы	4.1. Устройство шифров	Контрольная работа
			4.3. Алгоритм DES	Контрольная работа
		5. Современные криптографические системы	5.1. Предварительные математические понятия	Контрольная работа
			5.3. Спецификация алгоритма	Контрольная работа
		6. Теория сложности вычислений	6.1. Сложность вычислений.	Контрольная работа
		7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма	7.1. Алгоритм RSA	Контрольная работа
		8. Электронная подпись.	8.1. Общая схема цифровой подписи	Контрольная работа
			8.3. Электронная подпись на основе алгоритма RSA	Контрольная работа
		9. Хеш-функции	9.1. Требования к хэш-функциям	Контрольная работа
		10. Аутентификация и обмен ключами.	10.1. Алгоритмы распределения ключей с использованием третьей доверенной стороны Понятие мастер-ключа	Контрольная работа

			10.3. Использование шифрования с открытым ключом	Контрольная работа
ПК-17	Способность применять современные теоретические и экспериментальные методы исследования с целью создания новых перспективных средств электросвязи и информатики	**1.** Информационная безопасность	1.2. Модель сетевой безопасности .Классификация сетевых атак	Контрольная работа
		2. Защита информации при помощи криптографии	2.2. Задача подтверждения авторства сообщения	Отчеты по лабораторным работам, контрольная работа
		3. Классификация шифров	3.2. Классификация методов дешифрования. Модель предполагаемого противника. Правила Керкхоффа	Отчеты по лабораторным работам, контрольная работа
		4. Криптосистемы	4.2. Поточные шифры	Отчеты по лабораторным работам, контрольная работа
			4.4. Алгоритм ГОСТ 28147	Контрольная работа
		5. Современные криптографические системы	5.2. Обоснование разработки	Контрольная работа
			5.4. Преимущества алгоритма	Контрольная работа
		6. Теория сложности вычислений	6.2. Сложность алгоритмов.	Контрольная работа
		7. Алгоритм RSA. Математическая модель алгоритма. Стойкость алгоритма	7.2. Криптосистема Эль-Гамаля.	Контрольная работа
		8. Электронная подпись.	8.2. Стандарт цифровой подписи DSS	Контрольная работа
		9. Хеш-функции	9.2. Простые хэш-функции	Контрольная работа
		10. Аутентификация и обмен ключами.	10.2. Протоколы аутентификации	Контрольная работа

2. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать (ОПК-1): – основные закономерности передачи информации в инфокоммуникационных системах, основные виды сигналов, используемых в телекоммуникационных системах, особенности передачи различных сигналов по каналам и тракам телекоммуникационных системсистем; (ПК-17): - основы цифровой вычислительной техники, структуры и функционирование локальных вычислительных сетей и глобальной сети Интернет; **Уметь** (ОПК-1): – формулировать основные технические требования к телекоммуникационным сетям и систем; (ПК-17): – оценивать основные проблемы, связанные с эксплуатацией и внедрением новой телекоммуникационной техники **Владеть** (ОПК-1): - начальными навыками разработки программного обеспечения сигнальных процессов и микроконтроллеров. (ПК-17): начальными навыками отладки с использованием соответствующих отладочных средств программного обеспечения сигнальных процессов и микроконтроллеров;микроконтроллеров.	**зачтено**	Во время защиты лабораторных работ и контрольной работы студент ответил на поставленные преподавателем вопросы.
	не зачтено	Во время защиты лабораторных работ и контрольной работы студент не смог дать ответы на поставленные преподавателем вопросы. Либо отчет имеет ряд замечаний.

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 11.03.02 Инфокоммуникационные технологии и системы связи от «6» марта 2015 г. №174

для набора 2015 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «13» июля 2015г. № 475

для набора 2016 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «06» июня 2016г. № 429

для набора 2017 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «6» марта 2017г. № 125

для набора 2018 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «12» марта 2018г. № 130

Программу составил (и):

Ульянов А.Д. старший преподаватель кафедры УТС _____
Ф.И.О., должность, ученое звание, (степень) *(подпись)*

Рабочая программа рассмотрена и утверждена на заседании кафедры УТС
(сокращенное наименование)
от «28» __декабря___ 2018 г., протокол № 6

Заведующий кафедрой УТС _____ Игнатьев И.В.
(разработчик) *(подпись)* *(Ф.И.О.)*

СОГЛАСОВАНО:

Заведующий выпускающей кафедрой _____ Игнатьев И.В.
(подпись) *(Ф.И.О.)*

Директор библиотеки _____ Сотник Т.Ф.
(подпись)

Рабочая программа одобрена методической комиссией ЭиА факультета
(сокращенное наименование)
от «28» __декабря_____ 2018 г., протокол № 6

Председатель методической комиссии факультета _____ Ульянов А.Д.
(подпись) *(Ф.И.О.)*

СОГЛАСОВАНО:

Начальник
учебно-методического управления _____ Г.П. Нежевец

Регистрационный №_____