Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет инфокоммуникаций

Кафедра инфокоммуникационных технологий

М. Н. Бобов, О. Г. Шевчук

ЗАЩИТА ИНФОРМАЦИИ В СЕТЯХ ИНФОКОММУНИКАЦИЙ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-45 80 01 «Системы и сети инфокоммуникаций»

Минск БГУИР 2021

УДК 004.056(076)
ББК 32.973.202-018.2я7
Б72

Р е ц е н з е н т ы:

кафедра связи учреждения образования «Военная академия Республики Беларусь» (протокол №9 от 16.12.2019);

доцент кафедры автоматики, телемеханики и связи учреждения образования «Белорусский государственный университет транспорта» кандидат технических наук, доцент П. М. Буй

Бобов, М. Н.

Б72 Защита информации в сетях инфокоммуникаций : учеб.-метод. пособие / М. Н. Бобов, О. Г. Шевчук. – Минск : БГУИР, 2021. – 68 с. : ил.
ISBN 978-985-543-572-4.

Рассмотрены методы защиты информации в сетях инфокоммуникаций, технологии построения защищенных локальных и распределенных сетей, архитектура безопасности на уровнях 1–4 моделей взаимосвязи открытых систем. Приведено описание методов аутентификации и управления доступом в сетях, стандартизированных в Республике Беларусь алгоритмов шифрования, контроля целостности и электронной цифровой подписи. Изложены рекомендации по их использованию при создании сетей инфокоммуникаций.

Предназначено для студентов второй ступени образования, изучающих дисциплину «Методы защиты сетей инфокоммуникаций».

УДК 004.056(076)
ББК 32.973.202-018.2я7

ISBN 978-985-543-572-4

Содержание

ВВЕДЕНИЕ

Современные информационные технологии направлены на увеличение степени автоматизации всех информационных процессов, что является предпосылкой ускорения темпов научно-технического прогресса. Реализация информационных технологий невозможна без создания компьютерных сетей, в которых сочетается как различного рода аппаратура по обработке, хранению и передаче информации, так и различные виды обслуживания. Вычислительные машины и телекоммуникационные средства, объединенные в инфокоммуникационные сети, дают возможность мгновенного доступа к любым специализированным данным практически из любого места земного шара.

Безопасность телекоммуникационных сетей предполагает решения по защите инфраструктурных компонентов системы. Цель решений – защитить инфраструктуру от неавторизованного доступа и обеспечить высокую отказоустойчивость и доступность сервисов. Для защиты информационных ресурсов инфраструктуры используются следующие типовые решения:

1) Изоляция трафика различных систем путем создания виртуальных оверлейных сетей. Трафик каждой такой сети логически изолирован и не может попасть в другую оверлейную сеть. Для усиления сегментации функциональных систем между ними создаются демилитаризованные зоны в виде дополнительных оверлейных сетей.

2) Контроль доступа пользователей в сеть и к ресурсам сети на основе использования механизмов управления доступом.

3) Закрытие криптографическими средствами как каналов обмена информацией при взаимодействии между сетями и внутри сетей, так и информации, хранимой в базах данных.

4) Обеспечение контроля целостности обрабатываемой информации, а также используемого программного обеспечения от целенаправленного несанкционированного изменения.

5) Развертывание и ввод в действие системы мониторинга безопасности компонентов инфраструктуры (SIEM).

6) Организация системы контроля доступа протоколирования всех событий, связанных с безопасностью инфокоммуникационной структуры.

Документом ИСО 7498-2-99 «Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 2. Архитектура защиты информации» определены базовые сетевые механизмы защиты, в том числе аутентификация, управление доступом, шифрование, электронная цифровая подпись, контроль целостности, аудит. Кроме базовых для защиты инфокоммуникаций используются специфические механизмы, к которым относятся: межсетевое экранирование, системы обнаружения вторжений, сканеры безопасности, средства защиты коммутаторов и маршрутизаторов. В данном учебно-методическом пособии будут рассмотрены указанные выше базовые механизмы, используемые для защиты сетей инфокоммуникаций.

1 ОПИСАНИЕ СТРУКТУРЫ ЗАЩИЩЕННОЙ ЛОКАЛЬНО-ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

Структура защищенной локально-вычислительной сети (ЛВС) приведена на рисунке 1.1.

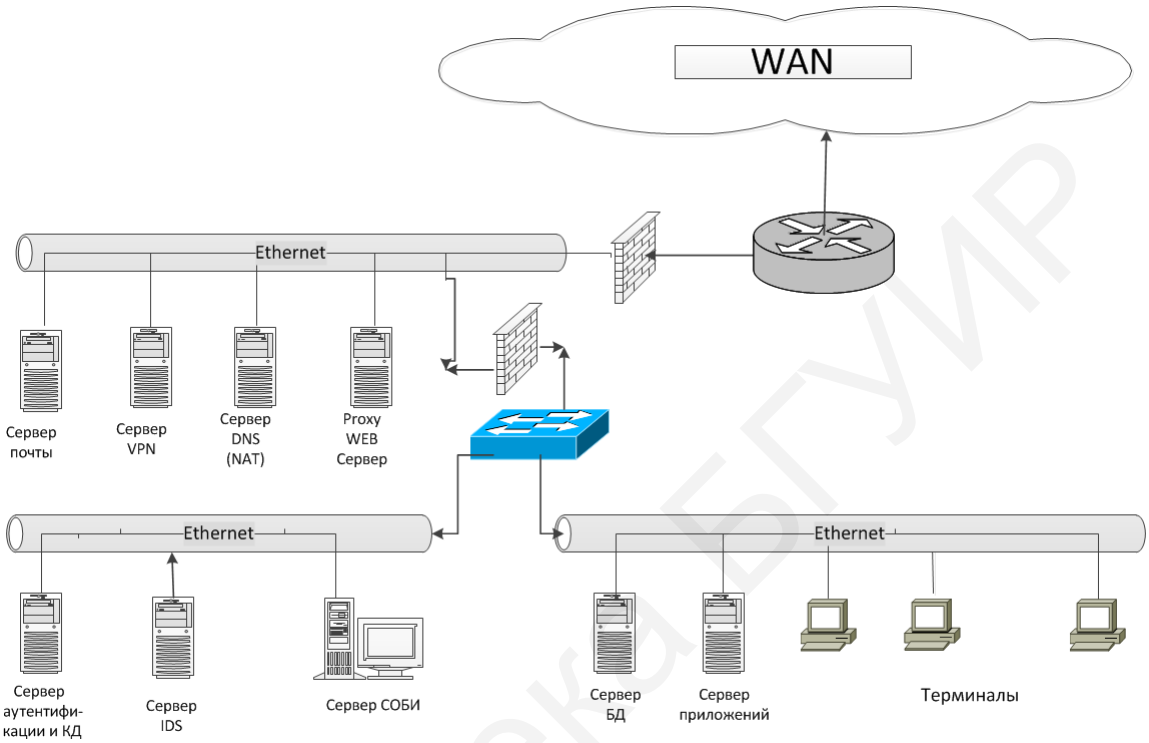


Рисунок 1.1 – Структура защищенной ЛВС

ЛВС разделяется на три зоны безопасности: 1) подключения к глобальной сети; 2) управления безопасностью и ресурсами сети; 3) защищаемых данных, обрабатываемых в ЛВС.

Для разделения зоны подключения к WAN (Wide Area Network) и внутренних сетей используются внутренний МСЭ (межсетевой экран) и коммутатор. Внутренний МСЭ служит для контроля информационных потоков между внутренними сетями и обеспечивает изоляцию зоны управления от остальной сети, управление доступом для соединений VPN (Virtual Private Network), защиту внутренней сети путем запрета трафика из менее защищенной зоны внешних подключений. Коммутатор служит для разделения зон управления и защищенных данных, ограничивая нежелательный доступ пользователей к ресурсам управления.

Все компьютеры защищенной ЛВС снабжаются средствами доверенной загрузки, обеспечения контроля состава и конфигурации ПО, физической целостности аппаратуры.

1.1 Зона подключения к глобальной сети

Зона подключения к глобальной сети включает:

1) Пограничный маршрутизатор. Представляет собой первую линию защиты и обеспечивает фильтрацию нежелательного трафика путем настройки списков управления доступа (ACL), защиту внешнего МСЭ от трафика, направленного на IP-адреса МСЭ.

2) Внешний МСЭ. Обеспечивает защиту ЛВС от атак извне и разрешает ограниченный набор трафика в соответствии с принятой политикой безопасности.

3) Почтовый сервер. Обеспечивает обмен сообщениями в защищенном режиме, в том числе защиту от спама, замену заголовка для удаления информации о внутренней маршрутизации, изменение SMTP-приглашения для затруднения получения информации о версии ПО, исключение команд для просмотра списка почтовых ящиков.

4) Сервер VPN. Обеспечивает защищенный обмен с удаленными абонентами ЛВС и аутентификацию сторон с использованием сертификатов, шифрование информации в соответствии с протоколом IPSec, аудит и протоколирование сеансов VPN.

5) Сервер DNS. Используется для разрешения имен внутренней сети и обеспечивает преобразование адресов для скрытия конфигурации ЛВС.

6) Proxy WEB сервер. Служит для обеспечения исходящего доступа в Интернет и реализует политику использования Интернета, ограничивая доступ к определенным веб-ресурсам.

1.2 Зона управления безопасностью и ресурсами сети

Зона управления безопасностью и ресурсами сети включает:

1) Сервер аутентификации и контроля доступа. Обеспечивает процесс аутентификации пользователей на основе сертификатов. Аутентификация используется для доступа посредством VPN, к веб-сайтам по протоколу TLS (SSL), для доступа к базе данных.

2) Сервер IDS. Используется для обнаружения и предотвращения вторжений и обеспечивает управление конфигурированием МСЭ, мониторинг производительности и инвентаризации сети.

3) Сервер системы обеспечения безопасности информации (СОБИ) предназначен для управления безопасностью пользователей и обеспечивает: создание учетных записей пользователей, генерацию секретных параметров безопасности, мониторинг безопасности сети и т. д.

1.3 Зона защищаемых данных, обрабатываемых в ЛВС

Зона защищаемых данных включает в себя сервер базы данных (БД), сервер приложений и терминалы.

На серверах БД и приложений реализуется система управления доступом пользователей к информации в БД.

Защита терминалов включает в себя аутентификацию пользователей и другие меры, используемые для защиты ПЭВМ.

2 АУТЕНТИФИКАЦИЯ

Краеугольным вопросом при организации взаимодействия пользователей между собой, пользователей с серверами приложений и баз данных, между серверами приложений и баз данных в процессе функционирования инфокоммуникационных систем является установление подлинности субъектов, пытающихся получить доступ к ресурсам системы. Данная проблема решается путем использования средств и методов аутентификации (АИ), которые по месту приложения можно разделить на два вида:

1) **Локальная аутентификация**, когда вся система, включая механизм аутентификации и управления доступом, размещается внутри одного физического периметра безопасности. Владелец системы и/или пользователь ведут и обновляют базу аутентификационных данных внутри этого периметра. Аутентификация предполагает непосредственное взаимодействие субъекта с устройством проверки подлинности при входе в систему.

2) **Удаленная аутентификация,** используемая в современных сетевых серверных системах, содержит несколько точек обслуживания, которые требуют управления доступом и могут размещаться в различных местах. При необходимости пользователи обращаются к службам системы удаленным образом.

Независимо от вида процесс определения подлинности включает два элемента:

1) **Идентификация –** процесс распознавания элемента системы, обычно с помощью заранее определенного идентификатора или другой априорной информации. При идентификации происходит выбор элемента из множества.

2) **Аутентификация –** проверка истинности пользователя, процесса, устройства или другого компонента системы. При аутентификации происходит проверка подлинности заявленного идентификатора.

2.1 Локальная аутентификация

Реализация процедур опознания пользователей включает в себя идентификацию и проверку подлинности и является первой задачей для любых систем, в которых требуется обеспечивать разграничение доступа к обрабатываемой информации. Существуют три класса локальной аутентификации, которые базируются:

1) на условных, заранее присваиваемых признаках (сведениях), известных субъекту (что знает субъект);

2) на физических средствах, действующих аналогично физическому ключу (что имеет субъект);

3) на индивидуальных характеристиках субъекта, его физических данных, позволяющих выделить его среди других лиц (что присуще субъекту).

2.1.1 Опознание на основе принципа «что знает субъект»

Метод паролей. Данный метод заключается в том, что пользователь на

клавиатуре компьютера набирает только ему известную комбинацию букв и цифр, являющуюся, собственно, паролем. Данный пароль сравнивается с эталонным, хранящимся в системе, и при положительном результате проверки пользователь получает доступ к системе. Такая схема опознания является простой с точки зрения реализации, так как не требует никакой специальной аппаратуры и реализуется посредством небольшого объема программного обеспечения.

Модернизацией схемы простого пароля является пароль однократного использования. В этой схеме пользователю выдается список из N паролей. Такие же N паролей хранятся в системе. Данная схема обеспечивает большую степень безопасности, но она является и более сложной. Здесь при каждом обращении к системе синхронно используется пароль с текущим номером, а все пароли с предыдущими номерами вычеркиваются. В случае если старый пароль из предыдущего сеанса стал известен другому пользователю, система его не воспринимает, так как действующим будет следующий по списку пароль.

При использовании генератора паролей в ЭВМ реализуется алгоритм, осуществляющий преобразование

$$F(x,k)=y, \qquad (2.1)$$

где x, k, y – двоичные векторы характеристического номера, ключа и пароля, соответственно.

Реализация процедуры опознания пользователя сводится к двум задачам: заготовке паролей и установлению подлинности.

При заготовке паролей с помощью преобразования (2.1) получают набор чисел (X_i^j, Π_i^j), где i – номер пользователя; j – номер обращения данного пользователя; П – текущее значение пароля, сформированное на ключе k. Полученный набор выдается соответствующим пользователям.

При опознании обращающийся к системе пользователь с номером i вводит парольный набор (X_i^j, Π_i^j) в ЭВМ. Программа опознания выделяет номер пользователя X_i^j, а также запоминает пароль Π_i^j. Для каждого i-го пользователя существует свой счетчик обращений S_i. В случае если $j \le S_i$, программа выдает сообщение о несанкционированном доступе. В противном случае включается генератор паролей. Преобразование $F(x_i^j, k)$ на действующем ключе k выдает число y, которое сравнивается с паролем Π_i^j. В случае совпадения y и Π_i^j пользователь считается опознанным, а в случае несовпадения – выдается сигнал о несанкционированном доступе.

Метод «запрос – ответ». В методе «запрос – ответ» набор ответов на m стандартных и n ориентированных на пользователя вопросов хранится в ЭВМ и управляет программой опознания. Когда пользователь делает попытку включиться в работу, программа опознания случайным образом выбирает и задает ему некоторые (или все) из этих вопросов. Пользователь должен дать правильный ответ на все вопросы, чтобы получить разрешение на доступ к системе.

Вопросы могут быть выбраны таким образом, чтобы пользователь запомнил ответы и не записывал их.

Существует два варианта использования метода «запрос – ответ», вытекающие из условий $m = 0$ или $n = 0$. Вариант с $m = 0$ предполагает, что вопросы составлены на основе различных фактов биографии индивидуального пользователя, представляют собой имена его друзей, дальних родственников, старые адреса и т. д. Иногда предпочтительнее вариант с $n = 0$, т. е. пользователям задается большее количество стандартных вопросов и от них требуются ответы на те, которые они сами выберут.

Стойкость паролей. Стойкость парольной системы зависит в основном от двух параметров: длины пароля и объема алфавита. Основные формулы оценки парольных систем:

а) вероятность подбора пароля с первой попытки:

$$P_{П_1} = 1 / A^S,$$

где A – объем алфавита;

S – длина пароля.

б) вероятность подбора пароля с i-й попытки: $P_{П_i} = 1/(A^S + 1 - i)$;

в) вероятность подбора пароля за k попыток: $P_{П_k} = k / A^S$;

г) вероятность подбора пароля в период его безопасного времени действия:

$$P_{T_Б} = \frac{3600 \cdot T_Б}{A^S \cdot t_П},$$

где $T_Б$ – безопасное время действия;

$t_П$ – время набора пароля.

2.1.2 Опознание на основе принципа «что имеет субъект»

К этому классу опознания относятся методы, основывающиеся на физических средствах, которые имеет при себе данный пользователь, обращающийся к системе. К ним относятся магнитные карточки, смарт-карты, USB-ключи, таблетки Touch Memory и прочие подобные средства, которые можно объединить общим названием – электронный ключ.

Идентификационные магнитные карты. В магнитных картах информация записывается на нескольких дорожках магнитного слоя и представляет собой данные, используемые для идентификации, а именно: номер пользователя или его имя, пароль, количество допустимых использований карты и т. д. Наряду с очевидной простотой использования магнитные карты обладают низкой защищенностью от копирования содержимого. Для защиты от копирования магнитные карты снабжаются различными защитными средствами. Один из методов состоит в нанесении магнитного слоя обычного типа поверх второго слоя с более высокой коэрцитивной силой, т. е. для изменения состояния первого слоя требуется более сильное магнитное поле. В этом случае обычными методами невозможно считать или изменить запись нижнего слоя. В другом методе используется постоянная

магнитная разметка ленты, которая наносится в процессе ее производства. Метод, известный под названием «влажной разметки» состоит в определенной ориентации осей ферромагнитных кристаллов до момента, пока наполнитель еще не высох, причем селективная ориентация осей кристаллов в различных частях ленты создает магнитную запись, которую никак нельзя изменить. Чтобы прочесть эту запись, необходимо подвергнуть кристаллы воздействию постоянного магнитного поля с определенной ориентацией. Изменение положения кристаллов вдоль ленты будет наводить внешнее поле, которое можно прочитать с помощью обычных, удобно расположенных головок. Изготовленные таким образом идентификационные карточки могут обеспечить «уникальную» идентичность, которую трудно подделать, поскольку для этого требуется овладеть технологией производства магнитных покрытий и влажной разметки.

Электронные ключи. Электронные ключи в самом общем смысле представляют собой физические носители идентификатора субъекта и его пароля. Кроме того, в них содержится дополнительная информация, необходимая в процессе опознания субъекта. Структурная схема электронного ключа представлена на рисунке 2.1.

Рисунок 2.1 – Структурная схема электронного ключа

Для своего восприятия электронный ключ должен взаимодействовать с «замком» (ответной частью), запрашивающим ключ и проверяющим его идентичность, например, смарт-карта должна иметь ридер. В процессе обмена информацией с ридером осуществляется опознание смарт-карты. Опознание субъ-

екта происходит после подтверждения им того, что именно он является владельцем смарт-карты в результате ввода с клавиатуры PIN-кода.

Независимо от сферы использования электронный ключ содержит в своем составе узел памяти для хранения секретного кода, узел преобразования для обеспечения безопасной передачи кода ключа за его пределы, а также узел сопряжения для соединения и обеспечения интерфейса с ответной частью. Ответная часть, являясь, как указывалось выше, обязательным элементом электронного ключа, содержит, помимо перечисленных на рисунке 2.1 узлов, узел питания и синхронизирующий блок для задания программы работы электронного ключа.

Секретность электронного ключа. В силу того что электронный ключ является физическим средством хранения аутентификатора пользователя, существует вероятность его копирования и подделки. Однако устройство можно сконструировать таким образом, чтобы попытка несанкционированного чтения ключа приводила к его уничтожению.

Секретность электронного ключа, как и всех устройств опознания, относящихся ко второму классу, определяется выражением

$$P_{\text{СК}} = 1-(1-P_{\text{Н}})(1-P_{\text{П}})(1-P_{Y}),$$

где $P_{\text{Н}}$ – вероятность несанкционированного копирования секретных параметров ключа;

$P_{\text{П}}$ – вероятность подбора входных воздействий;

P_{Y} – вероятность подбора выходных воздействий (т. е. обхода).

Обратной величиной секретности является вероятность подделки, которая определяется по формуле

$$P_{K} = 1-P_{\text{СК}}.$$

Для уменьшения вероятности подделки при разработке устройств опознания используются следующие механизмы:

1) ограничение попыток НСД;

2) контроль цикла работы;

3) маскирование выходных сообщений.

При использовании первого механизма ограничивается количество возможных попыток ввода неправильных кодов аутентификации. В этом случае, при допустимом числе попыток K, вероятность подбора входных воздействий $P_{\text{П}}$ равна

$$P_{\text{П}} = 1-\prod_{i=1}^{K}\left(1-P_{\text{П}_i}\right) = K/N,$$

где N – объем входного алфавита.

Так как $K << N$ и $K = \text{const}$, то при использовании этого механизма получаем выигрыш в защите.

Реализация механизма контроля цикла работы заключается в том, что устройство опознания функционирует по заранее установленным жестким циклам и ни при каких входных воздействиях такая цикличность его работы не нарушается. В результате все выходные сообщения из устройства выдаются только в заранее определенные моменты времени, когда процессы приема и обработки входного

воздействия полностью завершены. Так как цикл работы жестко задан, всегда можно ввести определенную задержку по времени его выполнения таким образом, чтобы повысить безопасное время секретного кода, которое определяется по формуле

$$T_{\text{Б}} = W \cdot T_{\text{Ц}} = W \cdot \left(t_{\text{р}} + \tau\right),$$

где W – количество операций, которые необходимо выполнить для подбора секретного кода;

$T_{\text{Ц}}$ – время выполнения цикла;

$t_{\text{р}}$ – время непосредственной работы;

τ – задержка по циклу.

При маскировании выходных сообщений алфавит выходных сообщений должен быть многоальтернативным, т. е. таким, чтобы в процессе осуществления попыток компрометации ключа невозможно было установить соответствие между входными и выходными сообщениями. Реализация механизма обеспечивается использованием генератора случайных чисел (ГСЧ) и криптосхемы.

2.1.3 Опознание на основе принципа «что присуще субъекту»

К наиболее широко используемым персональным характеристикам относятся голос, личная подпись, форма ладони и отпечатки пальцев. В последнее время появилось еще несколько методов физического опознания – по структуре сетчатки глаз, сопротивлению определенных участков кожи, запаху тела и др. В каждом случае способ опознания состоит в измерении индивидуальных характеристик и вычислении индексов, аналогичных характеристическим параметрам распознавания образов, которые можно передать в центральную ЭВМ для сопоставления с набором индексов, хранящихся в памяти ЭВМ и взятых непосредственно у интересующего лица.

Параметры идентификации физических признаков. Механизм опознания личной подписи может измерять число касаний и отрывов пера от бумаги, среднюю вертикальную скорость движения пера, число вертикальных отклонений и множество других подобных параметров. Эти характеристики могут быть самыми разнообразными, однако не все из них являются независимыми, и задача состоит в том, чтобы выбрать хороший набор характеристик с достаточно малой взаимной корреляцией. Проверка подлинности подписи зависит от движения пера, которое нельзя воспроизвести по виду подписи, зафиксированной на бумаге. Это почти полностью исключает возможность подлога, так как мастерство тех, кто профессионально подделывает подписи, основано на внешнем виде почерка. Набор измеряемых характеристик должен сохраняться в тайне, так как их знание может привести к подделке подписи посредством тренировки в копировании измеряемых характеристик. Как показала практика, это очень сложная задача.

Аналогичные особенности характерны и для других методов опознания этого класса. Например, некоторые устройства, определяющие форму ладони, измеряют прозрачность тканей кожи между пальцами для защиты от подлогов с помощью картонных шаблонов.

Механизмы, построенные на анализе отпечатков пальцев, используют мельчайшие детали в виде разветвлений, окончаний и пробелов в линиях на кончиках пальцев. Так как в каждом отпечатке содержится множество таких отличий, измеряемые характеристики могут базироваться на выбранном наборе деталей. Существуют два основополагающих алгоритма распознавания отпечатков пальцев: по отдельным деталям (характерным точкам) и по рельефу всей поверхности пальца.

В первом случае устройство регистрирует только некоторые участки, уникальные для конкретного отпечатка, и определяет их взаимное расположение. Во втором случае обрабатывается изображение всего отпечатка.

Метод опознания субъекта по лицу основан на уникальности черт лица. Метод заключается в преобразовании черт конкретного лица в алгоритмическую модель, которая сравнивается или с фотографией на пропуске, или с содержимым базы фотографических данных.

Метод опознания субъекта по радужной оболочке глаза основан на уникальности рисунка радужной оболочки каждого субъекта. Радужная оболочка субъекта сканируется, разворачивается и преобразуется в цифровую последовательность. Подтверждение подлинности субъекта происходит на основании сравнения полученной цифровой последовательности с эталонной.

Метод опознания по образцу голоса основан на том, что у каждого субъекта неповторимый голосовой рисунок, который зависит от пола, физических особенностей, типа строения голосовых связок, полости носа, формы рта, таких характеристик, как частота и амплитуда. Этот метод построен на выделении различных сочетаний частотных и статистических характеристик голоса.

Особенности опознания по физическим признакам. Значения характеристик, получаемых в процессе работы средств опознания по физическим признакам, всегда имеют разброс в небольшой области с некоторым вероятностным распределением, поэтому для принятия решения необходимо определить «окно приемлемости» для каждого параметра. При экспериментальной оптимизации качества механизма опознания размер «окна» может меняться, но он всегда остается больше некоторого минимума, так как практически не существует абсолютно надежного набора параметров для опознания по физическим признакам. Если взять слишком широкое значение «окон», то можно принять любой запрос, а если очень узкое, то на все попытки последует отказ, в том числе и на запросы законных пользователей. На рисунке 2.2 представлены типичные кривые, показывающие соотношение между ошибками этих двух типов.

На практике, как видно из рисунка 2.2, оба типа ошибок нельзя свести к нулю одновременно независимо от величины установленного порога. В этом коренное отличие последнего класса систем опознания от первых двух, где опознание считается установленным только после абсолютного совпадения предъявленного аутентификатора.

Процент ошибочных отказов можно уменьшить, если пользователю предоставить возможность выполнить процедуру идентификации более одного раза, причем успех любой из попыток принимается как достаточное условие

для подтверждения личности. Использование метода опознания по физическим признакам при опознании лица, не известного заранее, является практически неосуществимым, так как это требует выработки критериев оценки персональных характеристик, чтобы выделить одного индивидуума среди всех других, обслуживаемых данной системой.

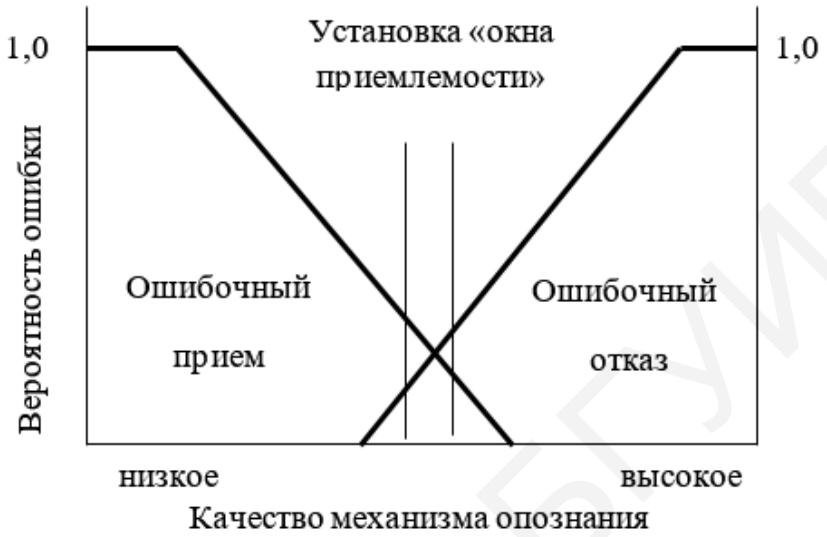


Рисунок 2.2 – Два вида ошибок при опознании по индивидуальным характеристикам

2.2 Удаленная аутентификация

Большинство современных информационных сетей, состоящих из компьютеров и других устройств, являются открытыми. Это значит, что пользователь, в роли которого может выступать компьютер, устройство, ресурс, провайдер, человек или их совокупность, может присоединяться к сети, чтобы передавать и получать сообщения от других пользователей, входящих в сеть, без разрешения «главного» администратора доступа. При функционировании открытых систем пользователи взаимодействуют в интерактивном режиме, причем, если два незнакомых пользователя захотят установить между собой секретную связь, они должны аутентифицировать друг друга и организовать защищенный канал связи. Данные задачи решаются с помощью протоколов аутентификации на основе применения криптографических методов.

2.2.1 Задачи и принципы построения протоколов аутентификации

Первой задачей протоколов аутентификации является установление подлинности участников взаимодействия, разделенных коммуникационной системой. Как правило, она выясняется путем определения подлинности соответствующих сообщений. Поэтому, чтобы убедиться в подлинности сообщения и его автора, используются механизмы аутентификации источника данных, предусматривающие выполнение следующих действий:

1) передача сообщения от отправителя к получателю, проверяющему достоверность сообщения до его обработки;

2) идентификация отправителя сообщения;

3) проверка целостности данных, полученных от отправителя;

4) проверка подлинности отправителя сообщения.

Следующая задача протоколов аутентификации – обеспечить согласование ключей шифрования, используемых для защиты передаваемой информации. Поэтому протоколы аутентификации для дальнейшего обмена информацией по защищенным каналам в качестве составной части содержат или механизм формирования аутентифицированных ключей, или механизм обмена ключами.

К основным протокольным конструкциям, принятым в качестве международных стандартов, относятся следующие:

– механизмы определения «актуальности» сообщения и существования пользователя;

– односторонняя и взаимная аутентификация;

– аутентификация с привлечением доверенного посредника.

Проверка «актуальности» сообщения – неотъемлемая часть аутентификации источника данных, в процессе которой пользователь должен активно обмениваться информацией с подлинным партнером. Для ее реализации используются механизмы «запрос – ответ» и «метка времени».

В механизме «запрос – ответ» проверяющая сторона получает комбинацию, состоящую из протокольного сообщения и криптографической операции, выполненной доказывающей стороной таким образом, чтобы проверяющая сторона могла убедиться в ее существовании, проверив «актуальность» полученной информации. Как правило, проверяющая сторона получает криптографическое преобразование заранее сгенерированного ей же одноразового случайного числа. Если обратное преобразование правильно восстанавливает случайное число, то проверяющая сторона имеет основания считать, что доказывающая сторона действительно зашифровала его после получения запроса, и сообщение считается «актуальным».

Используя механизм метки времени, доказывающая сторона указывает время создания своего сообщения с применением криптографической операцию. Следовательно, время создания сообщения становится его неотъемлемой частью. После расшифрования сообщения проверяющая сторона может сравнить извлеченную метку времени со своим собственным временем (предполагается, что участники протокола используют общемировое стандартное время, например, по Гринвичу). Если разница во времени относительно мала, сообщение считается «актуальным».

Указанные механизмы проверки «актуальности» сообщения и существования пользователя обеспечивают одностороннюю проверку подлинности, в которой аутентифицировался только один из двух участников протокола. Для правильной взаимной аутентификации недостаточно дважды выполнить протокол односторонней проверки подлинности в противоположных направлениях. Для этого случая используется специальная конструкция протокола, основанная на комбинации стандартных механизмов.

Как правило, для аутентификации и формирования ключей в открытых системах используется централизованная служба аутентификации, известная под названием «доверенный посредник». Считается, что между доверенным

посредником и большим количеством пользователей в системе существуют долговременные отношения. Данная служба поддерживает БД, содержащую имена обслуживаемых клиентов, и может идентифицировать пользователя на основе криптографического ключа, заранее разделенного между ним и пользователем. Доверенный посредник является особым пользователем, которому доверяют все остальные пользователи (клиенты), т. е. он всегда отвечает на запрос клиента точно в соответствии со спецификацией протокола и не выполняет никаких других действий, которые могли бы непреднамеренно снизить степень безопасности (например, открыть третьей стороне секретные ключи, разделенные между ним и его клиентами).

2.2.2 Атаки на протоколы аутентификации

В открытой среде всегда имеют место атаки на протокол аутентификации, которые организуются противником или коалицией противников, преследующих незаконные цели. Как правило, протокол аутентификации считается некорректным, если пользователь считает, что протокол выполняется правильно и связь установлена с подлинным партнером, в то время как подлинный партнер приходит к противоположному выводу.

К наиболее распространенным и хорошо известным атакам на протоколы аутентификации относятся: «Повторная передача сообщения», «Человек посередине», «Параллельный сеанс», «Отражение сообщений», «Неправильная интерпретация», «Чередование сообщений», «Обезличивание сообщений», «Неправильное выполнение криптографических операций».

Атаки на протоколы, как правило, направлены на поиск недостатков, позволяющих противнику пройти аутентификацию без взлома криптографического алгоритма.

По этой причине при анализе протоколов аутентификации обычно предполагают, что лежащие в их основе криптографические алгоритмы являются стойкими, и не рассматривают их потенциальные слабости (анализ слабых мест криптографических алгоритмов является предметом криптоанализа).

«Модель противника». При создании протоколов аутентификации широко используется модель противника, в рамках которой он обладает характеристиками, представленными на рисунке 2.3.

При анализе стойкости протоколов принимают соглашения о поведении пользователя, участвующего в протоколе, в случае если пользователь:

– не понимает семантического смысла ни одного протокольного сообщения, пока протокол не завершится успешно;

– не способен ни распознать, ни создать, ни разложить сообщение на составные части, не имея правильного ключа;

– не способен распознавать псевдослучайные числа, порядковые номера и криптографические ключи, если они не сгенерированы самим пользователем в рамках текущего сеанса или не являются результатом выполнения протокола;

– не записывает протокольные сообщения, если этого не требует спецификация протокола, т. е. пользователь не должен запоминать никакую инфор-

мацию о состоянии, возникшем после успешного выполнения протокола, за исключением информации, являющейся результатом выполнения протокола и предназначенной для самого пользователя.

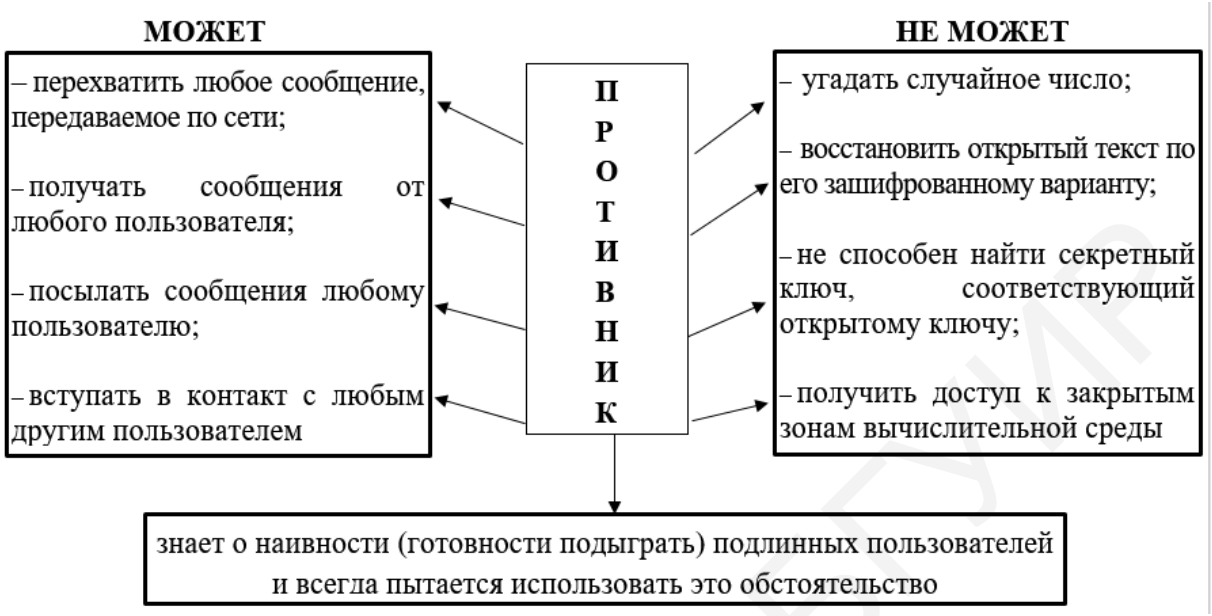


Рисунок 2.3 – Модель противника

2.3 Протоколы защищенных сокетов

Для обеспечения безопасности в сети World Wide Web используется протокол защищенных сокетов SSL, который выполняется в рамках протоколов на уровне приложений, например, в рамках протокола передачи гипертекстовых файлов HTTP, облегченного протокола службы каталогов LDAP или протокола доступа к сообщениям в сети Интернет IMAP, а также поверх стека протоколов TCP/IP. Протокол SSL в настоящее время является одним из самых популярных протоколов безопасности транспортного уровня, используемых в Интернете. Работая на транспортном уровне стека TPC/IP, он решает следующие задачи:

– обеспечивает конфиденциальность данных, т. е. уверенность, что они не были раскрыты в ходе транспортировки между клиентом и сервером путем шифрования данных всех вышестоящих уровней (представления и прикладного), для этого оставляет открытой только служебную информацию уровня TCP и ниже;

– обеспечивает аутентификацию сервера, т. е. уверенность пользователя (клиента), что он получает доступ именно к тому серверу, к которому необходимо;

– опционально может обеспечивать аутентификацию клиента, т. е. уверенность сервера, что он работает с авторизованным клиентом;

– обеспечивает целостность передаваемой информации, т. е. уверенность, что информация не была изменена в ходе транспортировки между клиентом и сервером;

– опционально может сжимать данные для обеспечения скорости передачи.

Протокол транспортного уровня TLS является преемником протокола SSL и стандарта безопасности в World Wide Web и разработан проблемной группой проектирования IETF. Протокол TLS основан на протоколе SSL и не очень сильно отличается от него, поэтому в дальнейшем при описании протокола защиты данных в Интернете будем рассматривать протокол TLS, который состоит из двух протоколов – протокола записи, определяющего формат передачи данных, и протокола взаимосвязи, определяющего механизм установки соединения.

2.3.1 Протокол записи TLS

Протокол TLS обеспечивает безопасную инкапсуляцию канала связи для применения в рамках протокола более высокого уровня приложений. Этот протокол запускается поверх TPC/IP и обеспечивает надежный сеанс связи. Он распределяет данные по блокам, сжимает их при необходимости, применяет код аутентификации сообщения для защиты целостности данных, шифрует сообщение с помощью симметричного алгоритма и передает результат адресату. Адресат получает зашифрованные блоки данных, расшифровывает их, верифицирует код MAC, разархивирует их при необходимости, собирает блоки в одно целое и доставляет результат на более высокий уровень приложений.

Ключи для симметричного шифрования и код MAC генерируются для каждого сеанса связи отдельно и основаны на секрете, согласованном в протоколе взаимосвязи TLS. Кроме того, данный протокол берет на себя функции генерирования сообщений об ошибках и закрытии сессии.

2.3.2 Протокол взаимосвязи

Протокол взаимосвязи обеспечивает настройку множества криптографических параметров. В момент, когда клиент пытается установить соединение с сервером, обе стороны осуществляют следующие действия (рисунок 2.4):

– обмениваются приветствиями, согласовывая алгоритм, пересылают друг другу случайные числа и проверяют возможность возобновления сеанса;

– обмениваются необходимыми криптографическими параметрами, позволяющими клиенту и серверу согласовать секрет (так называемый «главный секрет»);

– обмениваются сертификатами и криптографической информацией, позволяющей клиенту и серверу аутентифицировать друг друга;

– генерируют сеансовые секреты на основе главного секрета, обмениваясь случайными числами;

– убеждаются, что их партнер вычислил те же самые параметры безопасности, протокол выполнен успешно и не подвергся атаке;

– установленный защищенный канал передается протоколу записи TSL для обработки на более высоком уровне приложений.

Указанные на рисунке 2.4 этапы реализуются с помощью четырех обменов, включающих в себя:

1) обмен приветствиями;
2) предложения ключей сервером;
3) ответ клиента;
4) обмен заключительными сообщениями.

На этапе 4 клиент отправляет сообщение ClientFinished, означающее, что этап согласования закончен, при этом данное сообщение уже зашифровано в рамках новых параметров шифрования. Сервер также производит установку текущих параметров шифрования и отвечает сообщениями Change_cipher_spec и ServerFinished, позволяющими клиенту убедиться в согласовании параметров, выполненных сервером.

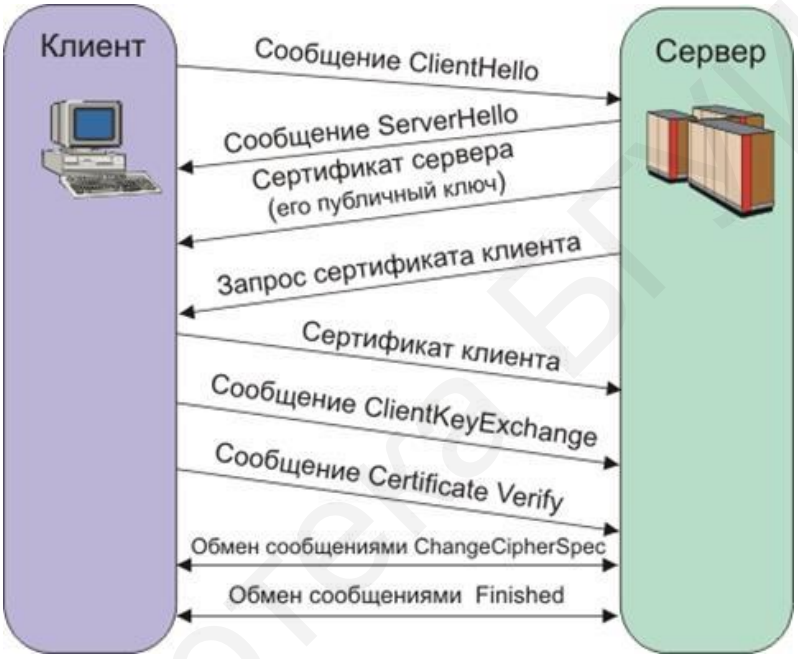


Рисунок 2.4 – Схема работы протокола взаимосвязи

Таким образом, взаимосвязь считается установленной, клиент и сервер могут переходить к обмену данными на уровне приложений.

2.4 Протокол обеспечения безопасности IPSec в сети Интернет

Протокол обеспечения безопасности в сети Интернет, известный под названием IPSec, предназначен для криптографической защиты IP-заголовка, состоящего из трех полей IP-пакета (рисунок 2.5).

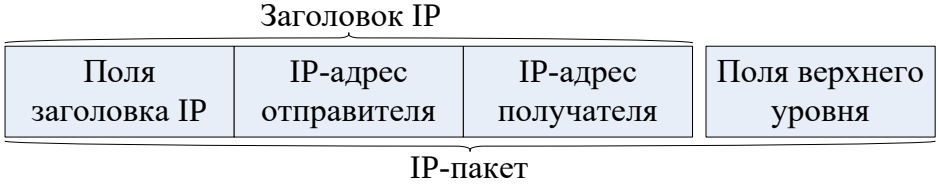


Рисунок 2.5 – Незащищенный IP-пакет

Назначение первых трех полей IP-пакета ясно по их названиям (рисунок 2.5). Четвертое поле, «Поля верхнего уровня», содержит, во-первых, спецификацию протокола, запускаемого на ближайшем верхнем уровне и обрабатывающего IP-пакет (т. е. TCP) и, во-вторых, данные, содержащиеся в IP-пакете.

IPSec обеспечивает сервис защиты на уровне IP, позволяя системе выбрать необходимые протоколы защиты, определить алгоритм для соответствующего сервиса и задать значения любых криптографических ключей, требующихся для запрашиваемого сервиса. Для защиты пакетов используются два протокола: протокол аутентификации, указанный заголовком аутентификации AH, и комбинированный протокол шифрования и аутентификации, определенный форматом пакета для протокола ESP. В таблице 2.1 показаны сервисы, обеспечиваемые применением указанных протоколов.

Таблица 2.1 – Сервисы, обеспечиваемые применением протоколов AH и ESP

Наименование сервиса	Протокол AH	Протокол ESP (только шифрование)	Протокол ESP (шифрование и аутентификация)
Управление доступом	×	×	×
Целостность без установки соединений	×	–	×
Аутентификация источника данных	×	–	×
Защита от воспроизведения пакетов	×	×	×
Конфиденциальность	–	×	×

Ключевым элементом реализации указанных в таблице 2.1 сервисов является защищенная связь SA, которая представляет собой одностороннее отношение между отправителем и получателем, применяющим сервис защиты к транспортному потоку. Если требуется равноправное отношение для двухстороннего защищенного обмена, необходимы две защищенные связи. Сервис защиты дает возможность для защищенной связи использовать либо AH, либо ESP, но никак не оба протокола одновременно.

Защищенная связь однозначно определяется следующими тремя параметрами:

1) **Индекс параметров защиты.** Строка битов, присваиваемая конкретной защищенной связи. Индекс параметров защиты передается в заголовках AH и ESP, чтобы принимающая система имела возможность выбрать защищенную связь, в рамках которой должен обрабатываться принимаемый пакет.

2) **Адрес IP пункта назначения.** Адрес пункта назначения защищенной связи, который может представлять систему конечного пользователя или сетевой объект типа VPN-агента.

3) **Идентификатор протокола защиты.** Этот идентификатор указывает, является ли данная защищенная связь защищенной связью AH или ESP.

Механизм управления ключами связывается с механизмами аутентификации и конфиденциальности только через параметры защиты. Таким образом,

он может быть определен независимо от механизмов аутентификации и конфиденциальности.

2.4.1 Протокол AH

Заголовок аутентификации обеспечивает поддержку целостности данных и аутентификацию пакетов IP. Функция аутентификации позволяет VPN-агенту идентифицировать пользователя или приложение, а также защититься от очень распространенных сегодня в Интернете атак с подменой сетевых адресов и несанкционированного воспроизведения сообщений. Аутентификация опирается на использование кодов аутентичности сообщений (MAC), при этом две стороны должны использовать общий секретный ключ.

Заголовок аутентификации состоит из следующих полей:

1) **Следующий заголовок (8 бит).** Идентифицирует тип заголовка, следующего непосредственно за данным заголовком.

2) **Длина значимых данных (8 бит).** Длина заголовка аутентификации в 32-битовых словах.

3) **Зарезервировано (16 бит).** Значение поля должно быть нулевым.

4) **Индекс параметров защиты (32 бита).** Идентифицирует защищенную связь.

5) **Последовательный номер (32 бита).** Монотонно возрастающий номер в диапазоне от 0 до $(2^{32} – 1)$, использующийся для нумерации пакетов.

6) **Данные аутентификации (переменной длины).** Поле переменной длины, содержащее код аутентификации сообщения для данного пакета.

Когда устанавливается новая защищенная связь, отправитель инициализирует счетчик последовательных номеров, установив соответствующее значение равным нулю. Каждый раз, когда по защищенной связи посылается пакет, отправитель увеличивает значение данного счетчика и размещает его в поле последовательных номеров. На приемной стороне осуществляется контроль последовательности номеров принимаемых пакетов, и пакеты с одинаковыми номерами отбрасываются. Когда значение счетчика превысит значение $(2^{32} – 1)$, отправитель должен завершить данную защищенную связь и инициализировать новую защищенную связь с новым ключом.

Для вычисления кода аутентификации сообщения выбираются поля заголовка IP, которые не изменяются в пути следования, заголовок AH и все данные протокола следующего выше уровня, т. е. внутренний пакет IP.

Имеющиеся на сегодня спецификации протокола требуют, чтобы любая реализация поддерживала следующие алгоритмы для вычисления кода аутентификации сообщения: HMAC – MD5 и HMAC – SHA1. Кроме того, протокол AH может быть реализован с использованием отечественных алгоритмов хэширования и формирования цифровой подписи СТБ 34.101.45-2013 [1] и СТБ 34.101.77-2016 [2], а также российских алгоритмов хэширования и формирования цифровой подписи ГОСТ Р 34.10-2012 [3] и ГОСТ Р 34.11-2012 [4] или алгоритма криптографического преобразования ГОСТ 28147-89 [5] в режиме вычисления имитовставки.

2.4.2 Протокол ESP

Протокол ESP выполняет все функции протокола AH по защите данных и обеспечивает их конфиденциальности путем шифрования внутреннего пакета IP. Пакет ESP содержит следующие поля:

1) **Индекс параметров защиты (32 бита).** Идентифицирует защищенную связь.

2) **Последовательный номер (32 бита).** Значение счетчика, используемого для защиты от атак воспроизведения, как и при использовании протокола AH.

3) **Значимые данные (переменной длины).** Внутренний пакет IP, который защищается шифрованием.

4) **Заполнитель (0–255 байт).** Поле заполняется нулями до кратности целому числу байтов в соответствии с форматами блоков используемых алгоритмов шифрования.

5) **Длина заполнителя (8 бит).** Указывает число байтов заполнителя, предшествующего данному полю.

6) **Следующий заголовок (8 бит).** Идентифицирует тип данных, содержащихся в поле значимых данных, указывая первый заголовок значимых данных.

7) **Данные аутентификации (переменной длины).** Поле переменной длины, содержащее код аутентификации сообщения, вычисляемый для данного пакета ESP, без поля данных аутентификации.

Сервис ESP предполагает шифрование полей значимых данных, заполнителя, длины заполнителя и следующего заголовка. Если для алгоритма, используемого при шифровании значимых данных, требуется синхронизация данных, то необходимая синхропосылка вставляется в начало поля значимых данных.

Существующие спецификации требуют, чтобы любая реализация поддерживала использование алгоритма DES в режиме со сцеплением блоков (CBC), однако могут применяться и другие алгоритмы шифрования, например: 3DES, RC5, AES и др. При обеспечении аутентификации сообщений в протоколе ESP используются механизмы, рассмотренные в подразделе 2.1.

2.4.3 Режимы работы протоколов

Каждый из протоколов AH и ESP поддерживает два режима работы – транспортный и туннельный.

В транспортном режиме механизмы безопасности применяются только для протоколов, начиная с транспортного (TCP) уровня и выше, оставляя данные самого сетевого уровня (заголовок IP) без дополнительной информации, вставляемой протоколами в пакет.

Туннельный режим характеризуется тем, что обеспечивает защиту также и данных сетевого уровня путем добавления нового IP-заголовка. После определения ассоциаций безопасности (например, между двумя шлюзами) истинные адреса хостов отправления и назначения (и другие служебные поля) полностью защищаются от модификаций для AH или вообще скрываются для ESP, а в но-

вый заголовок выставляются адреса и другие данные шлюзов (отправления/получения). Для обеспечения высокого уровня безопасности, применение обоих протоколов совмещается.

2.4.4 Ассоциация безопасности

Концепция ассоциаций безопасности (SA) является основополагающей для IPSec. SA определяет взаимоотношения между двумя или более участниками безопасной связи и описывает, какие сервисы буду использованы для обеспечения безопасности: алгоритмы шифрования, алгоритмы аутентификаций и общие сессионные ключи.

SA всегда однонаправлена, поэтому для одного двунаправленного соединения между двумя участниками требуются две SA (по одной на каждое направление).

Набор всех SA, установленных на узле для связи с другим узлом (узлами), хранится в специальной БД ассоциаций безопасности (SAD). Каждый узел поддерживает две SAD: одну для входящего трафика, вторую – для исходящего. В зависимости от реализации может потребоваться несколько пар SAD для мультиинтерфейсных узлов – по одной SAD на каждый интерфейс.

Когда у узла существует несколько соединений с другим узлом (узлами), необходимо определить, которую из SA применять к пакетам какого соединения. Для этой цели служит база данных политик безопасности (SPD). Запись SPD состоит из полей, связывающих SA с идентификатором пакетов соединения – селектором. Селектор состоит из следующих параметров:

1) IP-адрес назначения (индивидуальный, групповой, широковещательный), диапазон адресов + маска или групповой символ.

2) IP-адрес источника (индивидуальный, групповой, широковещательный), диапазон адресов + маска или групповой символ.

3) Имя в двух вариантах: идентификатор пользователя и наименование системы в формате DNS или X.500.

4) Уровень секретности данных по меткам IPSO/CIPSO (такие как «не определено», «коммерческая собственность», «секретно» и т. п.). Этот параметр опционален.

5) Протокол транспортного уровня.

6) Порт назначения.

7) Порт источника.

Протокол безопасности IPSec предпочтительно использовать при организации частных виртуальных сетей VPN, хотя технологии VPN могут включать в себя и схемы защиты собственной разработки.

2.5 Протоколы управления ключами

Поскольку основным механизмом обеспечения безопасности данных в VPN являются криптографические методы, участники защищенного соединения должны наладить обмен соответствующими криптографическими ключами. Обеспечить настройку процесса такого обмена можно вручную и автомати-

чески. Первый способ допустим для небольшого количества достаточно статичных систем, а в общем случае это производится автоматически.

Для автоматического обмена ключами по умолчанию используется протокол управления ключами в Интернете (IKMP). Дополнительно или альтернативно могут быть применены другие протоколы, такие как SKIP или Kerberos.

2.5.1 Протокол IKMP

Протокол IKMP совмещает в себе три отдельных протокола:

– Протокол защищенной связи и управления ключами в Интернете (ISAKMP) – обеспечивает каркас схемы управления ключами в Интернете и необходимые форматы процедуры согласования атрибутов защиты. Он не регламентирует использование конкретного алгоритма обмена ключами, а предлагает набор типов сообщений, позволяющих задействовать любой подобный алгоритм.

– Протокол определения ключей Oakley – является конкретным алгоритмом обмена ключами, основанным на усовершенствованной схеме обмена ключами Диффи – Хеллмана.

– Механизм безопасного обмена ключами в Интернете (SKEMI) – описывает многофункциональные технологии, предоставляющие такие услуги защиты, как анонимность, защита от отказа передачи или получения сообщений и быстрое обновление ключей.

В ходе установления защищенной связи IKMP согласовывает следующие атрибуты: алгоритм шифрования, алгоритм хэширования, метод аутентификации и данные о группе преобразования алгоритма Диффи – Хеллмана.

Аутентификация может быть произведена с помощью следующих методов:

– шифрования с симметричным ключом (аутентификация осуществляется путем шифрования параметров обмена с использованием секретного ключа, известного обеим сторонам соединения до начала установления соединения);

– шифрования с открытым ключом (аутентификация обмена данными осуществляется с помощью шифрования некоторых параметров обмена с использованием открытого ключа получателя);

– цифровой подписи (аутентификация обмена данными осуществляется с помощью подписи доступного обеим сторонам хэш-кода с использованием личного ключа).

Алгоритм Диффи – Хеллмана предполагает предварительное соглашение о двух глобальных параметрах (q – большое простое число, являющееся модулем конечного поля; a – основание степени) и определяет следующее взаимодействие между сторонами A и B.

Сторона A выбирает случайное число X_A, которое будет личным ключом A, и передает стороне B открытый ключ $Y_A = a^{X_A}$. Точно так же сторона B выбирает случайное число X_B, которое будет личным ключом B, и передает сто-

роне A открытый ключ $Y_B = a^{X_B}$. Каждая из сторон теперь может вычислить секретный сеансовый ключ по формуле

$$K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q = a^{X_A X_B} \bmod q.$$

Имеющиеся сегодня спецификации протокола определяют следующие группы для обмена ключами Диффи – Хеллмана:

1) Возведение в степень в арифметике классов вычетов с 768-битовым модулем:

$$q = 2^{768} - 2^{704} + \left(\left\lfloor 2^{638} \cdot \pi \right\rfloor + 149686\right)2^{64} - 1,\ a = 2.$$

2) Возведение в степень в арифметике классов вычетов с 1024-битовым модулем:

$$q = 2^{1024} - 2^{960} + \left(\left\lfloor 2^{894} \cdot \pi \right\rfloor + 129093\right)2^{64} - 1,\ a = 2.$$

ISAKMP содержит две фазы согласования ключей. В первой фазе происходит создание защищенного канала, во второй – согласование и обмен ключами, установление защищенной связи. Сообщение ISAKMP состоит из заголовка и следующих за ним значимых данных и передается с помощью транспортного протокола UDP.

2.5.2 Протокол SKIP

Отличительной особенностью протокола SKIP является исключительное использование в качестве криптографического алгоритма метода Диффи – Хеллмана. Однако в данном протоколе совместно используемый ключ $K = a^{X_A X_B} \bmod q$ является долговременным и его не требуется менять для каждого сеанса передачи данных.

Поскольку K – долговременный ключ, его использование для защиты самих данных небезопасно, так как дает возможность накопления материала для криптоанализа. Поэтому для шифрования самих данных применяется отдельный сеансовый ключ K_C, а долговременный ключ K используется только для шифрования сеансового ключа. Тогда зашифрование IP-пакета B выглядит следующим образом:

$$E_K(K_C) \| E_{K_C}(B),$$

где || – операция конкатенации.

Аутентификация в данном случае обеспечивается предположением, что если пакет успешно расшифрован, значит, он был зашифрован именно тем, кто знает секретный ключ отправителя, т. е. самим отправителем.

В качестве дополнительной меры обеспечения защиты используется механизм учета количества использований долговременного ключа, который можно определить в виде

$$K_n = h(K, n),$$

где h – хэш-функция;

n – постоянно увеличивающийся счетчик.

Таким образом, обеспечивается дополнительная защита от возможного повторения посылки пакета, скопированного нарушителем в предыдущем сеансе.

3 МЕХАНИЗМЫ УПРАВЛЕНИЯ ДОСТУПОМ К ИНФОРМАЦИИ

К настоящему времени разработано несколько механизмов управления доступом, причем наиболее распространенными являются следующие:

– управление доступом по ключам;
– управление доступом по спискам;
– управление доступом с использованием матриц;
– управление доступом по уровням (кольцам) секретности.

3.1 Управление доступом по ключам (мандатная система доступа)

Доступ по ключам состоит в том, что пользователю выдаются ключи на доступ к соответствующим ресурсам системы. При каждом обращении к защищаемому ресурсу осуществляется проверка наличия ключа. Сама процедура управления является достаточно простой: предъявленный ключ сравнивается с эталонным, хранящимся вместе с данными или в специально защищенном каталоге, и по результатам сравнения принимается решение о допуске (рисунок 3.1).

Однако запоминание или хранение пользователем ключей для всех разрешенных ему ресурсов затруднительно. Поэтому в системе разграничения доступа для пользователя создается каталог ключей, обеспечивающих доступ к определенным ресурсам. Каталог ключей защищается ключом каталога. При такой организации пользователю не требуется непосредственно предъявлять ключ для доступа к конкретному объекту. В этом случае пользователь предъявляет ключ каталога, а специальная программа выбирает из каталога ключ данных для его сравнения с эталонным (рисунок 3.2).

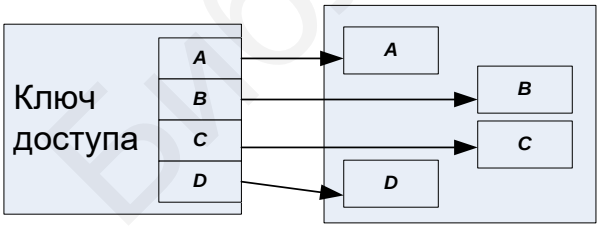


Рисунок 3.1 – Прямое сравнение ключей

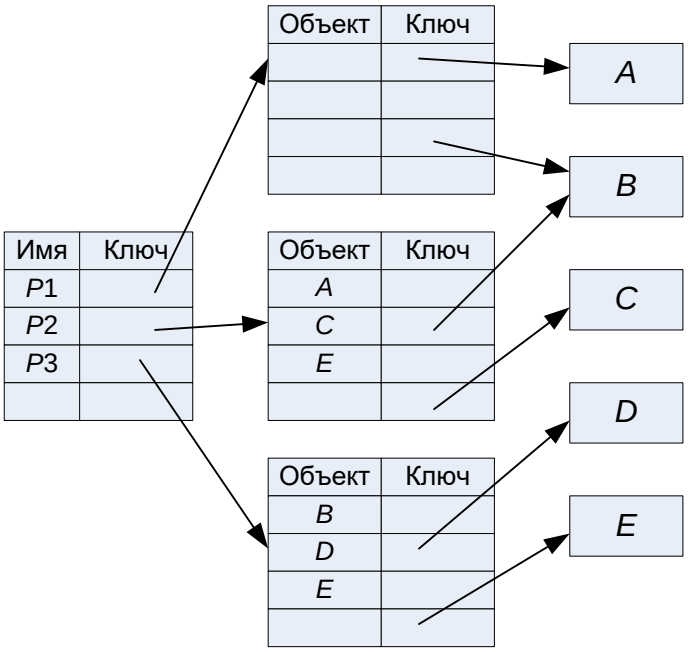


Рисунок 3.2 – Сравнение ключей через каталог

При организации совместной обработки информации используется таблица, содержащая имена пользователей и ключи каталогов, чем обеспечивается любая структура полномочий доступа, когда полномочия не изменяются или изменяются редко. Однако во многих случаях требуется, чтобы один пользователь, создав новый массив информации, имел возможность передать полномочия на доступ к нему другому пользователю.

Передача полномочий может осуществляться путем предоставления возможности записи ключа одного пользователя *А* в каталог другого пользователя *Б*, но при этом возможно разрушение других ключей. Более «безопасный» способ заключается в использовании общего сегмента («почтового ящика»), через который они посылают друг другу сообщения и новые ключи. Так как количество сегментов велико (для *N* пользователей необходимо *N* сегментов), то необходим механизм динамического создания сегментов.

Среди достоинств мандатной системы прежде всего нужно назвать эффективность, простоту и гибкость. Эффективностью система обязана тому, что проверка корректности обращения всегда выполняется за один шаг (путем проверки ключа), если он предоставляется непосредственно пользователем. Простота системы объясняется тем, что по смыслу ключи можно считать разновидностью адресных ссылок. Наконец, гибкость заложена в основном принципе мандатной системы, согласно которому пользователь сам определяет место для хранения ключей.

С другой стороны, можно отметить некоторые сложности при работе с мандатной системой: проблема отслеживания полномочий и необходимость контроля над передачей полномочий.

Основная сложность при работе с ключами заключается в том, что при копировании ключа всегда происходит передача соответствующих полномочий. Связь этих двух процессов (не всегда желательная) может быть разрушена только путем создания ссылок при каждом копировании. Для получения копии достаточно располагать экземпляром ключа, так что контроль над процессом передачи полномочий представляет собой самостоятельную задачу. Ограничение права на копирование позволяет контролировать распределение полномочий, однако при этом система усложняется и становится менее гибкой. Таким образом, использование ключей более всего уместно в ситуациях, когда нужно передавать параметры доступа от одной процедуры к другой и система полномочий редко изменяется.

3.2 Управление доступом по спискам

В списковых системах права пользователей на доступ заданы в виде списков. При этом либо для каждого защищаемого объекта задан список пользователей, имеющих право доступа к нему, либо для каждого пользователя задан перечень тех объектов, к которым ему разрешен доступ. В любом случае процедуры управления осуществляются в такой последовательности (рисунок 3.3):

а) по данным, содержащимся в запросе, выбирается соответствующая строка списка, перечень пользователей, допущенных к запрашиваемому объекту, или пе-

речень объектов, к которым допущен обратившийся с запросом пользователь;

б) в выбранной строке проверяется наличие имени пользователя, обратившегося с запросом, или имени объекта, к которому обращается пользователь;

в) по данным проверки принимается решение о допуске: при положительном решении доступ к запрашиваемым данным разрешается, при отрицательном – запрос блокируется.

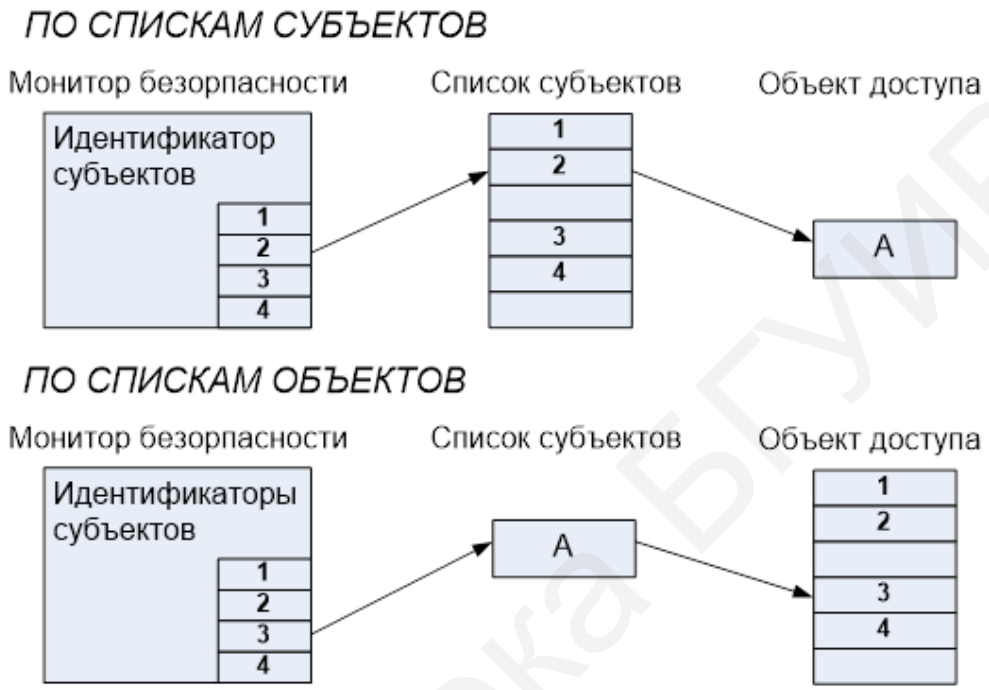


Рисунок 3.3 – Управление доступом по спискам

В списковой системе упрощается задача отмены полномочий. При вычеркивании имени пользователя из списка допуска немедленно блокируются все дальнейшие обращения к данному объекту со стороны этого пользователя.

Перечень пользователей, имеющих право доступа к некоторому ресурсу, может быть получен непосредственно из его списка допуска.

Часто неудобно перечислять имена всех пользователей, имеющих право на доступ к некоторому объекту. Действительно, такой список может оказаться слишком длинным, либо подверженным частым изменениям. В данных ситуациях в системе со списками доступов позволяется вводить групповые идентификаторы, каждым из которых может пользоваться некоторая группа лиц. При обнаружении в списке допуска такого идентификатора система обеспечивает доступ к информации всем членам группы. Кроме того, можно усовершенствовать схему проверки пропуска таким образом, чтобы ключами поиска в списке допуска были одновременно несколько идентификаторов.

К недостаткам списковой системы относятся: обращение к информационному ресурсу в несколько приемов; процесс поиска списков более медленный по сравнению с мандатной системой; выделение памяти для хранения списков допуска, имеющих переменную длину, на практике является сложной задачей.

3.3 Управление доступом на основе матриц

Управление доступом с помощью матриц является более гибким по сравнению с управлением по спискам, оно позволяет не только запрещать или разрешать доступ к объектам, но и регулировать характер выполняемых операций (чтение, запись, модификация данных и т. п.). Обеспечивается это тем, что полномочия пользователей задаются в виде матрицы, по строкам которой представлен список пользователей, а по столбцам – перечень объектов защищаемого ресурса (рисунок 3.4). Элементами матрицы доступа являются коды, соответствующие действиям, которые могут быть выполнены пользователем над компонентами ресурса. Множество возможных прав определяется разрядностью кода. Так, если код прав будет двухразрядным, то различные значения кода могут иметь, например, такое содержание: 00 – доступ запрещен, 01 – разрешается только чтение, 10 – разрешается только запись, 11 – разрешается и чтение и запись.

Процедуры управления доступом с помощью матриц выполняются в следующем порядке:

а) по имени пользователя, содержащемся в запросе, определяется номер строки матрицы, отражающий полномочия доступа пользователя;

б) по имени объекта доступа, содержащемся в запросе, определяется номер столбца матрицы;

в) по номеру строки и номеру столбца выбирается элемент матрицы, содержащий информацию о действиях, которые разрешается выполнять пользователю с данным объектом;

г) осуществляется сравнение запрашиваемого типа доступа с установленными в матрице: при положительном результате сравнения доступ к запрашиваемому объекту разрешается, при отрицательном – запрос блокируется.

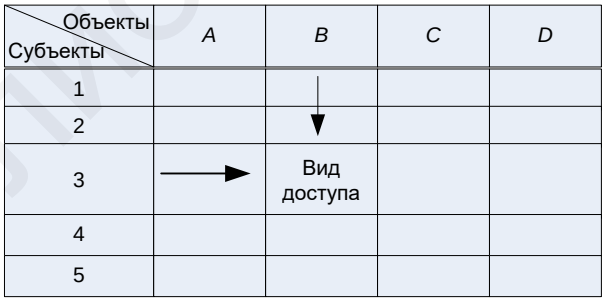


Рисунок 3.4 – Управление доступом на основе матриц

При необходимости элементы матрицы могут содержать указатели на процедуры. Эти процедуры исполняются при каждой попытке доступа с данного терминала к заданному элементу данных и могут принимать те решения о доступе, которые зависят от информации, представленной не столь очевидно, как в простой матрице доступа. Необходимость в такой сложной матрице доступа возникает, например, в следующих случаях:

1) Решение о доступе основывается на истории доступов других ресурсов.

Пользователь *А* может записывать данные в файл *F* только в том случае, если он не читал файл *G*.

2) Решение о доступе основывается на динамическом состоянии системы. Пользователь *Б* может открыть файл *H* только в то время, когда база данных, в которой размещен файл, находится в открытом состоянии.

3) Решение о доступе принимается на основе предписанного использования ресурса. Когда определенный пользователь вызывает программу сортировки для определенного файла, ее права читать данные от его имени выше прав пользователя при условии, что программа сортировки не возвращает данные пользователю.

4) Решение о доступе основывается на текущем значении ресурса. Пользователю может быть запрещено чтение данных из записи, в которой значение некоторого атрибута превышает определенный предел.

5) Решение о доступе основывается на значении определенных внутрисистемных переменных. Доступ может быть разрешен пользователю данной группы только в определенный интервал времени, исключая работу со специального терминала.

Недостатками метода управления по матрице полномочий считаются следующие обстоятельства. Во-первых, для больших систем с большим объемом защищаемых данных матрицы полномочий оказываются громоздкими. Во-вторых, динамическое ведение матриц в процессе функционирования системы является достаточно сложной задачей.

3.4 Управление доступом по уровням секретности

Управление доступом по уровням (кольцам) секретности заключается в том, что защищаемые ресурсы делятся на части в соответствии с уровнями их секретности. Могут быть выделены, например, такие уровни секретности информации: «общего доступа», «для служебного пользования», «секретно», «совершенно секретно», «особой важности». Полномочия каждого пользователя задаются максимальным уровнем секретности данных, доступ к которым ему разрешен. В соответствии с этим пользователю разрешается доступ ко всем данным, уровень секретности которых не выше его полномочий (рисунок 3.5).

В основе механизма управления доступом по уровням секретности лежит модель безопасности Белла – Лападулы (рисунок 3.6). В модели Белла – Лападулы взаимодействие субъектов и объектов описывается следующими правилами:

1) Процесс, работающий на любом уровне секретности, может читать только объекты своего уровня или более низких уровней секретности.

2) Процесс, работающий на любом уровне секретности, может записывать только в объекты своего уровня или более высоких уровней секретности.

Итак, процессы могут читать снизу и записывать наверх, но не наоборот. Если система гарантированно реализует эти два свойства, можно доказать, что не будет утечки информации с уровня большей секретности на уровень меньшей секретности.

На рисунке 3.6 сплошными стрелками от объекта к процессу показано направление информации при ее чтении процессом из объекта. Аналогично

штриховая стрелка от процесса к объекту означает запись данных в объект. Таким образом, вся информация «течет» по направлению стрелок. Например, процесс B может читать данные из объекта 1, но не из объекта 3.

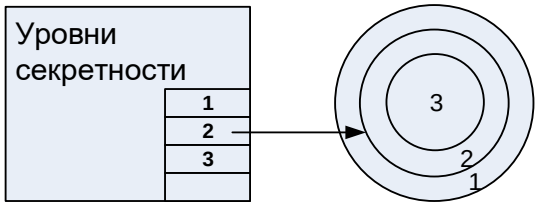


Рисунок 3.5 – Управление доступом по уровням секретности

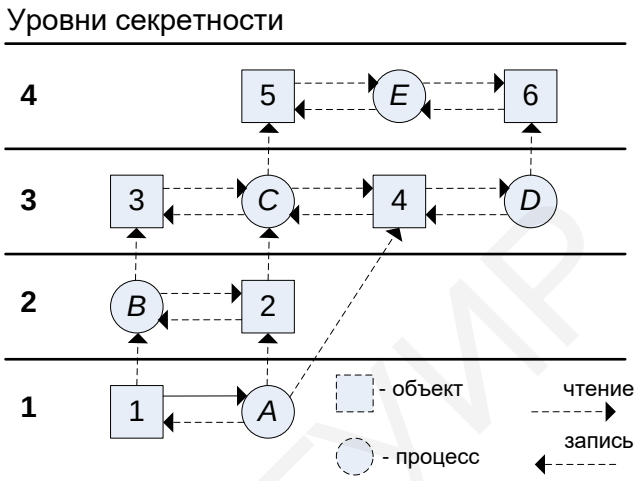


Рисунок 3.6 – Модель безопасности Белла – Лападулы

Правила модели Белла – Лападулы утверждают, что все сплошные и штриховые стрелки могут направляться в стороны или вверх. Поскольку информация распространяется только горизонтально или вверх, она не может попасть с более высокого уровня на более низкий, поэтому не существует пути информации сверху вниз, чем и гарантируется ее защищенность.

4 КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

4.1 Основы криптографической защиты информации

Криптография представляет собой совокупность методов преобразования данных, направленных на то, чтобы сделать эти данные бесполезными для противника. Такое преобразование позволяет решить две главные проблемы защиты данных: **проблему секретности** – лишение противника возможности извлечь информацию из канала передачи и **проблему имитостойкости** – лишение противника возможности ввести ложную информацию в канал передачи или изменить сообщение так, чтобы изменился его смысл.

Проблемы секретности и имитостойкости тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой. Из двух названных проблем секретность рассматривается первой, как наиболее исследованная на протяжении веков. На рисунке 4.1 представлена модель канала передачи данных, обеспечивающая секретность благодаря криптографическому преобразованию.

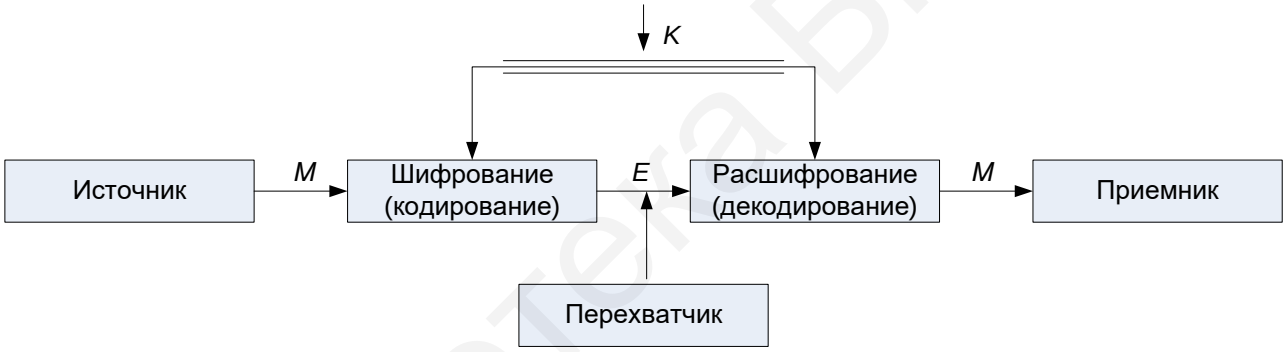


Рисунок 4.1 – Модель канала передачи с криптографической защитой

На рисунке 4.1 источник информации генерирует открытый текст или незашифрованное сообщение M, которое должно быть передано соответствующему получателю по незащищенному каналу, за которым следит перехватчик. Для того чтобы перехватчик не смог распознать сообщение M, отправитель шифрует его с помощью обратимого преобразования S_K и получает криптограмму или шифрованный текст $E = S_K(M)$, который отправляет получателю.

Законный получатель, приняв криптограмму E, расшифровывает ее с помощью обратного преобразования S_K^{-1} и получает исходное сообщение в виде открытого сообщения

$$M\colon S_K(E) = S_K^{-1}(S_K(M)) = M.$$

Преобразование S_K выбирается из семейства криптографических преобразований, называемых криптографической или общей системой.

Параметр, выбираемый в качестве отдельного преобразования, называется **криптографическим ключом** или просто **ключом**. Общая система – это набор инструкций (часть аппаратуры или программа ЭВМ), с помощью которо-

го можно зашифровать открытый текст и расшифровать зашифрованный текст различными способами (каждый выбирается с помощью конкретного ключа). Причем ключ K_i выбирается из конечного множества K, называемого пространством ключей.

Поскольку вся секретность сосредоточена в секретности ключа, то его надо передавать отправителю и получателю по защищенному каналу распространения ключей. На рисунке 4.1 этот канал показан экранированной линией.

Криптографическое преобразование может быть симметричным или асимметричным относительно преобразования расшифрования. Это важное свойство функции преобразования определяет два класса криптосистем: симметричные (одноключевые) и асимметричные (двухключевые, с открытым ключом) криптосистемы.

Схема симметричной криптосистемы с одним секретным ключом показана на рисунке 4.1. В ней используются одинаковые секретные ключи в блоке шифрования и блоке расшифрования.

На рисунке 4.2 представлена модель канала передачи данных, обеспечивающая секретность с использованием асимметричного криптографического кодирования. В этой криптосистеме один из ключей является известным (открытым), а другой – секретным.

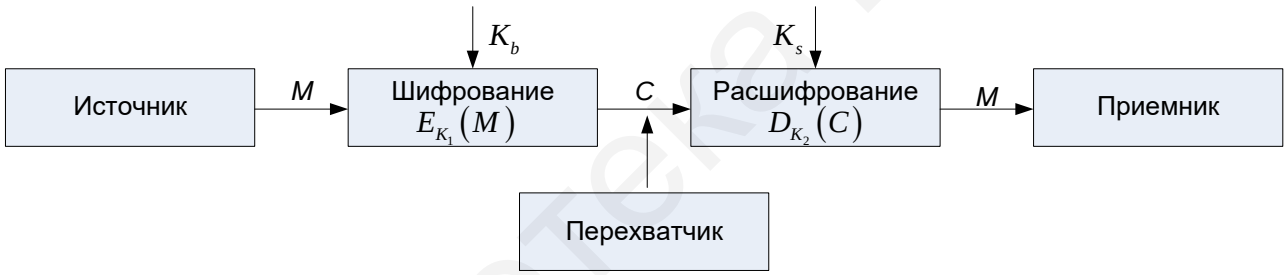


Рисунок 4.2 – Обобщенная схема асимметричной криптосистемы

В асимметричной криптосистеме для зашифрования данных используется один ключ, а для расшифрования – другой (отсюда и название – асимметричные). Первый ключ является *открытым* и может быть опубликован для использования всеми пользователями системы, которые зашифровывают данные. Расшифрование данных с помощью открытого ключа невозможно, для этого используется второй ключ, являющийся *секретным*. Разумеется, ключ расшифрования не может быть определен из ключа зашифрования.

В асимметричных системах нет необходимости в защищенном канале для передачи ключей, так как открытый ключ передается по незащищенному каналу.

На рисунке 4.3 показана криптографическая система, используемая для решения задачи имитостойкости. В этом случае противник не только видит все криптограммы, передаваемые по каналу, но может также изменять их по своему желанию. Законный получатель защищает себя от обмана измененными или ложными сообщениями, расшифровывая все полученные сообщения и принимая только те, которые зашифрованы правильным ключом.

Любая попытка со стороны перехватчика расшифровать криптограмму *E* для получения открытого текста *M* или зашифровать свой собственный текст *M'* для получения приемлемой криптограммы *E'* без получения ключа из канала распространения ключей называется **криптоанализом**. Если криптоанализ невозможен и криптоаналитик не может вывести *M* из *E* или *E'* из *M'* без предварительного получения ключа, то такая криптографическая система называется **криптостойкой.**

Для определения достаточности криптографической системы используют классификацию типов нападения, которым она может подвергаться. На рисунке 4.4 приведена схема классификации криптографических нападений, отображающая их влияние на уровень стойкости системы.

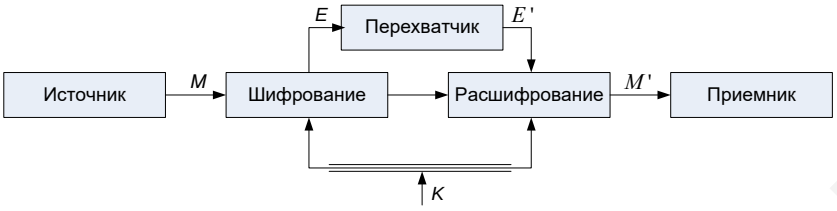


Рисунок 4.3 – Модель канала передачи с криптографической защитой, обеспечивающей имитостойкость

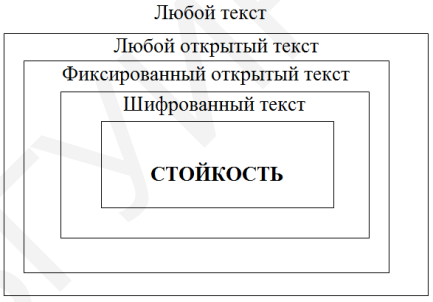


Рисунок 4.4 – Классификация криптографических нападений

Задача криптоаналитика заключается в том, чтобы найти ключ с целью его использования в дальнейшем для шифрования и расшифрования других сообщений.

Самым стойким является шифр, выдерживающий нападение при наличии любого текста, следующий по уровню стойкости – нападение при наличии известного открытого текста, содержание которого может определяться жесткой структурой формальных языков, используемых в программировании, либо таких данных, как коммерческие бланки или стандартные форматы пакетов в электронной почте. С другой стороны, если по криптографической системе, незащищенной от нападения при известном открытом тексте, посылаются сообщения, которые затем могут быть опубликованы, то все сообщения, зашифрованные с помощью того же самого ключа, будут скомпрометированы.

В этих двух случаях задача по определению следующего действующего ключа практически не решается, и криптографические системы считаются стойкими.

Самыми нестойкими являются системы, не выдерживающие атак криптоаналитика, который имеет в своем распоряжении только зашифрованные сообщения.

4.2 Стандарты симметричных систем шифрования

4.2.1 Стандарт шифрования данных ГОСТ 28147-89

Алгоритм криптографического преобразования данных для систем обработки информации в сетях ЭВМ, отдельных вычислительных комплексов и

ЭВМ был разработан в СССР и опубликован в виде государственного стандарта ГОСТ 28147-89 в 1989 году. Алгоритм криптографического преобразования данных предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации.

Алгоритм шифрования данных представляет собой 64-битовый блочный алгоритм с 256-битовым ключом и имеет четыре режима работы: 1) простая замена (режим кодовой книги); 2) гаммирование; 3) гаммирование с обратной связью; 4) выработка имитовставки.

4.2.1.1 Режим простой замены. Схема алгоритма шифрования данных в режиме простой замены приведена на рисунке 4.5.

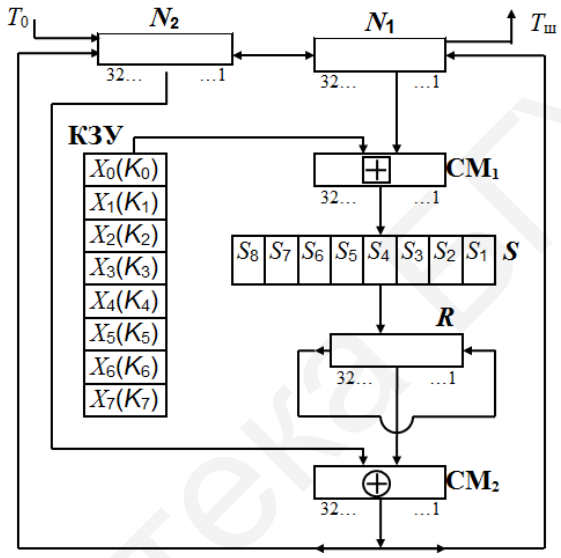


Рисунок 4.5 – Схема реализации режима простой замены

Обозначения на рисунке 4.5:

– N_1, N_2 – 32-разрядные накопители;

– $СМ_1$ – 32-разрядный сумматор по модулю 2^{32} (⊞);

– $СМ_2$ – 32-разрядный сумматор по модулю 2 (⊕);

– R – 32-разрядный регистр циклического сдвига;

– КЗУ – ключевое запоминающее устройство на 256 бит, состоящее из восьми 32-разрядных накопителей X_0, X_1, X_2, ..., X_7;

– S – блок подстановки, состоящий из восьми узлов замены (S-блоков замены) S_1, S_2, S_3, ..., S_7, S_8.

4.2.1.2 Зашифрование открытых данных в режиме простой замены. Открытые данные, подлежащие зашифрованию, разбивают на 64-разрядные блоки T_0. Процедура зашифрования 64-разрядного блока T_0 в режиме простой замены включает 32 цикла (j = 1…32). В ключевое запоминающее устройство вводят 256 бит ключа K в виде восьми 32-разрядных подключей (чисел) K_i: $K=K_7K_6K_5K_4K_3K_2K_1K_0$. Последовательность битов блока T_0 разбивают на две по-

следовательности по 32 бита, которые вводят в накопители N_1 и N_2 перед началом первого цикла зашифрования.

В первом цикле начальное заполнение накопителя N_1 суммируется по модулю 2^{32} в сумматоре $СМ_1$ с заполнением накопителя X_0, при этом значение накопителя N_1 сохраняется. Результат суммирования преобразуется в блоке подстановки S и полученный вектор поступает на вход регистра R, где циклически сдвигается на 11 шагов в сторону старших разрядов. Результат сдвига суммируется поразрядно по модулю 2 в сумматоре $СМ_2$ с 32-разрядным заполнением накопителя N_2 . Полученный в $СМ_2$ результат записывается в N_1, при этом старое заполнение N_1 переписывается в N_2. Первый цикл завершен.

Последующие циклы осуществляются аналогично, при этом во втором цикле из КЗУ считывают заполнение X_1 – подключ K_1, в третьем цикле – подключ K_2 и т. д., в восьмом цикле – подключ K_7. В циклах с 9-го по 16-й, а также в циклах с 17-го по 24-й подключи из КЗУ считываются в том же порядке: $K_0, K_1, K_2, \ldots, K_6, K_7$. В последних восьми циклах, с 25-го по 32-й порядок считывания подключей из КЗУ обратный: $K_0, K_1, K_2, \ldots, K_6, K_7$. Таким образом, при зашифровании в 32 циклах осуществляется следующий порядок выборки из КЗУ подключей:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

В 32-м цикле результат из сумматора $СМ_2$ вводится в накопитель N_2, а в накопителе N_1 сохраняется прежнее заполнение. Полученные после 32-го цикла зашифрования заполнения накопителей N_1 и N_2 являются блоком зашифрованных данных $T_ш$, соответствующим блоку открытых данных T_0.

Остальные блоки открытых данных зашифровываются в режиме простой замены аналогично.

4.2.1.3 Расшифрование в режиме простой замены. Криптосхема, реализующая алгоритм расшифрования в режиме простой замены, имеет тот же вид, что и при зашифровании. В КЗУ вводят 256 бит ключа, на котором осуществлялось зашифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки $T_ш$ по 64 бита в каждом. Расшифрование осуществляется по тому же алгоритму, что и зашифрование, с тем изменением, что заполнения накопителей $X_0, X_1, \ldots, X_7$ считываются из КЗУ в циклах расшифрования в следующем порядке:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

Полученные после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных T_0, соответствующий блоку зашифрованных данных $T_ш$. Аналогично расшифровываются остальные блоки зашифрованных данных.

Режим простой замены допустимо использовать для шифрования данных только в ограниченных случаях – при выработке ключа и зашифровании его с обеспечением имитозащиты для передачи по каналам связи или для хранения в памяти ЭВМ.

4.2.1.4 Режим гаммирования. Криптосхема, реализующая алгоритм зашифрования в режиме гаммирования, показана на рисунке 4.6. Открытые данные разбивают на 64-разрядные блоки $T_0^{(i)}$, где $T_0^{(i)}$ – i-й 64-разрядный блок открытых данных, $i = 1…m$, а m определяется объемом шифруемых данных. Эти блоки поочередно зашифровываются в режиме гаммирования путем поразрядного сложения по модулю 2 в сумматоре $СМ_5$ с гаммой шифра $Г_ш$, которая вырабатывается блоками по 64 бита $Г_ш^{(i)}$, где $Г_ш^{(i)}$ – i-й 64-разрядный блок, $i = 1…m$.

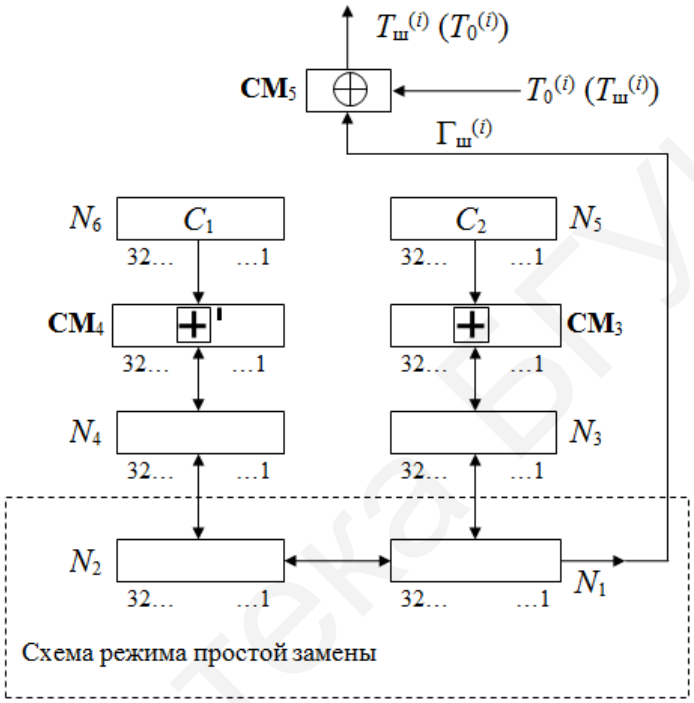


Рисунок 4.6 – Схема реализации режима гаммирования

Процедура зашифрования в режиме гаммирования выполняется следующим образом. В накопители N_6 и N_5 заранее записаны 32-разрядные двоичные константы C_1 и C_2, имеющие следующие значения (в шестнадцатеричной форме):

$$C_1 = 01010104_{(16)},\ C_2 = 01010101_{(16)}.$$

В КЗУ вводится 256 бит ключа, в накопители N_1 и N_2 – 64-разрядная двоичная последовательность (синхропосылка) $\tilde{S} = (S_1, S_2, ..., S_{64})$, которая зашифровывается в режиме простой замены. Результат зашифрования переписывается в 32-разрядные накопители N_3 и N_4 так, что заполнение N_1 переписывается в N_3, а заполнение N_2 – в N_4. Заполнение накопителя N_4 суммируется по модулю $(2^{32}–1)$ в сумматоре $СМ_4$ 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 $СМ_3$ 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 – в N_2, при этом заполнения N_3, N_4 сохраняются. Заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение

накопителей N_1, N_2 образует первый 64-разрядный блок гаммы шифра, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ с первым 64-разрядным блоком открытых данных $T_0^{(1)}$. В результате суммирования по модулю 2 получают первый 64-разрядный блок зашифрованных данных $T_ш^{(1)}$.

Аналогично вырабатываются блоки гаммы шифра $Г_ш^{(2)},Г_ш^{(3)},...,Г_ш^{(m)}$ и зашифровываются блоки открытых данных $T_0^{(2)},T_0^{(3)},...,T_0^{(m)}$. В канал связи передаются синхропосылка $\tilde{S}$ и блоки зашифрованных данных $T_ш^{(1)},T_ш^{(2)},...,T_ш^{(m)}$.

При расшифровании криптосхема имеет тот же вид, что и при зашифровании (рисунок 4.6).

Следует отметить, что расшифрование данных возможно только при наличии синхропосылки, которая не является секретным элементом шифра и может передаваться по каналам связи вместе с зашифрованными данными.

При расшифровании в КЗУ вводят 256 бит ключа, с помощью которого осуществлялось зашифрование данных $T_0^{(1)},T_0^{(2)},...,T_0^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка, и осуществляется процесс выработки m блоков гаммы шифра $Г_ш^{(1)},Г_ш^{(2)},...,Г_ш^{(m)}$. Блоки зашифрованных данных $T_ш^{(1)},T_ш^{(2)},...,T_ш^{(m)}$ суммируются поразрядно по модулю 2 в сумматоре $СМ_5$ с блоками гаммы шифра $Г_ш^{(1)},Г_ш^{(2)},...,Г_ш^{(m)}$. В результате получаются блоки открытых данных $T_0^{(1)},T_0^{(2)},...,T_0^{(m)}$, при этом $T_0^{(m)}$ может содержать меньше 64 разрядов.

4.2.1.5 Режим гаммирования с обратной связью. Криптосхема, реализующая алгоритм зашифрования в режиме гаммирования с обратной связью, имеет вид, показанный на рисунке 4.7.

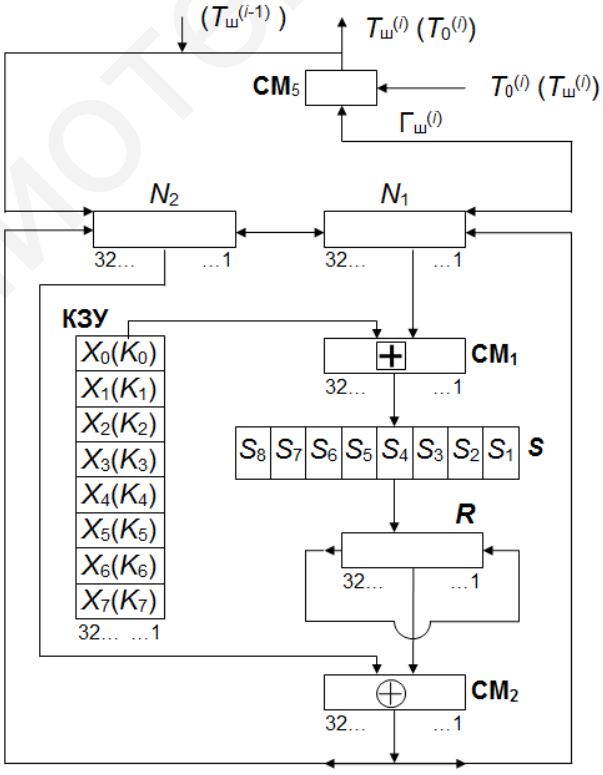


Рисунок 4.7 – Схема реализации режима гаммирования с обратной связью

Как видно из рисунка 4.7, открытые данные, разбитые на 64-разрядные блоки $T_0^{(1)},T_0^{(2)},...,T_0^{(m)}$, зашифровываются в режиме гаммирования с обратной связью путем поразрядного сложения по модулю 2 с гаммой шифра $\Gamma_ш$, которая вырабатывается блоками по 64 бита: $\Gamma_ш = (\Gamma_ш^{(1)},\Gamma_ш^{(2)},...,\Gamma_ш^{(m)})$. Число двоичных разрядов в блоке $T_0^{(m)}$ может быть меньше 64, при этом неиспользованная для шифрования часть гаммы шифра из блока $\Gamma_ш^{(m)}$ отбрасывается.

Процедура зашифрования данных в режиме гаммирования с обратной связью реализуется следующим образом. В КЗУ вводятся 256 бит ключа. В накопители N_1 и N_2 вводится синхропосылка $\tilde{S} = (S_1, S_2, ..., S_{64})$ из 64 бит. Исходное заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение накопителей N_1 и N_2 образует первый 64-разрядный блок гаммы шифра, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ с первым 64-разрядным блоком открытых данных. В результате получают первый 64-разрядный блок зашифрованных данных $Т_Ш^{(1)}$, который одновременно является также исходным состоянием накопителей N_1, N_2 для выработки второго блока гаммы шифра $\Gamma_Ш^{(2)}$, и поэтому по обратной связи $Т_Ш^{(1)}$ записывается в указанные накопители N_1 и N_2, заполнение которых зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_Ш^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ со вторым блоком открытых данных $T_0^{(2)}$:

$$\Gamma_Ш^{(2)} \oplus T_0^{(2)} = T_Ш^{(2)}.$$

Выработка последующих блоков гаммы шифра $\Gamma_Ш^{(i)}$ и зашифрование соответствующих блоков открытых данных $T_0^{(i)}$ ($i = 3…m$) производится аналогично.

Если длина последнего m-го блока открытых данных $T_0^{(m)}$ меньше 64 разрядов, то из $\Gamma_Ш^{(m)}$ используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

В канал связи передаются синхропосылка $\overline{S}$ и блоки зашифрованных данных $T_ш^{(1)},T_ш^{(2)},...,T_ш^{(m)}$.

При расшифровании криптосхема имеет тот же вид, что и при зашифровании (рисунок 4.7). Реализация процедуры расшифрования зашифрованных данных в режиме гаммирования с обратной связью происходит следующим образом. В КЗУ вводят 256 бит того же ключа, на котором осуществлялось зашифрование открытых блоков, а в накопители N_1 и N_2 вводится синхропосылка $\overline{S}$. Исходное заполнение накопителей N_1 и N_2 (синхропосылка $\overline{S}$) зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1 и N_2 образует первый блок гаммы шифра $\Gamma_Ш^{(1)}$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ с блоком зашифрованных данных $Т_Ш^{(1)}$. В результате получается первый блок открытых данных:

$$T_0^{(1)} = \Gamma_ш^{(1)} \oplus T_ш^{(1)}.$$

Блок зашифрованных данных $T_ш^{(1)}$ является исходным заполнением накопителей N_1 и N_2 для выработки второго блока гаммы шифра $\Gamma_Ш^{(2)}$. Полученное заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены.

Образованный в результате зашифрования блок $Г_Ш^{(2)}$ суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ со вторым блоком зашифрованных данных $T_Ш^{(2)}$ и в результате получают второй блок открытых данных. Аналогично в N_1, N_2 последовательно записывают блоки зашифрованных данных $T_ш^{(2)},T_ш^{(3)},...,T_ш^{(m)}$, из которых в режиме простой замены вырабатываются блоки гаммы шифра $Г_Ш^{(3)},Г_Ш^{(4)},...,Г_Ш^{(m)}$. Блоки гаммы шифра суммируются поразрядно по модулю 2 в сумматоре $СМ_5$ с блоками зашифрованных данных $T_ш^{(3)},T_ш^{(4)},...,T_ш^{(m)}$. В результате получают блоки открытых данных $T_0^{(3)},T_0^{(4)},...,T_0^{(m)}$, при этом последний блок $T_0^{(m)}$ может содержать меньше 64 разрядов.

4.2.2 Стандарт шифрования данных и контроля целостности СТБ 34.101.31-2011

СТБ 34.101.31-2011 [6] определяет семейство криптографических алгоритмов, предназначенных для обеспечения конфиденциальности и контроля целостности данных. Обрабатываемыми данными являются двоичные слова (сообщения). Криптографические алгоритмы шифрования и контроля целостности относятся к блочным алгоритмам с длиной блока $N = 128$ бит и длиной ключа $K = 256$ бит и построены на основе базового алгоритма шифрования блока данных. Схема алгоритма вычисления блока зашифрованных данных приведена на рисунке 4.8.

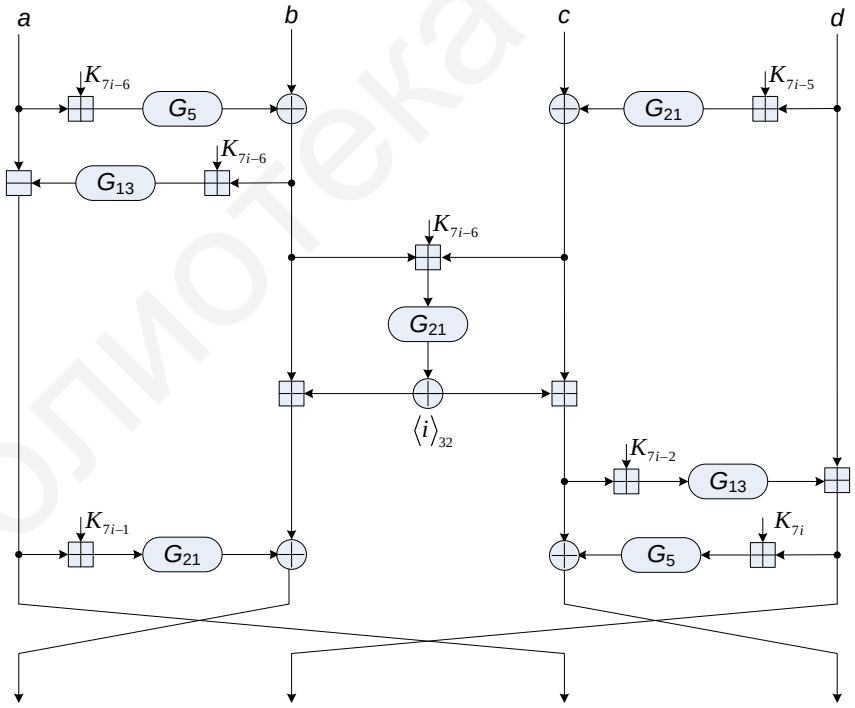


Рисунок 4.8 – Вычисления на i-м такте зашифрования

Входными данными алгоритмов зашифрования и расшифрования являются блок $X = 128$ бит и ключ $\theta = 256$ бит.

Входные данные подготавливаются следующим образом:

1) Слово X записывается в виде $X = X_1 \| X_2 \| X_3 \| X_4$, где $X_i = 32$ бита.

2) Ключ θ записывается в виде $\theta = \theta_1 \| \theta_2 \| ... \| \theta_8$ и определяются *тактовые ключи* $K_1 = \theta_1$, $K_2 = \theta_2$,..., $K_8 = \theta_8$, $K_9 = \theta_1$, $K_{10} = \theta_2$,..., $K_{56} = \theta_8$.

Выходными данными является блок Y = 128 бит – результат зашифрования либо расшифрования слова X на ключе θ:

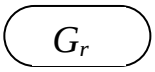
$$Y = F_0(X) \text{ либо } Y = F_0^{-1}(X).$$

На рисунке 4.8 используются следующие обозначения:

$\boxplus$ – 32-разрядный сумматор по модулю 2^{32};

$\boxminus$ – 32-разрядный вычитатель по модулю 2^{32};

$\oplus$ – поразрядный сумматор по модулю 2;



– преобразование 128-разрядного слова посредством подстановки и циклического сдвига влево на r разрядов.

Преобразование G_r ставит в соответствие слову $u = u_1 \| u_2 \| u_3 \| u_4$ слово

$$G_r(u) = RotH_i^r\left(H(u_1) \| H(u_2) \| H(u_3) \| H(u_4)\right),$$

где $H(u_i)$ – значение u_i, взятое из таблицы подстановки;

$RotH_i^r$ – циклический сдвиг влево на r разрядов.

В стандарте используется таблица подстановки с заранее установленными значениями (таблица 4.1).

Таблица 4.1 – Таблица подстановок

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**B1**	**94**	**BA**	**C8**	**0A**	**08**	**F5**	**3B**	**36**	**6D**	**00**	**8E**	**58**	**4A**	**5D**	**E4**
1	**85**	**04**	**FA**	**9D**	**1B**	**B6**	**C7**	**AC**	**25**	**2E**	**72**	**C2**	**02**	**FD**	**CE**	**0D**
2	**5B**	**E3**	**D6**	**12**	**17**	**B9**	**61**	**81**	**FE**	**67**	**86**	**AD**	**71**	**6B**	**89**	**OB**
3	**5C**	**BO**	**CO**	**FF**	**33**	**C3**	**56**	**B8**	**35**	**C4**	**05**	**AE**	**D8**	**E0**	**7F**	**99**
4	**E1**	**2B**	**DC**	**1A**	**E2**	**82**	**57**	**EC**	**70**	**3F**	**CC**	**F0**	**95**	**EE**	**8D**	**F1**
5	**C1**	**AB**	**76**	**38**	**9F**	**E6**	**78**	**CA**	**F7**	**C6**	**F8**	**60**	**D5**	**BB**	**9C**	**4F**
6	**F3**	**3C**	**65**	**7B**	**63**	**7C**	**30**	**6A**	**DD**	**4E**	**A7**	**79**	**9E**	**B2**	**3D**	**31**
7	**3E**	**98**	**B5**	**6E**	**27**	**D3**	**BC**	**CF**	**59**	**1E**	**18**	**1F**	**4C**	**5A**	**B7**	**93**
8	**E9**	**DE**	**E7**	**2C**	**8F**	**OC**	**0F**	**A6**	**2D**	**DB**	**49**	**F4**	**6F**	**73**	**96**	**47**
9	**06**	**07**	**53**	**16**	**ED**	**24**	**7A**	**37**	**39**	**CB**	**A3**	**83**	**03**	**A9**	**8B**	**F6**
A	**92**	**BD**	**9B**	**1C**	**E5**	**D1**	**41**	**01**	**54**	**45**	**FB**	**C9**	**5E**	**4D**	**0E**	**F2**
B	**68**	**20**	**80**	**AA**	**22**	**7D**	**64**	**2F**	**26**	**87**	**F9**	**34**	**90**	**40**	**55**	**11**
C	**BE**	**32**	**97**	**13**	**43**	**FC**	**9A**	**48**	**AO**	**2A**	**88**	**5F**	**19**	**4B**	**09**	**A1**
D	**7E**	**CD**	**A4**	**D0**	**15**	**44**	**AF**	**8C**	**A5**	**84**	**50**	**BF**	**66**	**D2**	**E8**	**8A**
E	**A2**	**D7**	**46**	**52**	**42**	**A8**	**DF**	**B3**	**69**	**74**	**C5**	**51**	**EB**	**23**	**29**	**21**
F	**D4**	**EF**	**D9**	**B4**	**3A**	**62**	**28**	**75**	**91**	**14**	**10**	**EA**	**77**	**6C**	**DA**	**1D**

В таблице 4.1 используется шестнадцатеричное представление слов $u \in \{0,1\}^8$. Если $u = IJ_{16}$, то значение $H(u)$ находится на пересечении строки I и столбца J. Например: $H(A2_{16}) = 9B_{16}$.

Для зашифрования блока X на ключе θ выполняются следующие шаги.

Шаг 1. Установить $a \leftarrow X_1,\ b \leftarrow X_2,\ c \leftarrow X_3,\ d \leftarrow X_4$.

Шаг 2. Для $i = 1, 2, ..., 8$ выполнить (см. рисунок 4.8):

1) $b \leftarrow b \oplus G_5(a \boxplus K_{7i-6})$;
2) $c \leftarrow c \oplus G_{21}(d \boxplus K_{7i-5})$;
3) $a \leftarrow a \boxminus G_{13}(b \boxplus K_{7i-4})$;
4) $e \leftarrow G_{21}(b \boxplus c \boxplus K_{7i-3}) \oplus \langle i \rangle_{32}$;
5) $b \leftarrow b \boxplus e$;
6) $c \leftarrow c \boxminus e$;
7) $d \leftarrow d \boxplus G_{13}(c \boxplus K_{7i-2})$;
8) $b \leftarrow b \oplus G_{21}(a \boxplus K_{7i-1})$;
9) $c \leftarrow c \oplus G_5(d \boxplus K_{7i})$;
10) $a \leftrightarrow b$;
11) $c \leftrightarrow d$;
12) $b \leftrightarrow c$.

Шаг 3. Установить $Y \leftarrow b \| d \| a \| c$.

Шаг 4. Возвратить Y.

Для расшифрования блока X на ключе θ выполняются следующие шаги.

Шаг 1. Установить $a \leftarrow X_1,\ b \leftarrow X_2,\ c \leftarrow X_3,\ d \leftarrow X_4$.

Шаг 2. Для $i = 1, 2, ..., 8$ выполнить (см. рисунок 4.8):

1) $b \leftarrow b \oplus G_5(a \boxplus K_{7i})$;
2) $c \leftarrow c \oplus G_{21}(d \boxplus K_{7i-1})$;
3) $a \leftarrow a \boxminus G_{13}(b \boxplus K_{7i-2})$;
4) $e \leftarrow G_{21}(b \boxplus c \boxplus K_{7i-3}) \oplus \langle i \rangle_{32}$;
5) $b \leftarrow b \boxplus e$;
6) $c \leftarrow c \boxminus e$;
7) $d \leftarrow d \boxplus G_{13}(c \boxplus K_{7i-4})$;
8) $b \leftarrow b \oplus G_{21}(a \boxplus K_{7i-5})$;
9) $c \leftarrow c \oplus G_5(d \boxplus K_{7i-6})$;
10) $a \leftrightarrow b$;
11) $c \leftrightarrow d$;
12) $b \leftrightarrow c$.

Шаг 3. Установить $Y \leftarrow c \| a \| d \| b$.

Шаг 4. Возвратить Y.

Криптографические алгоритмы шифрования и контроля целостности включают в себя следующие алгоритмы:

1) шифрование в режиме простой замены;
2) шифрование в режиме сцепления блоков;
3) шифрование в режиме гаммирования с обратной связью;

4) шифрование в режиме счетчика;
5) выработка имитовставки;
6) одновременное шифрование и имитозащита данных;
7) одновременное шифрование и имитозащита ключа;
8) хэширование.

Первые четыре алгоритма предназначены для обеспечения конфиденциальности сообщений. Каждый алгоритм включает режимы зашифрования и расшифрования. Стороны, располагающие общим ключом, могут организовать конфиденциальный обмен сообщениями путем их зашифрования перед отправкой и расшифрования после получения. В режимах простой замены и сцепления блоков шифруются сообщения, которые содержат хотя бы один блок, а в режимах гаммирования с обратной связью и счетчика – сообщения произвольной длины.

При шифровании в режимах сцепления блоков, гаммирования с обратной связью и счетчика, а также при одновременном шифровании и имитозащите данных используется синхропосылка $S = 128$ бит.

Синхропосылка является несекретным параметром, может добавляться к зашифрованному сообщению и передаваться вместе с ним. При шифровании в режимах гаммирования с обратной связью и счетчика, а также при одновременном шифровании и имитозащите данных должны использоваться уникальные синхропосылки. Уникальность означает, что при зашифровании или установке защиты на одном и том же ключе используются либо заведомо различные синхропосылки, либо вероятность совпадения синхропосылок пренебрежимо мала. В режиме сцепления блоков синхропосылка должна быть не только уникальной, но и случайной. Это означает, что вероятность угадать, какая синхропосылка будет использоваться, пренебрежимо мала. Синхропосылки можно вырабатывать случайным или псевдослучайным методом, строить по меткам времени, значениям монотонного счетчика, неповторяющимся номерам сообщений и др. В режиме сцепления блоков для построения синхропосылок предсказуемые значения не должны использоваться напрямую, а должны предварительно зашифроваться на том же ключе, который используется для шифрования сообщений.

В алгоритме выработки имитовставки и в алгоритмах одновременного шифрования и имитозащиты данных вычисляется либо проверяется имитовставка $T = 64$ бит. Если требуются не все, а $P < 64$ символов имитовставки, то должны использоваться первые P символов. При выборе P следует учитывать, что при навязывании ложного сообщения вероятность угадать с одной попытки его имитовставку, не зная ключ, равняется $1/2^p$.

4.2.2.1 Шифрование в режиме простой замены. Входными данными алгоритмов зашифрования и расшифрования являются сообщение $X \in \{0,1\}^*$ и ключ $\theta \in \{0,1\}^{256}$. Длина X должна быть не меньше 128.

Выходными данными является слово $Y \in \{0,1\}^{|X|}$ – результат зашифрования либо расшифрования X на ключе θ.

Входное сообщение X записывается в виде

$$X = X_1 \| X_2 \| ... \| X_n,\ |X_1| = |X_2| = ... = X_{n-1} = 128,\ 0 < |X_n| \le 128.$$

При шифровании словам X_i ставятся в соответствие слова $Y_i \in \{0,1\}^{|X_i|}$, из которых затем составляется Y.

Зашифрование сообщения X на ключе θ состоит в выполнении следующих шагов.

Шаг 1. Если $|X_n| = 128$, то для $i = 1,2,...,n$ выполнить:

$$Y_i \leftarrow F_0(X_i).$$

Шаг 2. Если $|X_n| < 128$, то для $i = 1,2,...,n-2$ выполнить:

$$Y_i \leftarrow F_0(X_i);$$
$$Y_{n\|r} \leftarrow F_0(X_{n-1});$$
$$Y_{n-1} \leftarrow F_0(X_{n\|r}).$$

Шаг 3. Установить $Y \leftarrow Y_1 \| Y_2 \| ... \| Y_n$.

Шаг 4. Возвратить Y.

Расшифрование сообщения X на ключе θ состоит в выполнении следующих шагов.

Шаг 1. Если $|X_n| = 128$, то для $i = 1,2,...,n$ выполнить:

$$Y_i \leftarrow F_0^{-1}(X_i).$$

Шаг 2. Если $|X_n| < 128$, то для $i = 1,2,...,n-2$ выполнить:

$$Y_i \leftarrow F_0^{-1}(X_i);$$
$$Y_{n\|r} \leftarrow F_0^{-1}(X_{n-1});$$
$$Y_{n-1} \leftarrow F_0^{-1}(X_{n\|r}).$$

Шаг 3. Установить $Y \leftarrow Y_1 \| Y_2 \| ... \| Y_n$.

Шаг 4. Возвратить Y.

4.2.2.2 Режим гаммирования с обратной связью. Входными данными алгоритмов зашифрования и расшифрования являются сообщение $X \in \{0,1\}^*$, ключ $\theta \in \{0,1\}^{256}$ и синхропосылка $S \in \{0,1\}^{128}$. Выходными данными является слово $Y \in \{0,1\}^{|X|}$ – результат зашифрования либо расшифрования X на ключе θ при использовании синхропосылки S.

Входное сообщение X записывается в виде

$$X = X_1 \| X_2 \| ... \| X_n,\ |X_1| = |X_2| = ... = X_{n-1} = 128,\ 0 < |X_n| \le 128.$$

При шифровании словам X_i ставятся в соответствие слова $Y_i \in \{0,1\}^{|X_i|}$, из которых затем составляется Y. При зашифровании используется вспомогательный блок $Y_0 \in \{0,1\}^{128}$, а при расшифровании – вспомогательный блок $X_0 \in \{0,1\}^{128}$.

Зашифрование сообщения X на ключе θ при использовании синхропосылки S состоит в выполнении следующих шагов.

Шаг 1. Установить $Y_0 \leftarrow S$.

Шаг 2. Для $i = 1,2,...,n$ выполнить:

$$Y_i \leftarrow X_i \oplus L_{|X_i|}(F_0(Y_{i-1})).$$

Шаг 3. Установить $Y \leftarrow Y_1 \| Y_2 \| ... \| Y_n$.

Шаг 4. Возвратить Y.

Расшифрование сообщения X на ключе θ при использовании синхропосылки S состоит в выполнении следующих шагов:

Шаг 1. Установить $X_0 \leftarrow S$.

Шаг 2. Для $i = 1,2,...,n$ выполнить:

$$Y_i \leftarrow X_i \oplus L_{|X_i|}\left(F_0\left(X_{i-1}\right)\right).$$

Шаг 3. Установить $Y \leftarrow Y_1 \| Y_2 \| ... \| Y_n$.

Шаг 4. Возвратить Y.

4.3 Асимметричные криптосистемы

4.3.1 Криптосистема шифрования данных RSA

Алгоритм RSA предложили в 1978 г. три автора: Р. Райвест (Rivest), А. Шамир (Shamir) и А. Адлеман (Adleman). Алгоритм получил свое название по первым буквам фамилий его авторов.

Надежность алгоритма основывается на трудности факторизации больших чисел в произведение простых множителей. В криптосистеме RSA открытый ключ $K_в$, секретный ключ $k_в$, сообщение M и криптограмма C принадлежат множеству целых чисел $\overline{Z_N} = \{0,1,2,...,N-1\}$, где N – модуль $N = P*Q$. Здесь P и Q – случайные большие простые числа. Для обеспечения максимальной безопасности выбирают P и Q равной длины и хранят в секрете. Множество $\overline{Z_N}$ с операциями сложения и умножения по модулю N образует арифметику по модулю N. Открытый ключ $K_в$ выбирают случайным образом так, чтобы выполнялись условия:

$$\begin{cases} 1 < K_в \le \varphi(N), \\ \text{НОД}\left(K_в, \varphi(N)\right) = 1, \\ \varphi(N) = (P-1)(Q-1), \end{cases} \tag{4.1}$$

где $\varphi(N)$ – функция Эйлера.

Второе из условий системы (4.1) означает, что открытый ключ $K_в$ и значение функции Эйлера $\varphi(N)$ должны быть взаимно простыми. Далее, используя расширенный алгоритм Евклида, вычисляют секретный ключ $k_в$, такой, что

$$k_в \cdot K_в \equiv 1\left(\text{mod}\,\varphi(N)\right) \text{ или } k_в = K_в^{-1}\left(\text{mod}(P-1)(Q-1)\right).$$

Это можно осуществить, так как при известной паре простых чисел (P,Q) можно легко найти $\varphi(N)$. Заметим, что $k_в$ и N должны быть взаимно простыми. Открытый ключ $K_в$ используют для шифрования данных, а секрет-

ный ключ $k_в$ – для расшифрования.

Преобразование шифрования определяет криптограмму *C* через пару (открытый ключ $K_в$, сообщение *M*) в соответствии со следующей формулой:

$$C = E_{K_в}(M) = E_B(M) = M^{K_в} \pmod N.$$

В качестве алгоритма быстрого вычисления значения *C* используют ряд последовательных возведений в квадрат целого *M* и умножений на *M* с приведением по модулю *N*.

Обращение функции $C = M^{K_в} \pmod N$, т. е. определение значения *M* по известным значениям *C*, $K_в$ и *N*, практически неосуществимо при $N \approx 2^{512}$. Однако обратную задачу, т. е. задачу расшифрования криптограммы *C*, можно решить, используя пару – секретный ключ $k_в$ и криптограмму *C* – по следующей формуле:

$$M = D_{k_в}(C) = D_B(C) = C^{k_в} \pmod N. \qquad (4.2)$$

Подставляя в уравнение (4.2) значение *C*, получаем

$$\left(M^{K_в}\right)^{k_в} = M \pmod N \text{ или } M^{K_в k_в} = M \pmod N. \qquad (4.3)$$

Сопоставляя выражения (4.2) и (4.3), получаем

$$k_в \cdot K_в = n \cdot \varphi(N) + 1 \text{ или } k_в \cdot K_в \equiv 1 \left(\bmod \varphi(N)\right). \qquad (4.4)$$

Именно поэтому для вычисления секретного ключа $k_в$ используют соотношение (4.4).

Следовательно, если криптограмму $C = M^{K_в} \pmod N$ возвести в степень $k_в$, то в результате восстанавливается исходный открытый текст *M*, так как

$$\left(M^{K_в}\right)^{k_в} = M^{K_в k_в} = M^{n \cdot \varphi(N)+1} \equiv M \pmod N.$$

Таким образом, получатель, который создает криптосистему, защищает два параметра: секретный ключ $k_в$ и пару чисел (P,Q), произведение которых дает значение модуля *N*. С другой стороны, отправителю известны значения модуля *N* и открытого ключа $K_в$. Противнику также известны лишь значения $K_в$ и *N*. Если бы он смог разложить число *N* на множители *P* и *Q*, то узнал бы «потайной ход» – тройку чисел $\{P, Q, K_в\}$, вычислил значение функции Эйлера $\varphi(N) = (P-1)(Q-1)$ и определил значение секретного ключа $k_в$.

Однако, как уже отмечалось, разложение очень большого *N* на множители вычислительно неосуществимо (при условии, что длины выбранных *P* и *Q* составляют не менее 100 десятичных знаков).

4.3.1.1 Процедуры зашифрования и расшифрования в криптосистеме RSA. Предположим, что пользователь *A* хочет передать пользователю *B* сообщение в зашифрованном виде, используя криптосистему RSA. В таком случае пользователь *A* выступает в роли отправителя сообщения, а пользователь *B* – в роли получателя. Как отмечалось выше, криптосистему RSA должен сформировать по-

лучатель сообщения, т. е. пользователь *B*. Рассмотрим последовательность действий пользователя *B* и пользователя *A*:

1) Пользователь *B* выбирает два произвольных больших простых числа *P* и *Q*.

2) Пользователь *B* вычисляет значение модуля $N = P \cdot Q$.

3) Пользователь *B* вычисляет значение функции Эйлера $\varphi(N) = (P-1)(Q-1)$ и выбирает случайным образом значение открытого ключа $K_{\text{в}}$ с учетом выполнения условий системы (4.1).

4) Пользователь *B* вычисляет значение секретного ключа $k_{\text{в}}$, используя расширенный алгоритм Евклида при решении сравнения $k_{\text{в}} \equiv K_{\text{в}}^{-1} \left(\text{mod}\, \varphi(N)\right)$.

5) Пользователь *B* пересылает пользователю *A* пару чисел $(N, K_{\text{в}})$ по незащищенному каналу.

Если пользователь *A* хочет передать пользователю *B* сообщение *M*, то выполняются следующие шаги:

1) Пользователь *A* разбивает исходный открытый текст *M* на блоки, каждый из которых может быть представлен в виде числа $M_i = 0,1,...,N-1$.

2) Пользователь *A* шифрует текст, представленный в виде последовательности чисел M_i по формуле $C_i = M_i^{K_{\text{в}}} (\text{mod}\, N)$ и отправляет криптограмму $C_1, C_2, ..., C_i$ пользователю *B*.

3) Пользователь *B* расшифровывает принятую криптограмму $C_1, C_2, ..., C_i$, используя секретный ключ $k_{\text{в}}$, по формуле $M_i = C_i^{k_{\text{в}}} (\text{mod}\, N)$.

В результате будет получена последовательность чисел M_i, которые представляют собой исходное сообщение *M*. Чтобы алгоритм RSA имел практическую ценность, необходимы возможность без существенных затрат генерировать большие простые числа и умение оперативно вычислять значения ключей $K_{\text{в}}$ и $k_{\text{в}}$.

Криптосистемы RSA реализуются как аппаратным, так и программным путем. Для аппаратной реализации операций зашифрования и расшифрования RSA разработаны специальные процессоры. Эти процессоры, реализованные на сверхбольших интегральных схемах (СБИС), позволяют выполнять операции RSA, связанные с возведением больших чисел в колоссально большую степень по модулю *N*, за относительно короткое время. Лучшими из серийно выпускаемых СБИС являются процессоры фирмы CYLINK, выполняющие 1024-битовое шифрование RSA.

Программная реализация RSA гораздо медленнее программной реализации симметричных алгоритмов шифрования. С развитием технологии эта оценка может несколько изменяться, но асимметричная криптосистема RSA никогда не достигнет быстродействия симметричных криптосистем.

4.3.2 Схема шифрования Эль Гамаля

Схема Эль Гамаля, предложенная в 1985 г., может быть использована как для шифрования, так и для цифровых подписей. Безопасность данной схемы

обусловлена сложностью вычисления дискретных логарифмов в конечном поле.

Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое число P и большое целое число G, причем $G < P$. Числа P и G могут быть распространены среди группы пользователей. Затем выбирают случайное целое число X, причем $X < P$. Число X является секретным ключом и не должно разглашаться. Далее вычисляют $Y = G^X \bmod P$. Число Y является открытым ключом.

Для того чтобы зашифровать сообщение M, выбирают случайное целое число K, такое, что $1 < K < P - 1$, где числа K и $(P - 1)$ являются взаимно простыми. Затем вычисляют числа $a = G^K \bmod P$, $b = Y^K M \bmod P$. Пара чисел (a, b) является шифротекстом. Заметим, что длина шифротекста вдвое больше длины исходного открытого текста M.

Для того чтобы расшифровать шифротекст (a, b), вычисляют

$$M = b / a^X \bmod P. \qquad (4.5)$$

Поскольку $a^X \equiv G^{KX} \bmod P$, $b / a^X \equiv Y^K M / a^X \equiv G^{KX} M / G^{KX} \equiv M \pmod P$, то соотношение (4.5) справедливо.

4.3.3 Комбинированный метод шифрования

Главным достоинством криптосистем с открытым ключом является их потенциально высокая безопасность: нет необходимости ни передавать, ни сообщать кому бы то ни было значения секретных ключей, ни убеждаться в их подлинности. В симметричных криптосистемах существует опасность раскрытия секретного ключа во время передачи. Однако алгоритмы, лежащие в основе криптосистем с открытым ключом, имеют следующие недостатки:

– генерация новых секретных и открытых ключей основана на генерации новых больших простых чисел, а проверка простоты чисел занимает много процессорного времени;

– процедуры шифрования и расшифрования, связанные с возведением в степень многозначного числа, достаточно громоздки.

Поэтому быстродействие криптосистем с открытым ключом обычно в сотни и более раз меньше быстродействия симметричных криптосистем с секретным ключом.

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для шифрования, передачи и последующего расшифрования только секретного ключа симметричной криптосистемы. Симметричная криптосистема применяется для шифрования и передачи исходного открытого текста. В результате криптосистема с открытым ключом не заменяет симметричную криптосистему с секретным ключом, а лишь дополняет ее, позволяя повысить в целом защищенность передаваемой информации. Такой подход иногда называют схемой электронного цифрового конверта.

Если пользователь *A* хочет передать зашифрованное комбинированным методом сообщение *M* пользователю *B*, то порядок его действий будет следующий:

1) Создать (например, сгенерировать случайным образом) симметричный ключ, называемый в этом методе сеансовым ключом K_S.

2) Зашифровать сообщение M на сеансовом ключе K_S.

3) Зашифровать сеансовый ключ K_S на открытом ключе $K_в$ пользователя *B*.

4) Передать по открытому каналу связи в адрес пользователя *B* зашифрованное сообщение вместе с зашифрованным сеансовым ключом.

Действия пользователя *B* при получении зашифрованного сообщения и зашифрованного сеансового ключа должны быть обратными:

1) Расшифровать на своем секретном ключе $k_в$ сеансовый ключ K_S.

2) С помощью полученного сеансового ключа K_S расшифровать и прочитать сообщение *M*.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь *B* сможет правильно расшифровать ключ K_S и прочитать сообщение *M*.

Таким образом, при комбинированном методе шифрования применяются криптографические ключи как симметричных, так и асимметричных криптосистем. Очевидно, выбор длин ключей для каждого типа криптосистемы следует осуществлять таким образом, чтобы злоумышленнику было одинаково трудно атаковать любой механизм защиты комбинированной криптосистемы.

В таблице 4.2 приведены распространенные длины ключей симметричных и асимметричных криптосистем, для которых трудность атаки полного перебора примерно равна трудности факторизации соответствующих модулей асимметричных криптосистем.

Таблица 4.2 – Длины ключей для симметричных и асимметричных криптосистем при одинаковой их криптостойкости

Длина ключа симметричной криптосистемы, бит	Длина ключа асимметричной криптосистемы, бит
56	384
64	512
80	768
112	1792
128	2304
256	9232

Комбинированный метод шифрования является наиболее рациональным, так как объединяет в себе высокое быстродействие симметричного шифрования и высокую криптостойкость, гарантируемую системами с открытым ключом.

5 МЕХАНИЗМЫ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

Электронная цифровая подпись (ЭЦП) – контрольная характеристика сообщения, которая формируется с использованием личного ключа, проверяется с использованием открытого ключа, служит для контроля целостности и подлинности сообщения и обеспечивает невозможность отказа от авторства.

Механизм электронной цифровой подписи определяет две процедуры:

1) подпись блока данных;

2) верификацию (проверку подписи) подписанного блока данных.

Первый процесс оперирует информацией, являющейся личной (т. е. единственной и конфиденциальной) по отношению к подписавшему лицу. Второй процесс использует процедуры и информацию, которые являются общественно доступными, но из которых не может быть выделена личная информация подписавшего лица.

Процесс подписания включает либо шифрование блока данных, либо выработку криптографического контрольного значения блока данных путем использования в качестве личного ключа личной информации подписавшего лица. Процесс проверки подписи включает применение процедур и информации общего пользования с целью определения образования подписи с помощью личной информации подписавшего лица.

Существенной характеристикой механизма подписи является то, что подпись может быть произведена только с использованием личной информации подписавшего лица. Таким образом, при верификации подписи можно впоследствии в любой момент времени доказать третьему лицу (например, судье или арбитру), что только единственный обладатель личной информации мог произвести эту подпись.

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

СТБ 34.101.45-2013 [1] определяет алгоритмы ЭЦП, которые предназначены для контроля целостности и подлинности сообщений. Автор сообщения использует свой личный ключ для выработки ЭЦП, а связанный с личным ключом открытый ключ применяется другими сторонами для проверки ЭЦП. При правильном управлении ключами корректность проверяемой подписи означает, что она была выработана владельцем личного ключа и после этого сообщение не изменялось. Только владелец личного ключа может выработать корректную ЭЦП, что не позволяет ему отказаться от авторства сообщения и может быть использовано другими сторонами для доказательства такого авторства.

Алгоритмы ЭЦП установлены также в СТБ 1176.2-99 [7]. Переход от алгоритмов СТБ 1176.2-99 к алгоритмам СТБ 34.101.45-2013 позволит уменьшить время выработки и проверки ЭЦП, сократить длины параметров и ключей при сохранении уровня криптографической стойкости.

Алгоритмы выработки и проверки ЭЦП построены на основе усовершенствованной схемы Эль Гамаля (схема Шнорра). При их выполнении используются вычисления в группе точек эллиптической кривой над конечным простым полем. В стандарте определяются алгоритмы генерации и проверки параметров, описывающих искомую группу. Определены также два алгоритма: генерации пары ключей (личного и открытого) и проверки открытого ключа. Алгоритмы проверки параметров эллиптической кривой и открытого ключа следует применять в тех случаях, когда отсутствует гарантия их математической корректности. Такая гарантия обеспечивает достоверность выводов о стойкости алгоритмов ЭЦП. Вместе с тем алгоритм проверки открытого ключа не гарантирует, что ключ действительно принадлежит определенной стороне или что сторона знает соответствующий личный ключ. Проверка знания личного ключа, удостоверение принадлежности открытого ключа и проверка такой принадлежности реализуются с помощью дополнительных методов и средств, в совокупности называемых инфраструктурой открытых ключей.

Параметры эллиптической кривой, личный и открытый ключи могут быть использованы не только для контроля целостности и подлинности, но и для обеспечения конфиденциальности защищаемой информации. В стандарте определяются алгоритмы транспорта ключа, предназначенные для защищенной передачи ключей и других секретных данных между двумя сторонами. С помощью транспортируемого ключа стороны могут выполнять шифрование или другие криптографические операции. Для реализации транспорта отправитель выполняет алгоритм создания токена ключа и передает полученный токен получателю. Получатель выполняет алгоритм разбора токена и восстанавливает транспортируемый ключ. При создании токена отправитель использует открытый ключ получателя. При разборе токена получатель использует свой личный ключ.

В приложении Б СТБ 34.101.45-2013 приводятся стандартные наборы параметров эллиптической кривой, которые были получены с помощью соответствующего алгоритма генерации и могут быть использованы напрямую, без повторного построения.

Алгоритмы ЭЦП построены так, что злоумышленнику вычислительно трудно решить задачу подделки ЭЦП. В этой задаче у злоумышленника имеются параметры эллиптической кривой и открытый ключ ЭЦП. Злоумышленник не знает личный ключ, но может передавать для подписи на нем произвольные сообщения, получать и анализировать результаты. Ему требуется создать корректную ЭЦП к любому сообщению, отличному от ранее подписанных.

Стойкость алгоритмов ЭЦП определяется уровнем $L \in \{128,192,\ 256\}$. На уровне L для подделки ЭЦП злоумышленнику требуется выполнить порядка 2^L операций. Стойкость основывается на сложности дискретного логарифмирования в группе точек эллиптической кривой и на стойкости используемых функций хэширования.

Уровень L определяет длины параметров, ключей, подписей, а также быстродействие алгоритмов ЭЦП.

5.1.1 Параметры эллиптической кривой

Модуль p. Используется простое число *p,* которое удовлетворяет условиям: $2^{2l_1} < p < 2^{2l}$, $p \equiv 3(\mathrm{mod}\,4)$. Модуль определяет поле F_p, над которым строится эллиптическая кривая. Можно использовать произвольное допустимое *p,* в том числе простое специального вида (например, $p = 2^{2l} - c$, где c – малое натуральное число).

Коэффициенты a, b. Используются числа $a, b \in F_p$, которые удовлетворяют условиям: $a \neq 0$, $(b/p) = 1$, $4a^3 + 27b^2 \neq 0(\mathrm{mod}\,p)$. Коэффициенты a, b вместе с модулем p определяют группу точек эллиптической кривой $E_{a,b}(F_p)$.

Параметр seed. Числа p и a выбираются, а b строится по ним. При построении b используется дополнительный параметр $seed \in \{0,1\}^{64}$, который может быть выбран произвольным образом.

Порядок q. После построения группы $E_{a,b}(F_p)$ рассчитывается ее порядок $q = |E_{a,b}(F_p)|$. Выбирается группа, порядок которой удовлетворяет следующим ограничениям: q – простое, $2^{2l_1} < q < 2^{2l}$, $q \neq p$, q не делит числа вида $p^m - 1$ для $m = 1, 2, ..., 50$.

Базовая точка G. Используется базовая точка $G \in E^*_{a,b}(F_P)$ вида $G = (0, y_G)$, где $y_G = b^{(p+1)/4} \bmod p$. Кратные $G, 2G, ..., (q-1)G$ базовой точки проходят все элементы $E^*_{a,b}(F_p)$, а $qG = O$.

5.1.2 Ключи

Личным ключом является число $d \in \{1, 2, ..., q-1\}$. По этому ключу определяется открытый ключ $Q \in E^*_{a,b}(F_p)$. Алгоритмы генерации личного и открытого ключей, алгоритм проверки открытого ключа также определяются СТБ 34.101.45-2013. При хранении и распространении должны обеспечиваться конфиденциальность и контроль целостности личного ключа, а также контроль целостности открытого ключа. Применяемые методы управления ключами должны гарантировать принадлежность открытого ключа стороне, подпись которой проверяется, и владение данной стороной соответствующим личным ключом. Кроме личного ключа *d,* в алгоритме выработки ЭЦП используется одноразовый личный ключ $k \in \{1, 2, ..., q-1\}$. Также он применяется в алгоритме создания токена ключа. Одноразовые личные ключи должны вырабатываться без возможности предсказания и уничтожаться после использования.

Для создания личных ключей может быть использован физический генератор случайных чисел, удовлетворяющий техническим нормативным правовым актам (ТНПА), или алгоритм генерации псевдослучайных чисел, определенный в СТБ 34.101.47-2017 [8] или в другом ТНПА. Входные данные алгоритма должны включать секретный ключ, доступный только владельцу личного

ключа, и уникальную синхропосылку. Длина ключа алгоритма генерации должна быть не меньше *l*.

5.1.3 Функция хэширования

В алгоритмах выработки и проверки ЭЦП используется функция хэширования h, которая ставит в соответствие подписываемому или проверяемому сообщению X его хэш – значение $h(X)$.

На уровне стойкости L должна использоваться функция h, значениями которой являются двоичные слова длиной $2L$. Например, при $L=128$ в качестве h можно выбрать функцию, заданную алгоритмом belt-hash.

В СТБ 34.101.45-2013 используется алгоритм хэширования belt-hash**,** определенный в СТБ 34.101.31-2011. Входными данными алгоритма является слово $X \in \{0,1\}^*$, выходными – хэш-значение $Y \in \{0,1\}^{256}$.

5.1.4 Выработка и проверка электронной цифровой подписи

Входные и выходные данные. Входными данными алгоритмов ЭЦП являются параметры *p, a, b, q, G,* которые описывают группу точек эллиптической кривой. По модулю p определяется уровень стойкости l как минимальное натуральное число, для которого $p<2^{2l}$.

Кроме параметров эллиптической кривой, входными данными алгоритма выработки ЭЦП являются сообщение $X \in \{0,1\}^*$ и личный ключ $d \in \{1,2,...,q-1\}$, выходными данными – слово $S \in \{0,1\}^{3l}$ – подпись *X*.

Кроме параметров эллиптической кривой, входными данными алгоритма проверки ЭЦП являются сообщение $X \in \{0,1\}^*$, подпись $S \in \{0,1\}^*$ и открытый ключ $Q \in E^*_{a,b}(F_p)$. Выходными данными алгоритма проверки ЭЦП является ответ ДА или НЕТ. Ответ ДА означает, что S является корректной подписью X, ответ НЕТ – обратное.

Вспомогательные преобразования и переменные:

1) Одноразовый личный ключ *k.* При выработке ЭЦП используется одноразовый личный ключ $k \in \{1,2,...,q-1\}$.

2) Переменная *H*. Используется переменная $H \in \{0,1\}^{2l}$.

3) Переменная *R*. Используется переменная $R \in E_{a,b}(F_p)$.

4) Переменная *t*. При проверке ЭЦП используется переменная $t \in \{0,1\}^l$.

Алгоритм выработки ЭЦП. ЭЦП составляется из частей $S_0 \in \{0,1\}^l$ и $S_1 \in \{0,1\}^{2l}$. Выработка ЭЦП состоит в выполнении следующих шагов:

1) Установить $H \leftarrow h(X)$.

2) Выработать $k \xleftarrow{R} \{1,2,...,q-1\}$.

3) Установить $R \leftarrow kG$.

4) Установить $S_0 \leftarrow \left\langle \text{belt-hash}(OID(h) \| \langle R \rangle_{2l} \| H) \right\rangle_L$.

5) Установить $S_1 \leftarrow \left\langle (k - \overline{H} - (\overline{S_0} + 2^l)d) \bmod q \right\rangle_{2l}$.

6) Установить $S \leftarrow S_0 \| S_1$.

7) Возвратить S.

Алгоритм проверки ЭЦП состоит из следующих шагов:

1) Если $|S| \neq 3l$, то возвратить НЕТ.

2) Представить S в виде $S = S_0 \| S_1$, где $S_0 \in \{0,1\}^l$, $S_1 \in \{0,1\}^{2l}$.

3) Если $\overline{S_1} \geq q$, то возвратить НЕТ.

4) Установить $H \leftarrow h(X)$.

5) Установить $R \leftarrow ((\overline{S_1} + \overline{H}) \bmod q)\, G + (\overline{S_o} + 2^l)Q$.

6) Если $R = 0$, то возвратить НЕТ.

7) Установить $t \leftarrow \left\langle \text{belt-hash}(OID(h) \| \langle R \rangle_{2l} \| H) \right\rangle_l$.

8) Если $S_0 \neq t$, то возвратить НЕТ.

9) Возвратить ДА.

5.2 СТБ 1176.2-99. Процедуры выработки и проверки электронной цифровой подписи

СТБ 1176.2-99 определяет процедуры выработки и проверки ЭЦП на базе асимметричного криптографического алгоритма с применением функции хэширования, соответствующей СТБ 1176.1-99 [9]. ЭЦП позволяет обеспечить целостность сообщений (документов), передаваемых в системах обработки информации различного назначения, с гарантированной идентификацией ее автора (лица, подписавшего документ). Для выработки ЭЦП применяется личный ключ подписи, для проверки – открытый ключ проверки подписи. Используемые в процедурах ЭЦП числа *p, l, q, r* и *a* являются открытыми параметрами и генерируются в соответствии с пунктом 5.2.3. Данные числа обладают следующими свойствами:

– *p, q* – простые числа;

– *q* делит p_1;

– *l* является длиной записи числа *p* в системе счисления по основанию 2;

– *r* является длиной записи числа *q* в системе счисления по основанию 2;

– порядок *a* в группе, определяемой множеством B_p и операцией $\circ$, равен *q*.

Конкретный выбор значений указанных параметров должен быть единым в рамках автоматизированной системы или выделенной группы пользователей.

Процедуры ЭЦП допускают как программную, так и аппаратную реализацию.

5.2.1 Процедура выработки ЭЦП

Исходные данные и параметры. В процедуре выработки ЭЦП в качестве исходных параметров используются числа *p, l, q, r* и *a*, генерируемые процедурами, описанными в пункте 5.2.3, и являющиеся открытыми параметрами.

Исходными данными для процедуры выработки ЭЦП являются:

– M – последовательность чисел $M = (m_1, m_2, \ldots, m_z)$, где $m_i \in Z(8)$ для $i = 1, 2, \ldots, z$ и z – длина последовательности M;

– x – целое число, $0 < x < q$, являющееся личным ключом подписи и хранящееся в тайне, где q – параметр, определяемый в пункте 5.2.3.

Используемые переменные. В процедуре выработки ЭЦП используются следующие переменные:

– k – целое число, $1 < k < q$, которое хранится в тайне и должно быть уничтожено сразу после использования;

– t – целое число, $0 < t < p$;

– M_i – последовательность чисел из $Z(8)$, имеющая конечную длину;

– U – целое число, $0 \le U < 2^{r-1}$;

– V – целое число, $0 \le V < q$;

– S – целое число, $0 < S < 2^{2r-1}$.

Алгоритм выработки ЭЦП. Алгоритм выработки ЭЦП включает в себя следующие шаги:

1) Выработать с помощью физического датчика случайных чисел или псевдослучайным методом с использованием секретных параметров число $k (1 < k < q)$.

2) $t := a^{(k)}$.

3) Представить число t в виде разложения по основанию 2^8: $t = \sum_{i=0}^{n-1} t_i \cdot (2^8)^i$.

4) $M_t := (t_0, t_1, \ldots, t_{n-1}, m_1, \ldots, m_z)$.

5) $U := h(M_t)$. Если $U = 0$, то перейти к шагу 1.

6) $V := (k - x \cdot U) \bmod q$. Если $V = 0$, то перейти к шагу 1.

7) $S := U \cdot 2^r + V$.

ЭЦП последовательности M является число S.

5.2.2 Процедура проверки ЭЦП

Исходные данные и параметры. В процедуре проверки ЭЦП используются те же исходные параметры, что и в процедуре выработки: числа *p, l, q, r* и *a*, генерируемые в результате выполнения процедур, описанных в пункте 5.2.3, и являющиеся открытыми параметрами.

Исходными данными для процедуры проверки ЭЦП являются:

– M – последовательность чисел $M = (m_1, m_2, \ldots, m_z)$, где $m_i \in Z(8)$ для $i = 1, 2, \ldots, z$ и z – длина последовательности M;

– S – ЭЦП последовательности M;

– $y = a^{(x)}$ – открытый ключ проверки подписи лица, подписавшего S, где x – личный ключ подписи этого лица.

Используемые переменные. В процедуре проверки ЭЦП используются следующие переменные:

– t – целое число, $0 < t < p$;

– M_t – последовательность чисел из $Z(8)$, имеющая конечную длину;

– U – целое число, $0 \le U < 2^{r-1}$;

– V – целое число, $0 \le V < q$;

– W – целое число, $0 \le W < 2^{r-1}$.

Алгоритм проверки ЭЦП включает в себя следующие шаги:

1) $V := S \bmod 2^r$.

2) $U := (S - V)/2^r$.

3) Проверить условия $0 < U < 2^{r-1}$ и $0 < V < q$. Если хотя бы одно из этих условий не выполнено, то ЭЦП считается недействительной и работа алгоритма завершается.

4) $t = a^{(V)} \circ y^{(U)}$.

5) Представить число t в виде разложения по основанию 2^8: $t = \sum_{i=0}^{n-1} t_i \cdot (2^8)^i$.

6) $M_t := (t_0, t_1, \ldots, t_{n1}, m_1, \ldots, m_z)$.

7) $W := h(M_t)$.

8) Проверить условие $W = U$. При совпадении значений W и U принимается решение о том, что ЭЦП была создана с помощью личного ключа подписи x, связанного с открытым ключом проверки подписи y, а также ЭЦП и последовательность M не были изменены с момента их создания. В противном случае подпись считается недействительной.

5.2.3 Процедуры получения параметров

Выбор r и l. Выбор значений параметров r и l производится в зависимости от необходимого уровня криптографической стойкости в соответствии с таблицей 5.1 (уровни криптографической стойкости приведены в порядке возрастания стойкости).

Процедуры генерации параметров p, q и a приведены в подразделах 7.2 и 7.3 СТБ 1176.2-99.

Таблица 5.1 – Уровни криптографической стойкости

Уровень криптографической стойкости	r	l
1	143	638
2	154	766
3	175	1022
4	182	1118
5	195	1310
6	208	1534
7	222	1790
8	235	2046
9	249	2334
10	257	2462

6 КОНТРОЛЬ ЦЕЛОСТНОСТИ

Одним из важных средств, предназначенных для защиты данных в инфокоммуникационных системах (ИКС) от несанкционированного доступа (НСД), является обнаружение искажений (добавлений, подмены, исключения) в информации, хранящейся в ЭВМ или передаваемой по каналам связи. Причем под данными понимаются как информация, так и программы. Следовательно, вносимые несанкционированные изменения могут нарушить (изменить) порядок обработки информации в ЭВМ (например, исключить отдельные проверки со стороны средств защиты), исказить собственно данные системы защиты, что может привести к НСД, а также изменить информацию в базе данных (это приводит к навязыванию системе ложной информации).

Суть метода контроля целостности заключается в следующем. После того как данные или программы полностью проверены, они подвергаются специальной обработке, заключающейся в получении взвешенной контрольной суммы. Затем целостность данных и программ может проверяться в процессе случайных повторных запусков указанной программы обработки и сравнения полученных контрольных сумм с исходной. Термин «контрольная сумма» используется здесь условно, так как это может быть результат любого преобразования, необязательно суммирования.

Стороны могут организовать контроль целостности сообщений как с использованием механизмов имитозащиты, так и путем применения хэш-функций и сравнения их хэш-значений с достоверными контрольными хэш-значениями. Изменение сообщения с высокой вероятностью приводит к изменению соответствующего хэш-значения и поэтому хэш-значения могут использоваться вместо самих сообщений, например в системах электронной цифровой подписи. Ниже рассмотрим оба указанных механизма.

6.1 Имитозащита сообщений в ИКС

Механизм обеспечения целостности информации в ИКС реализуется на основе анализа приемной стороной проверочных комбинаций, которые формируются и добавляются к защищаемым сообщениям в соответствии с определенными правилами на передающей стороне. Данные правила устанавливаются в зависимости от используемого метода преобразования сообщения и известны как методы обеспечения имитозащиты с помощью симметричных криптографических алгоритмов.

В модели имитозащиты сообщений представлены четыре участника. Это отправитель (законный пользователь *A*), получатель (законный пользователь *B*), нарушитель в канале связи (*C*) и доверенная сторона (*Д*). Исходное распределение функций между участниками модели следующее.

Задача пользователя *A* заключается в отправке сообщения *X* пользователю *B*, задача пользователя *B* – в получении сообщения *X* и проверке его подлинности. Задачей доверенной стороны является обеспечение функционирования защищенного тракта обмена ключевой системы, например рассылки ключей, вос-

становление после компрометации секретных ключей, запись в своей защищенной памяти различной информации о функционировании сети.

Рассмотрим различные способы обмана (нарушение подлинности сообщения), которые возможны в условиях данной модели имитозащиты сообщений с учетом того, что в действиях возможных нарушителей из числа участников модели *A*, *B* и *C* отсутствует кооперация:

1) Способ C_1^0: нарушитель в канале связи искажает сообщение (возможно после проведения анализа на основе этого и других сообщений), которое пользователь *A* передает пользователю *B* (подмена передаваемого сообщения).

2) Способ C_2^0: нарушитель в канале связи формирует и посылает пользователю *B* сообщение *X* от имени пользователя *A* (имитация передаваемого сообщения).

3) Способ C_3^0: нарушитель в канале связи повторяет ранее переданное сообщение, которое пользователь *A* посылал пользователю *B* (повтор ранее переданного сообщения).

Имитозащита сообщений подразумевает защиту против нарушителя в канале связи, т. е. против способов обмана C_1^0, C_2^0, C_3^0.

В зарубежной литературе описаны два метода классической имитозащиты: MAC (Message Anthentication Code – код аутентификации сообщений) и MDC (Manipulation Detection Code – код обнаружения имитаций).

Проверочная комбинация может быть вычислена как криптографическая функция от сообщения *X* и секретного ключа *K*, известного только взаимодействующим пользователям (в нашем случае пользователям *A* и *B*) и, возможно, доверенной стороне.

В случае метода MDC на длину *n* проверочной комбинации накладывается условие $n \geq 128$. Это вызвано необходимостью обеспечить защиту от метода криптоанализа под названием «день рождения».

В случае метода MAC использование секретного ключа при вычислении проверочной комбинации дает следующее ограничение на длину проверочной комбинации: $n \geq 32$.

6.1.1 Метод MAC

В этом методе пользователь ***A*** вычисляет проверочную комбинацию (Checksum – контрольную комбинацию – имитовставку) $Y = F(k, \overline{X})$ и отправляет пользователю *B* сообщение в открытом (или зашифрованном виде), т. е. в виде $\overline{X} = (X, Y)$ (или $Z = E_k(\overline{X})$, где под E_k понимается зашифрование на секретном ключе *K*).

Заметим, что в состав сообщения *X* входят как данные, подлежащие передаче, так и служебная информация, например, адрес отправителя, адрес получателя, дата, время отправки, номер сообщения.

Пользователь *B* получает $\overline{X}$, вычисляет $F(k, \overline{X})$, затем удостоверяется, что результат этого вычисления совпадает с полученным *Y*. Данный метод

нашел свое применение в открытых системах Федеральной резервной системы EFT в США, где контрольная комбинация вырабатывается на основе использования классического алгоритма DES в режиме со сцеплением блоков.

Проверочная комбинация может быть вычислена на основе использования так называемой необратимой функции сжатия сообщения X (хэш-функции).

6.1.2 Метод MDC

В этом методе пользователь A вычисляет проверочную комбинацию $Y = \Phi^*(\overline{X})$, зашифровывает сообщение вместе с ней и отправляет пользователю B шифротекст $Z = E_k(\overline{X})$, где $\overline{X} = (X, Y)$. Пользователь B расшифровывает полученный шифротекст Z, вычисляет $\Phi^*(\overline{X})$, сверяет результат этого вычисления с полученной проверочной комбинацией Y и по результату данного сравнения делает вывод о подлинности полученного сообщения. Этот метод был предложен национальным Бюро стандартов США.

6.1.3 ГОСТ 28147-89. Режим выработки имитовставки

Имитовставка – это блок из p бит, которые вырабатываются по определенному правилу из открытых данных с использованием ключа и затем добавляются к зашифрованным данным для обеспечения защиты системы шифрованной связи от навязывания ложных данных.

В ГОСТ 28147-89 определяется процесс выработки имитовставки, который единообразен для любого из режимов шифрования данных. Имитовставка $И_p$ вырабатывается из блоков открытых данных либо перед шифрованием всего сообщения, либо параллельно с шифрованием по блокам. Значение параметра p (число двоичных разрядов в имитовставке) определяется криптографическими требованиями с учетом того, что вероятность навязывания ложных помех равна $1/2^p$.

Для выработки имитовставки открытые данные представляют в виде последовательности 64-разрядных блоков $T_0^{(i)}$, $i = 1 \ldots m$. Первый блок открытых данных $T_0^{(1)}$ подвергают преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены. В качестве ключа для выработки имитовставки используют ключ длиной 256 бит, по которому шифруют данные. Полученное после 16 циклов 64-разрядное число суммируют по модулю 2 со вторым блоком открытых данных $T_0^{(2)}$, а результат суммирования снова подвергают преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены. Полученное 64-разрядное число суммируют по модулю 2 с третьим блоком $T_0^{(3)}$ и снова подвергают описанному выше преобразованию, получая 64-разрядное число.

Последний блок $T_0^{(m)}$ (при необходимости дополненный нулями до полного 64-разрядного блока) суммируют по модулю 2 с результатом вычислений на шаге m –1, после чего зашифровывают в режиме простой замены, используя

преобразование по процедуре 16 циклов в режиме простой замены. Из полученного 64-разрядного числа выбирают отрезок $И_p$ (имитовставку) длиной p бит, которая передается по каналу связи в конце зашифрованных данных, т. е. $T_ш^{(1)}$, $T_ш^{(2)}$, ..., $T_ш^{(m)}$, $И_p$.

Поступившие к получателю зашифрованные данные $T_ш^{(1)}, T_ш^{(2)}, ..., T_ш^{(m)}$ расшифровываются, и из полученных блоков открытых данных $T_0^{(1)}, T_0^{(2)}, ..., T_0^{(m)}$ аналогичным образом вырабатывается имитовставка $И'_p$. Эта имитовставка $И'_p$ сравнивается с имитовставкой $И_p$, полученной вместе с зашифрованными данными из канала связи. В случае несовпадения имитовставок полученные при расшифровании блоки открытых данных $T_0^{(1)}, T_0^{(2)}, ..., T_0^{(m)}$ считают ложными.

6.1.4 СТБ 34.101.31-2011. Выработка имитовставки

Входными данными алгоритма выработки имитовставки являются сообщение $X \in \{0,1\}^*$ и ключ $\theta \in \{0,1\}^{256}$, выходными данными – слово $T \in \{0,1\}^{64}$ – имитовставка X на ключе θ.

Входное сообщение X ненулевой длины записывается в виде

$$X = X_1 \| X_2 \| ... \| X_n, \; |X_1| = |X_2| = ... = |X_{n-1}| = 128, \; 0 < |X_n| \le 128.$$

Если X – пустое слово, то $n = 1$ и $|X_1| = 0$.

В алгоритме используются следующие вспомогательные преобразования и переменные: преобразования φ_1 и φ_2, отображение ψ и переменные r и s.

Преобразования $\varphi_1, \varphi_2 : \{0,1\}^{128} \to \{0,1\}^{128}$ действуют на слово $u = u_1 \| u_2 \| u_3 \| u_4$, $u_i \in \{0,1\}^{32}$ по правилам:

$$\varphi_1(u) = u_2 \| u_3 \| u_4 \| (u_1 \oplus u_2),$$
$$\varphi_2(u) = (u_1 \oplus u_4) \| u_1 \| u_2 \| u_3.$$

Отображение ψ ставится в соответствие двоичному слову u, длина которого меньше 128, слово $\psi(u) = u \| 1 \| 0^{127-|u|}$ длиной 128.

Используются переменные r и s со значениями из $\{0,1\}^{128}$.

Формирование имитовставки сообщения X на ключе θ состоит в выполнении следующих шагов:

1) Установить $s \leftarrow 0^{128}$, $r \leftarrow F_\theta(s)$.

2) Для $i = 1,2,...,n-1$ выполнить $s \leftarrow F_\theta(s \oplus X_i)$.

3) Если $|X_n| = 128$, то $s \leftarrow s \oplus X_n \oplus \varphi_1(r)$, иначе $s \leftarrow s \oplus \psi(X_n) \oplus \varphi_2(r)$.

4) Установить $T \leftarrow L_{64}(F_\theta(s))$.

5) Возвратить T.

6.2 Контроль целостности сообщений на основе функций хэширования

Функция хэширования – это детерминированная функция h, отображающая строку битов произвольной длины в хэшированное значение $|h|$, представляющее собой строку битов фиксированной длины. Функция хэширования должна обладать следующими свойствами.

1) При любом аргументе x хэшированное значение $h(x)$ должно быть неотличимым с вычислительной точки зрения от строки битов, равномерно распределенных в интервале $[0, 2^{|h|})$.

2) Поиск двух величин x и y, удовлетворяющих условиям $x \neq y$ и $h(x) = h(y)$, должен представлять собой неразрешимую задачу. Для того чтобы это условие выполнялось, необходимо, чтобы область значений функции h была большой, а число $|h|$ – не меньше 128.

3) Задача вычисления входной строки x по заданному хэшированному значению $h = h(x)$ должна быть неразрешимой вычислительной задачей. В этом случае также необходимо, чтобы область значений функции h была достаточно большой.

6.2.1 СТБ 34.101.31-2011. Алгоритм хэширования

Входными данными алгоритма хэширования является сообщение $X \in \{0,1\}^*$, выходными данными – слово $Y \in \{0,1\}^{256}$ – хэш-значение сообщения X. К входному сообщению X предварительно добавляется t нулевых символов, где t – минимальное неотрицательное целое число, такое, что $|X|+t$ кратно 256. Полученное слово записывается в виде

$$X \| 0^t = X_1 \| X_2 \| \ldots \| X_n, \; |X_1| = |X_2| = \ldots = |X_n| = 256.$$

В алгоритме используются следующие вспомогательные преобразования и переменные: отображения σ_1, σ_2 и переменные s, h.

Отображения $\sigma_2 \colon \{0,1\}^{512} \to \{0,1\}^{128}$ и $\sigma_2 \colon \{0,1\}^{512} \to \{0,1\}^{256}$ действуют на слово $u = u_1 \| u_2 \| u_3 \| u_4$, $u_i \in \{0,1\}^{128}$ по правилам:

$$\sigma_1(u) = F_{u_1 \| u_2}(u_3 \oplus u_4) \oplus u_3 \oplus u_4,$$

$$\sigma_2(u) = (F_{\theta_1}(u_1) \oplus u_1) \| (F_{\theta_2}(u_2) \oplus u_2),$$

где $\theta_1 = \sigma_1(u) \| u_4$;

$\theta_2 = (\sigma_1(u) \oplus 1^{128}) \| u_3$;

F_K – зашифрование блока данных на ключе K.

Переменная s используется со значениями из $\{0,1\}^{128}$, а переменная h – со значениями из $\{0,1\}^{256}$.

Хэширование сообщения X состоит в выполнении следующих шагов:

1) Установить $s \leftarrow 0^{128}$.

2) Установить

$h \leftarrow B194BAC80A08F53B366D008E584A5DE48504FA9D1BB6C7AC252E72C202FDCE0D_{16}$,
где присваиваемое значение определяется последовательными (слева направо и сверху вниз) элементами первых двух строк таблицы подстановки, используемой для шифрования блока данных.

3) Для $i = 1,2,...,n$ выполнить:

а) $s \leftarrow s \oplus \sigma_1(X_i \| h)$;

б) $h \leftarrow \sigma_2(X_i \| h)$.

4) Установить $Y \leftarrow \sigma_2(\langle |X| \rangle_{128} \| s \| h)$.

5) Возвратить Y.

6.2.2 СТБ 34.101.77-2016. Алгоритмы хэширования

Настоящий стандарт определяет семейство криптографических алгоритмов хэширования, предназначенных для контроля целостности и необратимого сжатия данных. Обрабатываемыми данными являются двоичные слова (сообщения). Алгоритм хэширования по сообщению произвольной длины строит хэш-значение – слово фиксированной длины.

Алгоритмы хэширования построены по схеме sponge (губка), ядром которой является шаговая функция, определяющая сложное биективное преобразование слов большой длины. В настоящем стандарте шаговая функция действует на слова длиной 1536. Действие задается алгоритмом bash-f, определенным в 6.2.2.2. Шаговая функция bash-f имеет самостоятельное значение и может использоваться за пределами настоящего стандарта для построения других криптографических алгоритмов.

Алгоритмы хэширования СТБ 34.101.77-2016 отличаются уровнем стойкости *l*. Это натуральное число, кратное 16 и не превосходящее 256. Алгоритм уровня *l* вычисляет хэш-значения длиной 2*l*, обрабатывая входные слова блоками длиной 1536 – 4*l*. Уровни $l = 128$, $l = 192$ и $l = 256$ являются стандартными, им следует отдавать предпочтение. При выборе *l* следует учитывать, что для определения сообщения с заданным хэш-значением требуется выполнить порядка 2^{2l} операций, а для определения двух различных сообщений с одинаковыми хэш-значениями – порядка 2^l операций. Следует учитывать также, что с ростом *l*, кроме повышения стойкости, снижается быстродействие алгоритмов. В частности, хэширование на уровне $l = 256$ выполняется примерно в два раза медленнее, чем на уровне $l = 128$.

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

В стандарте используются два вспомогательных алгоритма.

6.2.2.1 Алгоритм bash-s. Входными данными алгоритма bash-s являются слова $W_0, W_1, W_2 \in \{0,1\}^{64}$ и числа $m_1, n_1, m_2, n_2 \in \{1, 2, ..., 63\}$, выходными данными – преобразованные слова W_0, W_1, W_2. Используются переменные $T_0, T_1, T_2 \in \{0,1\}^{64}$.

Преобразование слов W_0, W_1, W_2 состоит в выполнении следующих шагов:

1) $T_0 \leftarrow RotHi^{m_1}(W_0)$.

2) $W_0 \leftarrow W_0 \oplus W_1 \oplus W_2$.

3) $T_1 \leftarrow W_1 \oplus RotHi^{n_1}(W_0)$.

4) $W_1 \leftarrow T_0 \oplus T_1$.

5) $W_2 \leftarrow W_2 \oplus RotHi^{m_2}(W_2) \oplus RotHi^{n_2}(T_1)$.

6) $T_0 \leftarrow \neg W_1$.

7) $T_1 \leftarrow W_0 \vee W_2$.

8) $T_2 \leftarrow W_0 \wedge W_1$.

9) $T_0 \leftarrow T_0 \vee W_1$.

10) $W_1 \leftarrow W_1 \oplus T_1$.

11) $W_2 \leftarrow W_2 \oplus T_2$.

12) $W_0 \leftarrow W_0 \oplus T_0$.

13) Возвратить (W_0, W_1, W_2).

6.2.2.2 Алгоритм bash-f. Входными данными алгоритма bash-f является слово $S \in \{0,1\}^{1536}$, выходными данными – преобразованное слово *S*, которое записывается в виде $S = S_0 \| S_1 \| ... \| S_{23}$, $S_i \in \{0,1\}^{64}$. Используются переменные $C \in \{0,1\}^{64}$ и $m_1, n_1, m_2, n_2 \in \{1, 2, ..., 63\}$.

Преобразование слова *S* состоит в выполнении следующих шагов:

1) $C \leftarrow B194BAC80A08F53B_{16}$.

2) Для $i = 1, 2, ..., 24$ выполнить:

а) $(m_1, n_1, m_2, n_2) \leftarrow (8, 53, 14, 1)$;

б) для $j = 0, 1, ..., 7$:

– $(S_j, S_{8+j}, S_{16+j}) \leftarrow$ bash-s$(S_j, S_{8+j}, S_{16+j}, m_1, n_1, m_2, n_2)$;

– $(m_1, n_1, m_2, n_2) \leftarrow (7m_1 \bmod 64, 7n_1 \bmod 64, 7m_2 \bmod 64, 7n_2 \bmod 64)$;

в) $S \leftarrow S_{15} \| S_{10} \| S_9 \| S_{12} \| S_{11} \| S_{14} \| S_{13} \| S_8 \| S_{17} \| S_{16} \| S_{19} \| S_{18} \|$
$\| S_{21} \| S_{20} \| S_{23} \| S_{22} \| S_6 \| S_3 \| S_0 \| S_5 \| S_2 \| S_7 \| S_4 \| S_1$;

г) $S_{23} \leftarrow S_{23} \oplus C$;

д) если $\overline{C}$ – четное, то $C \leftarrow ShLo(C)$, иначе $C \leftarrow ShLo(C) \oplus \text{AED8E07F99E12BDC}_{16}$.

3) Возвратить *S*.

В алгоритме хэширования входными данными алгоритма хэширования уровня стойкости *l* является сообщение $X \in \{0,1\}^*$, выходными данными – слово $Y \in \{0,1\}^{2l}$ – хэш-значение сообщения *X*, для вычисления которого используются алгоритм bash-f и переменная $S \in \{0,1\}^{1536}$.

Хэширование сообщения X на уровне стойкости l состоит в выполнении следующих шагов:

1) Дописать к X сначала слово 01, а затем t символов 0, где t – минимальное неотрицательное целое, для которого $|X|+2+t$ | кратно $1536-4l$.

2) Полученное слово $X\|1\|0^t$ записать в виде $X_1\|X_2\|...\|X_n$, $X \in \{0,1\}^{1536-4l}$

3) $S \leftarrow 0^{1472} \| \langle l/4 \rangle_{64}$.

4) Для $i=1,2,...,n$ выполнить:

а) $Lo_{1536-4l}(S) \leftarrow X_i$;

б) $S \leftarrow \text{bash-f}(S)$.

5) Возвратить Y.

6.2.3 СТБ 1176.1-99. Информационная технология. Защита информации. Функция хэширования

Настоящий стандарт определяет функцию хэширования, которая позволяет осуществлять проверку целостности сообщений (документов), передаваемых в системах обработки информации различного назначения, с гарантированной достоверностью. Данная функция хэширования используется в процедурах выработки и проверки электронной цифровой подписи в соответствии с СТБ 1.176.2-99.

Исходными данными для процедуры вычисления значения хэш-функции является последовательность чисел $M=(m_1, m_2,...m_i)$, где $m_i \in Z(8)$ для $i=1,2,...,z$ и z – длина последовательности M.

Параметрами являются числа H и L. Значение H выбирается произвольным образом и фиксируется. Значение L влияет на криптографическую стойкость описанного в настоящем стандарте алгоритма вычисления значения функции хэширования и выбирается максимально возможным. Конкретный выбор значений параметров должен быть единым в рамках автоматизированной системы или выделенной группы пользователей.

Используемые преобразования:

1) Преобразования $\rho_i : Z(512) \to Z(512)$, $i=\overline{0,3}$, для любого $x \in Z(512)$, где $x=\sum_{j=0}^{15} x_j (2^{32})^j$ выполняются следующим образом:

$$p_0(x)=\sum_{j=0}^{14} x_{j+1}(2^{32})^j + ((x_{15} \oplus x_{13} \oplus x_3 \oplus x_0) \oplus 'C_0)(2^{480}),$$

$$p_1(x)=\sum_{j=0}^{14} x_{j+1}(2^{32})^j + ((x_{15} \oplus x_2 \oplus x_0) \oplus 'C_1)(2^{480}),$$

$$p_2(x)=\sum_{j=0}^{14} x_{j+1}(2^{32})^j + ((x_9 \oplus x_4 \oplus x_0) \oplus 'C_2)(2^{480}),$$

$$p_3(x)=\sum_{j=0}^{14} x_{j+1}(2^{32})^j + ((x_{13} \oplus x_3 \oplus x_0) \oplus 'C_3)(2^{480}),$$

где $C_0 = 2\text{BDA732E}$, $C_1 = 3920\text{FE85}$, $C_2 = \text{BC1641F9}$, $C_3 = 75\text{FE243B}$ (в шестнадцатеричной системе счисления).

2) Преобразование $\omega\colon Z(128)\cdot Z(2048)\cdot Z(2048)\cdot Z(2048)\cdot Z(2048)\times$ $\times Z(2048)\cdot Z(2048)\cdot Z(2048)\cdot Z(2048)\to Z(128)$ для любого $X\in Z(128)$, где $X=\sum_{i=0}^{1}X_i(2^{64})^i$ и $X_i=\sum_{j=0}^{7}x_{ij}(2^8)^j$, $i=0,1$, и для любых $Y_i\in Z(2048)$, где $Y_i=\sum_{j=0}^{255}y_{ij}\cdot(2^8)^j$ и $i=\overline{0,7}$, определяется следующим алгоритмом, использующим переменные m и P, где m – целое число, $0\le m<33$, $P\in Z(64)$, $P=\sum_{i=0}^{63}p_i\cdot 2^i$:

Шаг 1. $m:=0$.

Шаг 2. $P:=\sum_{i=0}^{7}y_{ix_{0i}}\cdot(2^8)^i$.

Шаг 3. $P:=\sum_{i=0}^{60}p_i\cdot 2^{i+3}+p_{63}\cdot 2^2+p_{62}\cdot 2+p_{61}$.

Шаг 4. $P:=P\oplus X_1$.

Шаг 5. $X_1:=X_0$, $X_0:=P$.

Шаг 6. $m:=m+1$.

Шаг 7. Проверить условие $m=32$: если условие выполнено, то осуществляется выход из алгоритма, если нет – осуществляется переход к шагу 2.

Значением преобразования $\omega(X,Y_0,Y_1,Y_2,Y_3,Y_4,Y_5,Y_6,Y_7)$ является значение переменной X после завершения работы алгоритма.

3) Преобразование $\xi\colon Z(256)\to Z(256)$ для любого $X\in Z(256)$ такого, что $X=\sum_{i=0}^{7}x_i(2^{32})^i$ выполняется по формуле

$$\xi(X)=\sum_{i=1}^{7}x_{i-1}\cdot(2^{32})^i+(x_0\oplus x_2\oplus x_4\oplus x_7).$$

4) Преобразование $\varphi\colon Z(128)\cdot Z(256)\to Z(256)$ для любого $X\in Z(128)$, такого, что $X=\sum_{i=0}^{3}x_i(2^{32})^i$, и для любого $Y_i\in Z(256)$, такого, что $Y=\sum_{i=0}^{7}y_i\cdot(2^{32})^i$ определяется следующим алгоритмом, использующим переменную $S\in Z(128)$:

Шаг 1. $S:=\sum_{i=0}^{3}y_i\cdot(2^{32})^i$.

Шаг 2. $Y:=\sum_{i=4}^{7}y_i(2^{32})^i+\sum_{i=0}^{3}(x_i\oplus' y_i)(2^{32})^i$.

Шаг 3. $Y:=S\cdot(2^{32})^4+\sum_{i=0}^{3}(y_i\oplus y_{i+4})\cdot(2^{32})^i$.

Значением преобразования $\varphi(X,Y)$ является значение Y после завершения работы алгоритма.

Если длина последовательности M не кратна 32, то она дополняется минимальным количеством нулей таким образом, что $M=(m_1,m_2,...,m_{l+k})=$ $=(m_1,m_2,..,m_l,0,...,0)$ для некоторого неотрицательного целого числа k и $l+k$ кратно 32.

Вычисляется значение $n=(l+k)/32$, причем k полагается равным нулю, если l делится на 32. По последовательности M формируются числа

$M_1, M_2, \ldots, M_n$ таким образом, что $M_i = m_{(i-1)32+1} + m_{(i-1)32+2} \cdot 2^8 + \ldots + m_{i \cdot 32} \cdot 2^{248}$ для $i = 1, 2, \ldots, n$. Значение числа M устанавливается равным единице.

Процедура вычисления значения функции хэширования использует следующие переменные:

1) T_i – $T_i \in Z(2048)$, где $T_i = \begin{cases} \sum_{j=0}^{3} r_{ij} (2^{512})^j, & i = 0, 2, 4, 6, \\ \sum_{j=0}^{63} t_{ij} (2^{32})^j, & i = 1, 3, 5, 7. \end{cases}$ Начальные значения коэффициентов t_{ij} в шестнадцатеричной системе счисления определяются таблицей 5.1 СТБ 1176.1-99.

2) V – $V \in Z(512)$, $V = \sum_{i=0}^{15} v_i (2^{32})^i$. Начальные значения коэффициентов v_i в шестнадцатеричной системе счисления определяются таблицей 5.2 СТБ 1176.1-99.

3) K – $K \in Z(256)$, $K = \sum_{i=0}^{7} k_i \cdot (2^{32})^i = \sum_{i=0}^{1} K_i \cdot (2^{128})^i$.

4) W – $W \in Z(256)$, $W = \sum_{i=0}^{1} W_i \cdot (2^{128})^i$.

5) d – целое число, $0 < d < n + 2$.

Алгоритм вычисления значения функции хэширования включает в себя следующие шаги:

1) $d := 1$.

2) $K := M_d$.

3) $V := \sum_{i=0}^{7} (v_i \oplus' k_i) \cdot (2^{32})^i + \sum_{i=8}^{15} (v_i \oplus' h_{i-8}) \cdot (2^{32})^i$.

4) Для $i = 0, 2, 4, 6$ выполнить следующую последовательность вычислений:

$$V := \rho_0^{29}(V),\ r_{i0} := V;$$
$$V := \rho_1^{18}(V),\ r_{i1} := V;$$
$$V := \rho_2^{19}(V),\ r_{i2} := V;$$
$$V := \rho_3^{17}(V),\ r_{i3} := V.$$

5) $W := K \oplus H$.

6) $W_0 := \omega(W_0, T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7)$.

7) $W_1 := \omega(W_1, T_4, T_1, T_0, T_3, T_6, T_5, T_2, T_7)$.

8) $W := \xi^{31}(W)$.

9) $W := \varphi(H_0, \varphi(H_0, W))$.

10) $W := \varphi(K_0, \varphi(K_0, W))$.

11) $W := \varphi(H_1, \varphi(H_1, W))$.

12) $W := \varphi(K_1, \varphi(K_1, W))$.

13) $H := W$.

14) $d := d + 1$.

15) Проверить условие $d := n + 2$: если условие выполнено, то перейти к шагу 16, если не выполнено – к шагу 2.

16) $h(M) := H \bmod 2^L$.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТБ 34.101.45-2013. Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых. – Введ. 2014–01–01. – Минск : Госстандарт Респ. Беларусь, 2017.

2. СТБ 34.101.77-2016. Информационные технологии и безопасность. Алгоритмы хэширования. – Введ. 2016–10–01. – Минск : Госстандарт Респ. Беларусь, 2016.

3. ГОСТ Р 34.10-2012. Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. – Введ. 2013–01–01. – М. : Стандартинформ, 2018.

4. ГОСТ Р 34.11-2012. Национальный стандарт Российской Федерации. Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования (с поправкой). – Введ. 2013–01–01. – М. : Стандартинформ, 2018.

5. ГОСТ 28147-89. Национальный стандарт Российской Федерации. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. – Введ. 1990–07–01. – М. : ИПК Изд-во стандартов, 1996.

6. СТБ 34.101.31-2011. Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности. – Введ. 2011–07–01. – Минск : Госстандарт Респ. Беларусь, 2014.

7. СТБ 1176.2-99. Информационная технология. Защита информации. Процедуры выработки и проверки электронной цифровой подписи. – Введ. 2000–04–01. – Минск : Госстандарт Респ. Беларусь, 2000.

8. СТБ 34.101.47-2017. Информационные технологии и безопасность. Криптографические алгоритмы генерации псевдослучайных чисел. – Введ. 2017–09–01. – Минск : Госстандарт Респ. Беларусь, 2017.

9. СТБ 1176.1-99. Информационная технология. Защита информации. Функция хэширования. – Введ. 2000–04–01. – Минск : Госстандарт Респ. Беларусь, 2000.

Св. план 2020, поз. 41

Учебное издание

Бобов Михаил Никитич
Шевчук Оксана Геннадьевна

ЗАЩИТА ИНФОРМАЦИИ В СЕТЯХ ИНФОКОММУНИКАЦИЙ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 10.09.2020. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 4,07. Уч.-изд. л. 4,0. Тираж 50 экз. Заказ 58.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск